Database Rolling Upgrade Using Data Guard SQL Apply
Oracle Database 11g and 10gR2

Oracle Maximum Availability Architecture White Paper
July, 2009

Maximum Availability Architecture

Oracle Best Practices for High Availability

ORACLE

Database Rolling Upgrade Using Data Guard SQL Apply
Oracle Database 11g and 10gR2

EXECUTIVE SUMMARY

Oracle Maximum Availability Architecture (MAA) [2] is the Oracle best practices blueprint for implementing Oracle high-availability technologies. The goal of this MAA white paper is to provide best practices for using SQL Apply Rolling Upgrades to execute database upgrades with minimal downtime. Starting with Oracle Database 10g Release 1 (10.1.0.3 and onward), Data Guard SQL Apply (logical standby) can be used to perform a database rolling upgrade (either to a later patch set or to a higher Oracle Database release).

The rolling upgrade process is described in Figure 1. During a rolling upgrade a logical standby database is upgraded from release X to release X′ while production runs on the primary at release X. When the standby upgrade is deemed successful, Data Guard is used to resynchronize the standby with the primary and execute a switchover to transition the standby database to the production role running on the new release X′. While the standby database operates in the production role, the database on the original primary is upgraded to release X′. When the second half of the upgrade is complete, Data Guard again resynchronizes the two databases both operating release X′. 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. Data Guard SQL Apply (logical standby database) is required in order to perform a database rolling upgrade. You can also do a rolling upgrade when starting with a physical standby by temporarily converting the physical standby to a logical standby. If starting with 11g, see the Rolling Database Upgrades for Physical Standby Databases using Transient Logical Standby 11*g* paper for details on doing a rolling upgrade when starting with a physical standby. If starting with 10g, see MetaLink note 300479.1.

Oracle's enterprise eMail application, hosting almost 70,000 worldwide users, reduced downtime for a database upgrade by 96%. Using SQL Apply rolling upgrade compared to the conventional upgrade reduced downtime, from 48 minutes to 2 minutes.

Proof-of-concept tests using SQL Apply 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 SQL Apply rolling upgrade, there would have been no impact on the availability of the primary database. Using SQL Apply to execute the upgrade resulted in a 99% reduction in planned down time.

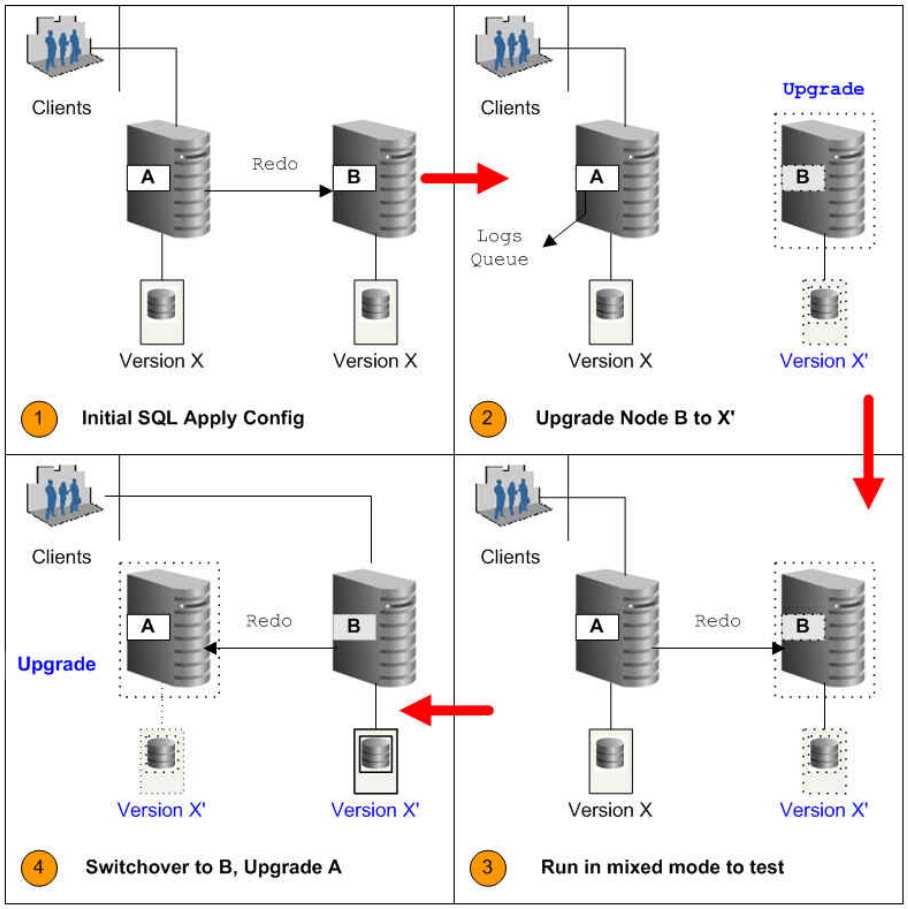

Figure 1 SQL Apply Rolling Upgrade Process

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.

