Database Rolling Upgrades Made Easy by Using a Data Guard Physical Standby Database

Oracle Maximum Availability Architecture White Paper
October 2011

Maximum
Availability
Architecture

Oracle Best Practices For High Availability

Executive Overview 2
Rolling Database Upgrade Using Transient Logical Standby 3
Prerequisites and Limitations 4
Overview of Upgrades Made Easy (Who does What) 4
    Administrator: Prepare for Upgrade 4
    `physru` Script: Pre-Upgrade Tasks 5
    Administrator: Standby Database Upgrade 5
    `physru` Script: Post-Upgrade Tasks 5
    Administrator: Prepare Original Primary Database for Upgrade 5
    `physru`: Final Upgrade Tasks 5
Details of the physru Upgrade Script 6
    Pre-Upgrade Tasks 6
    Detailed Upgrade Steps 7
    Recovery from Errors 10
    Multiple Standby Database Considerations 10
    Fallback Options 13
    