The SQL Apply Rolling Upgrade steps are detailed in Chapter 12 of the Data Guard Concepts & Administration 11g Release 1 guide [5]. Chapter 12 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 plus the 10g Upgrade Companion, MetaLink Note: 466181.1, and/or the 11g Upgrade Companion, MetaLink Note: 601807.1, must be reviewed before performing a SQL Apply rolling upgrade. This paper references 11g documentation but similar documentation exists for 10g. This paper complements

this information with best practices provided below and is not intended to duplicate information already provided in that documentation.

THE ROLLING UPGRADE PROCEDURE

This paper assumes you have a Data Guard configuration similar to the following examples:

- A primary database with a physical standby database and a logical standby database. The logical standby may already exist, or you may create it to use on a temporary basis for a rolling database upgrade (see Figure 2).

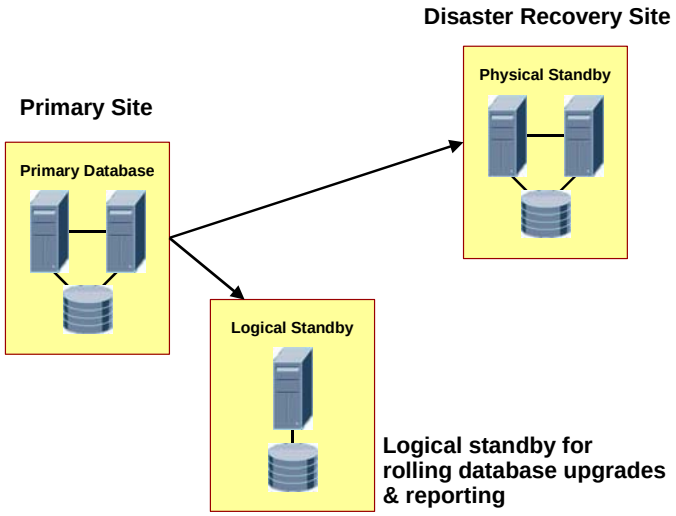

Figure 2 – Multi-Standby Data Guard Configuration

The physical standby is usually the primary disaster recovery failover target. The procedures described in this paper will upgrade the primary database and both standby databases with minimal production downtime. (Note: all standby databases must be configured as destinations of the primary)

- A primary database with an existing logical standby database or where a logical standby database will be created for the purpose of executing a rolling database upgrade.

The SQL Apply Rolling Upgrade procedure is described in Chapter 12 of the Data Guard Concepts & Administration 11g Release 1 guide [5]. There is also an Oracle By Example (OBE) of the SQL Apply Rolling Upgrade at http://www.oracle.com/technology/obe/demos/admin/demos.html. This paper

provides a detailed example of a complete rolling upgrade procedure in Appendix A – Detailed Example of a SQL Apply Rolling Upgrade. Any differences between 10g and 11g are pointed out in the table.

Note that for 10g if the primary database in a rolling upgrade configuration is a Real Application Cluster (RAC) database, ensure that all but one instance are shut down, and the corresponding threads are disabled before initiating a 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 a switchover. You re-enable the threads and start the instances after a 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. **11g does not have this requirement** since the logical standby switchover in 11g handles the multiple threads.

Special Instructions for Multi-Standby Configurations

If your configuration is similar to Figure 2, you must also execute the upgrade of any physical standby databases in your Data Guard configuration. This is done when you are at the point in the rolling upgrade process after the first switchover when the original logical standby database has been upgraded to the new release and is running in the primary role (at the conclusion of Section V in the example in Appendix A of this paper).

Before proceeding with Section VI of Appendix A, (Perform the Upgrade on the Former Primary), you will execute the standard procedure for upgrading a physical standby in-place. This procedure is described in the Data Guard Concepts and Administration guide in "B.2 Upgrading Oracle Database with a Physical Standby Database In Place"[5]. This does not require any additional production downtime because you are at the phase of the rolling upgrade process where production is running at the new release on the system that was originally the logical standby. The physical standby databases in your configuration continue to operate as physical standbys of the original primary. Once provisions have been made to upgrade the physical standbys in-place, the rolling upgrade process is resumed by executing the steps in Section VI of Appendix A, "Perform the Upgrade on the Former Primary",

Note: when reading "B.2 Upgrading Oracle Database with a Physical Standby Database In Place" [5], wherever it refers to "standby database", this is your physical standby, and wherever it refers to "primary", this is the original primary database in your rolling upgrade configuration.

Confirm Data Type Support

It is important to identify unsupported database objects on the primary database before you create a logical standby database because changes made to unsupported data types and tables on the primary database will be automatically skipped by SQL

Apply on the logical standby database. See Appendix C, "*Data Type and DDL Support on a Logical Standby Database*" in the *Oracle Data Guard Concepts and Administration* guide [5] for complete details on how to verify data type support. Appendix C.9 has specific information related to unsupported tables.

Even if unsupported data types are identified, there are cases when rolling upgrade using SQL apply procedure can still be used. The determination has to be made if there is a satisfactory way to handle the unsupported data types. Options for using rolling upgrade when unsupported data types exist are as follows:

- **Suspend or prohibit changes to the unsupported data type objects.** Temporarily suspend or prohibit changes to the unsupported tables for the period of time it takes to perform the upgrade procedure.
- **Use `DBA_LOGSTDBY_EVENTS` with Oracle Data Pump or with the Export/Import utility.** 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, use Oracle Data Pump or the Export/Import utility to import the changed tables to the upgraded databases. These features are described in the *Oracle Database Utilities* documentation [10].

  See Section 12.4 in Oracle Data Guard Concepts and Administration [5] for further information.
- **Use Extended Datatype Support (EDS**). EDS enables SQL Apply to replicate changes to tables that contain some data types not natively supported from one database to another. Beginning with Oracle Database 10g Release 2 (10.2.0.4) Patch Set 3, SQL Apply supports the ability for triggers to fire on the logical standby database, which provides the basis of EDS. For an overview of EDS, see the MAA white paper Extended Data Type Support. For details and examples of using EDS to support data types that are not natively supported by SQL Apply, see MetaLink Note 559353.1.

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 *Oracle Database Upgrade Guide* [7] has an excellent set of steps to follow. In addition to those steps, the following practices are also recommended:

- Review the documentation resources
  - Read the *Oracle Database Upgrade Guide* [7]
  - Read the *Oracle Database Readme* [11]
  - Review known issues and post-release Oracle *MetaLink* notes derived from Note: 161818.1

- 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.

- Prepare a new `ORACLE_HOME`:
  - If you are planning to do a patch or patch set upgrade, then clone a new `ORACLE_HOME` and apply the patch or patch set to it on both the primary and standby systems. This is not mandatory when applying patch sets but it is a recommended practice. Having a separate installation rather than directly applying the patch or patchset 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 patch set apply* (see "Cloning ORACLE_HOME for a Patchset Apply").
  - If you are planning to do a release upgrade, then install the new Oracle Database software into a new `ORACLE_HOME` on both the primary and the standby systems.

- Create a test plan that includes the best practices outlined in the 10g Upgrade Companion in Oracle *MetaLink* Note: 466181.1, and the 11g Upgrade Companion, *MetaLink* Note: 601807.1 plus following these 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

- Understand the fallback restoration options for each upgrade step. The following fallback methods are described in the "Fallback" section later in this paper:
  - Backups
  - Flashback Database
  - Downgrade procedure

- Maintain the same database `COMPATIBLE` initialization parameter setting on all databases in the Data Guard configuration until the upgrade has been completely evaluated. For example, if upgrading from release 11.1.0.6 to 11.1.0.7, then set the `COMPATIBLE` initialization parameter to 11.1.0.6.0.

- Follow the best practices described in the MAA "Oracle Database 10*g* Release 2 Best Practices: Data Guard Switchover and Failover" [4] white paper to optimize Data Guard switchover time.
- Ensure beforehand that you have followed the guidelines for relocating Database Services to the new primary database. See the MAA best practices paper "Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 10*g* Release 2" for more information.

PATCH APPLY/UPGRADE BEST PRACTICES

- If this is a patchset upgrade then 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 E, "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.

  If you are using Oracle Database 10g Release 1 (10.1) then guaranteed restore point cannot be used. Instead, capture the current SCN from the V$DATABASE view and designate this SCN as the flashback point.
  SQL> SELECT CURRENT_SCN FROM V$DATABASE;
- 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

- Note that the switch back to the original configuration can use the PREPARE step in the switchover since at this stage of the upgrade both database are on the same version.
- Consider using outage time during the switchover to also update the COMPATIBLE parameter setting.

- Cleanup any unnecessary guaranteed restore points
- Tune and adjust

GENERAL RESTRICTIONS

Some restrictions apply to using the SQL Apply Rolling Upgrade process.

- The databases must not be part of a Data Guard Broker configuration. Hence, Oracle Enterprise Manager Grid Control does not support the rolling upgrade process. To disable and enable the Broker configuration, see the "Preparation Steps" section.
- 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 determine 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 *X* to new release *X'* requires that the `COMPATIBLE` initialization parameter be set to *X* on both the primary and standby databases. Then, after the upgrade and all assurance tests have passed, you can update the `COMPATIBLE` parameter to the new target release, *X'*.

  NOTE: once you have updated the `COMPATIBLE` parameter to the target database release, you cannot downgrade to an earlier release with flashback database nor the database downgrade procedure
- When you perform the initial switchover in the rolling upgrade process, you cannot not use the `PREPARE TO SWITCHOVER` operation because the primary and standby databases are running different Oracle releases.

FALLBACK BEST PRACTICES

This section describes the following fallback practices:

1. Fallback Backups
2. Flashback Database
3. Downgrades

The Flowchart below (Figure 3, Fallback Options) presents an overview of which fallback options are viable at each step.

Backups

Ensure that you take database and software backups on the primary and the standby databases prior to starting the upgrade process. The software backups should include the `oraInventory` directory tree. Taking software backups are necessary only if they have never been done and if you are applying the patch set

directly to the existing ORACLE_HOME tree rather than applying the patch set to a newly installed separate ORACLE_HOME.

Flashback Database

Flashback Database is a method to achieve very fast point-in-time recovery and is most often the quickest way to fall back to the previous release. Note that data from any transactions that occur after the point in time to which the database is recovered are lost. Flashback Database is 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 is the fastest method to fallback as compared to a restore operation or a database downgrade.

Following switchover, and after the production workload is running on the original standby database, any Flashback operation will result in data loss. Deciding to use Flashback Database versus downgrading the database typically entails a tradeoff between the speed at which you need to return the primary database to the previous release and the amount of data loss that you can tolerate.

NOTE: You can use Flashback Database to fall back to the pre-upgrade release ***only if*** you have not changed the COMPATIBLE database parameter to the target release. See the "General Restrictions" section in this white paper for additional information.

The steps for flashing back the database after a failed upgrade (due to a failure while running the catupgrd.sql script or a DBUA failure) are as follows:

1. Shut down the upgraded database.
2. Startup mount the database under the new ORACLE_HOME
3. Use Flashback Database to return to the guaranteed restore point that was taken prior to the upgrade:

```
SQL> flashback database to restore point PRE_LOGICAL_UPGRADE;
```

4. Shut down the database
5. Startup mount the database under the old ORACLE_HOME
6. Issue the OPEN RESETLOGS command to open the database:

```
SQL> alter database open resetlogs;
```

Note: Guaranteed restore points used in the procedure above are not available for Oracle Database 10g Release 1 (10.1.0.x). 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.

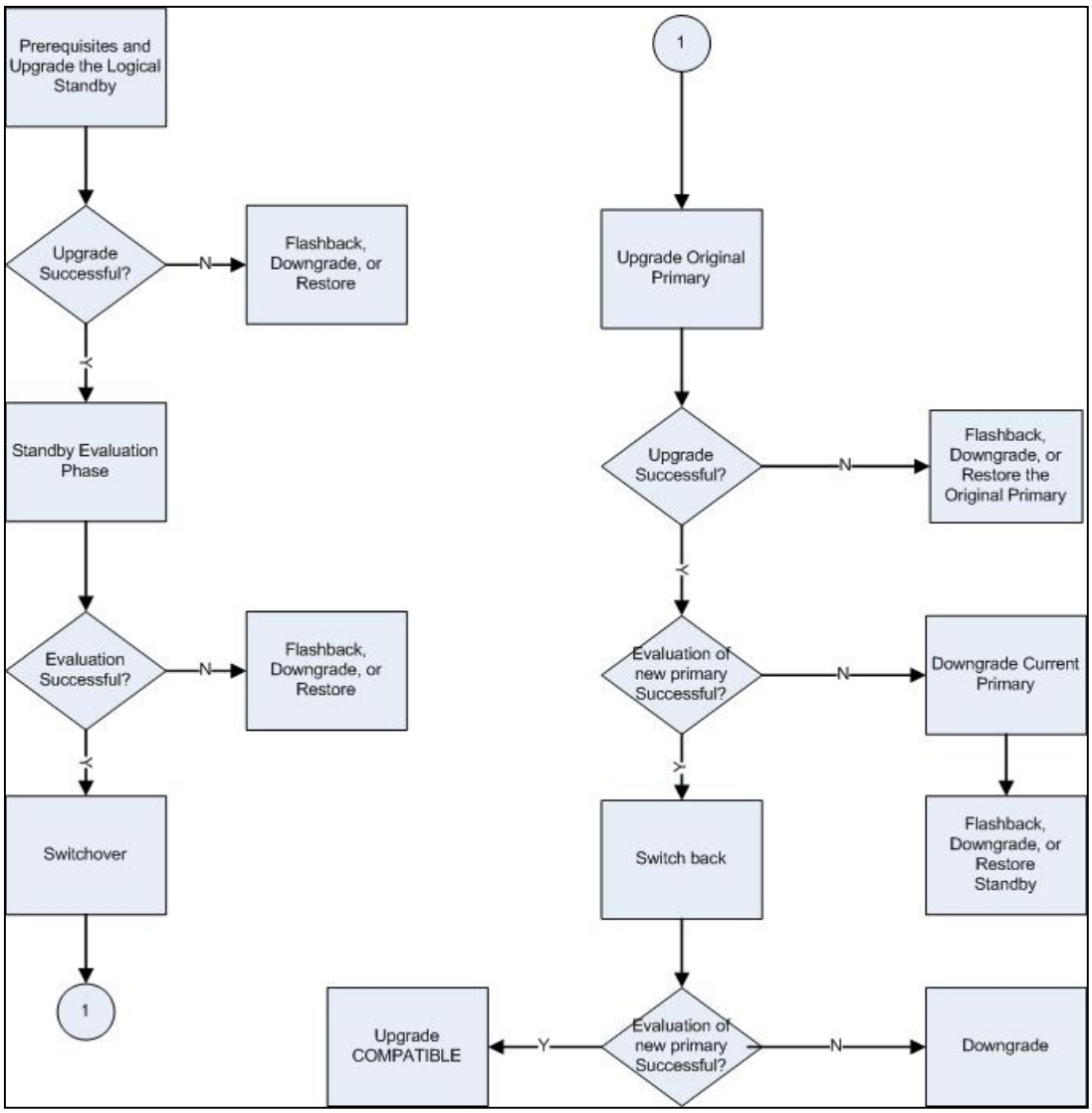

Figure 3 – Fallback Options

Downgrade

To run the downgrade procedure, use the manual downgrade steps documented in Chapter 7 of the *Oracle Database Upgrade Guide 11g Release 1 (11.1)* [7]. Performing a downgrade generally takes as long as the upgrade procedure and backs out the patch set 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 you should perform a downgrade procedure. A prerequisite to downgrading is that the `COMPATIBLE` database parameter must have remained at the release to which you are downgrading .

APPENDIX A - DETAILED EXAMPLE OF A SQL APPLY ROLLING UPGRADE

This example will illustrate the actual steps and commands used for doing a SQL Apply Rolling Upgrade from 11.1.0.6 to 11.1.0.7 using the manual upgrade method (see "Oracle Database Upgrade Guide" [7].) for a RAC primary and a RAC logical standby.

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')

High-level Steps

1. Prerequisites
2. Create a Temporary Archive redo log repository (optional)
3. Perform the Upgrade on the Standby
4. Switchover
5. Install the New ORACLE_HOME and Apply the Patchset on the Original Primary
6. Create a Temporary Archive redo log repository (optional)
7. Perform the Upgrade on the Former Primary
8. Switch back to the original config (optional)
9. Raise the COMPATIBLE Parameter Setting

Detailed Steps

Step	Primary	Standby	Notes
I.	Prerequisites		

1	Optional:
	Turn flashback database on
	`SQL> SHUTDOWN IMMEDIATE`
	`SQL> STARTUP MOUNT`
	`SQL> ALTER DATABASE FLASHBACK ON;`
	`SQL> ALTER DATABASE OPEN;`
		Optional:
		Turn flashback database on
		`SQL> SHUTDOWN IMMEDIATE`
		`SQL> STARTUP MOUNT`
		`SQL> ALTER DATABASE FLASHBACK ON;`
		`SQL> ALTER DATABASE OPEN;`
			If flashback is already on then this step can be skipped. This can also be optional if you just use restore points without flashback database. The database must be in a mount state to turn flashback on.

2		Create a guaranteed restore point `SQL> CREATE RESTORE POINT PRE_UPGRADE GUARANTEE FLASHBACK DATABASE;`	This will be used to flashback the standby site 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. Note that guaranteed restore point is available beginning with Oracle Database 10g Release 2. If you are upgrading from Oracle Database 10g Release 1 – see the note in step 5, below.
3	Ensure any necessary patches are applied as detailed in the "*Patch/Upgrade Preparation Best Practices*" section of this paper	Ensure any necessary patches are applied as detailed in the "*Patch /Upgrade Preparation Best Practices*" section of this paper.	
II.	Create a Temporary Archive redo log repository (optional)		
4		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 can be met in the event of a primary site failure during this step.
III.	Perform the Upgrade on the Standby		
5		Create a guaranteed restore point `SQL> CREATE RESTORE POINT PRE_LOGICAL_UPGRADE GUARANTEE FLASHBACK DATABASE;`	Note that if this an 10.1.0.3 database or a later 10.1 release, then capture the current SCN as the flashback database point. `SQL> SELECT CURRENT_SCN FROM V$DATABASE;`
6		Shutdown the logical standby database	
7		Apply 11.1.0.7 patchset to CRS and ASM	
8		**10g only** Follow MetaLink note 300479.1 to implement the workaround for issue 5236922	This will ensure that there are no Enterprise Manager components installed on the logical standby
9		Install the new 11.1.0.7 Patch set out-of-place	For an out-of-place patchset apply the existing ORACLE_HOME is cloned using the Cloning procedure in Appendix E.

Step	Primary	Standby	Notes
10		Stop Logical Standby Apply `SQL> ALTER DATABASE STOP LOGICAL STANDBY APPLY ;`	**SQL Apply should not be running**
11		If not using DBUA then set cluster_database=false in preparation for upgrade `SQL> ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;`	If using the Database Upgrade Assistant (dbua). dbua is executed from the new ORACLE_HOME. dbua requires CLUSTER_DATABASE='TRUE for a RAC database and it takes care of the OCR updates for you.
12		Upgrade the database	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. **If you did not use DBUA then remember to set cluster_database=true before bouncing the database after the upgrade**.

13		If a temporary log archive repository was used then register logs so they don't have to be resent:
		`SQL> ALTER DATABASE REGISTER LOGFILE '….`
		
		Optionally, you can use the RMAN CATALOG command to catalog the archived redo log repository archivelog destination. e.g.:
		`RMAN> CATALOG START WITH '+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:
			::::::::::::::
			register_logs.sql
			::::::::::::::
			REM The parameter is the SCN from the 'recover standby database command
			set head off feedback off lines 133 pages 0 verify off echo off
			spool register_logs_for_standby.sql
			select 'alter database register logfile ' || chr(39) || name || chr(39) || ';'
			from v$archived_log
			where first_change# >= &1
			/
			spool off

14		Start Logical Standby Apply @ 11.1.0.7 `SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;`	
15			
16		Verify that the logical standby is running correctly	See 10.3 Monitoring a Logical Standby Database [5]

Step	Primary	Standby	Notes
IV.	Switchover		

17	Switchover to Logical Standby on current/original primary
	`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`
	
	`SWITCHOVER_STATUS`
	`--------------------`
	`TO STANDBY`
	
	If the query returns "SESSIONS ACTIVE', then ensure it's ok to shut them down.
	`SELECT SID, PROCESS, PROGRAM FROM V$SESSION WHERE TYPE = 'USER' AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);`
			Make sure this is aligned with MAA Switchover best practices [4]. You cannot do an "ALTER DATABASE PREPARE TO SWITCHOVER" for this switchover due to the mixed versions. This is not supported.

18	**10g Oracle RAC only**
	`$ srvctl stop instance –d chicago –i chicago2`
	`SQL> ALTER DATABASE DISABLE THREAD 2;`
	`SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;`
	`SQL> ALTER SYSTEM ARCHIVE LOG CURRENT;`
			11g logical standby switchover does not require disabling threads or shutting down other instances in an Oracle RAC environment.

19	`SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;`		"`SWITCHOVER TO LOGICAL STANDBY;`" cannot use the "WITH SESSION SHUTDOWN" clause so you need to manually shutdown sessions.
20		Ensure archive logs are received and applied during the primary switchover	
21		Defer Redo `SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER;`	

22		**10g Oracle RAC only**
		`$ srvctl stop instance –d boston –i boston2`
		`SQL> ALTER DATABASE DISABLE THREAD 2;`	

23		Switchover to Primary on current logical standby
		
		`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`
		
		`SWITCHOVER_STATUS`
		`--------------------`
		`TO PRIMARY`
		
		`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 C, Canceling a Logical Standby Switchover.
			
			You cannot do an "ALTER DATABASE PREPARE TO SWITCHOVER" for this switchover due to the mixed versions.

24		**10g Oracle RAC only**
		Start other nodes of the primary (original standby database)
		`SQL> ALTER DATABASE ENABLE THREAD 2;`
		`$ srvctl start instance –d boston –i boston2`	

Step	Primary	Standby	Notes

25	**10g Oracle RAC only**
	Start other nodes of the primary (original standby database)
	`SQL> ALTER DATABASE ENABLE THREAD 2;`
	`$ srvctl start instance –d chicago –i chicago2`		

V.	Install the New ORACLE_HOME and Apply the Patchset on the Original Primary		
26	Install a separate ORACLE_HOME and patch it to 11.1.0.7		This must be done here if a temporary log archive repository will be used.
27	**10g only** Follow MetaLink note 300479.1 to implement the workaround for issue 5236922		This will insure that there are no Enterprise Manager components installed on the new logical standby
28	Apply 11.1.0.7 patchset to CRS and ASM		
29	Install the 11.1.0.7 Patch set		For an out-of-place patchset apply the existing ORACLE_HOME is cloned using the Cloning procedure in Appendix E.
VI.	Create a Temporary Archive redo log repository (optional)		
30	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 can be met in the event of a primary site failure during this step.
VII.	Perform the Upgrade on the Former Primary		
31	Create a guaranteed restore point `SQL> CREATE RESTORE POINT PRE_LOGICAL_UPGRADE GUARANTEE FLASHBACK DATABASE;`		Note that is this 10.1.0.3 or a later 10.1 release then capture the current SCN as the flashback database point. `SQL> SELECT CURRENT_SCN FROM V$DATABASE;` Note: if this is a multi-standby configuration that includes one or more physical standby databases, please see instructions in this paper for "Special Instructions for Multi-Standby Configurations"
32	If not using DBUA then set cluster_database=false in preparation for upgrade `SQL> ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE;`		If using the Database Upgrade Assistant (dbua). dbua is executed from the new ORACLE_HOME. dbua requires CLUSTER_DATABASE+TRUE for a RAC database and it takes care of the OCR updates for you.

Step	Primary	Standby	Notes
33	Upgrade the database		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. **If you did not use DBUA then remember to set cluster_database=true before bouncing the database after the upgrade**.

34	If a temporary log archive repository was used then register logs so they don't have to be resent:
	`SQL> ALTER DATABASE REGISTER LOGFILE '….`
	
	Optionally, you can use the RMAN CATALOG command to catalog the archived redo log repository archivelog destination. e.g.:
	`RMAN> CATALOG START WITH`
	`'+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:
			::::::::::::::
			register_logs.sql
			::::::::::::::
			REM The parameter is the SCN from the 'recover standby database command
			set head off feedback off lines 133 pages 0 verify off echo off
			spool register_logs_for_standby.sql
			select 'alter database register logfile ' || chr(39) || name || chr(39) || ';'
			from v$archived_log
			where first_change# >= &1
			/
			spool off

35	Start Logical Standby Apply @ 11.1.0.7 `SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE NEW PRIMARY TO_BOSTON;`		If the 'new primary' clause is not used then it will receive an "ORA-16100: not a valid Logical Standby database" error.
36		Enable redo transfer `SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=ENABLE;`	
37	Verify that the logical standby is running correctly		See 10.3 Monitoring a Logical Standby Database [5]

Step	Primary	Standby	Notes
VIII.	Switch back to the original config (optional)		

38		Switch back to the original configuration where boston will be the logical standby and chicago the primary:
		
		Check the status
		`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`
		
		`SWITCHOVER_STATUS`
		`--------------------`
		`TO STANDBY`
		
		If the query returns "SESSIONS ACTIVE', then ensure it's ok to shut them down.
		`SELECT SID, PROCESS, PROGRAM FROM V$SESSION WHERE TYPE = 'USER' AND SID <> (SELECT DISTINCT SID FROM V$MYSTAT);`	

39	**10g Oracle RAC only**
	`$ srvctl stop instance –d chicago –i chicago2`
	`SQL> ALTER DATABASE DISABLE THREAD 2;`
		10g Oracle RAC only
		`$ srvctl stop instance –d boston –i boston2`
		`SQL> ALTER DATABASE DISABLE THREAD 2;`	

40		SQL> `ALTER DATABASE PREPARE TO SWITCHOVER TO LOGICAL STANDBY;`	If the PREPARE does not complete on the logical standby, you can cancel the switchover operation by issuing the following statements in order: 1. Cancel switchover on the primary database: SQL> ALTER DATABASE PREPARE TO SWITCHOVER CANCEL; 2. Cancel the switchover on the logical standby database SQL> ALTER DATABASE PREPARE TO SWITCHOVER CANCEL;
41	`SQL> ALTER DATABASE PREPARE TO SWITCHOVER TO PRIMARY;`		

42		Ensure the current primary is ready
		`SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE;`
		`SWITCHOVER_STATUS`
		`-----------------`
		`TO LOGICAL STANDBY`
		`1 row selected`	

Step	Primary	Standby	Notes
43	Ensure archive logs are received and applied during the primary switchover	SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO LOGICAL STANDBY;	"SWITCHOVER TO LOGICAL STANDBY;" cannot use the "WITH SESSION SHUTDOWN" clause so you need to manually shutdown sessions.
44	Defer Redo SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2=DEFER;		
45	Switchover to Primary on current logical standby SQL> SELECT SWITCHOVER_STATUS FROM V$DATABASE; SWITCHOVER_STATUS -------------------- TO PRIMARY SQL> ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY;		
46 .	**10g Oracle RAC only** Start other nodes of the primary (original standby database) SQL> alter database enable thread 2; $ srvctl start instance –d chicago –i chicago2		
47		**10g Oracle RAC only** Start other nodes of the standby SQL> ALTER DATABASE ENABLE THREAD 2; $ srvctl start instance –d boston –i boston2	
48		SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;	
IX.	Raise the COMPATIBLE Parameter Setting		
49	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 eliminates any ability to downgrade.
50		ALTER SYSTEM SET compatible='11.1.0.7' SCOPE=SPFILE;	
51	ALTER SYSTEM SET compatible='11.1.0.7' SCOPE=SPFILE;		
52		Bounce the standby database > srvctl stop database –d boston > srvctl start database –d boston	
53	Bounce the primary database > srvctl stop database –d chicago > 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`) patch set apply, it is recommended to clone the existing `ORACLE_HOME`. The following procedure shows the commands you would use on a Linux system:

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/11.1.0.6
/u01/app/oracle/product/11.1.0.7
```

2. Validate that any file changes between the source ORACLE_HOME and the copied ORACLE_HOME are ok (only log files may differ):

```
> diff -q -r /u01/app/oracle/product/11.1.0.6
/u01/app/oracle/product/11.1.0.7
```

3. 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/11.1.0.7/clone/bin/clone.pl
ORACLE_HOME=/u01/app/oracle/product/11.1.0.7
ORACLE_HOME_NAME=10gR2P2 '-
O"CLUSTER_NODES={chi01,chi02}"' '-
O"LOCAL_NODE=chi01"'
```

```
echo "Clone ended at `date`" | tee -a clone.log
```

4. As root user run root.sh on each node

5. Apply the patchset to the newly cloned ORACLE_HOME (/u01/app/oracle/product/11.1.0.7)

Note: If the source ORACLE_HOME contains any hard links or symbolic links these will also be cloned. If any of these links in the cloned ORACLE_HOME reference the source ORACLE_HOME then you must manually change them to point to the cloned ORACLE_HOME path.

References

1. Oracle Data Guard
   http://www.oracle.com/technology/deploy/availability/htdocs/DataGuardOverview.html
2. Oracle Maximum Availability Architecture
   http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm
3. Oracle Database 10g Release 2 Best Practices: Data Guard SQL Apply
   www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SQLApplyBestPractices.pdf
4. Oracle Database 10g Release 2 Best Practices: Data Guard Switchover and Failover
   http://www.oracle.com/technology/deploy/availability/pdf/MAA_WP_10gR2_SwitchoverFailoverBestPractices.pdf
5. Oracle Data Guard Concepts and Administration 11.1 (Part #B28294-02)
   http://download.oracle.com/docs/cd/B28359_01/server.111/b28294/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
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 Data Guard SQL Apply: Oracle Database 11g and 10gR2
July 2009
Authors: Ray Dutcher
Contributing Authors: Lawrence To, Joseph Meeks

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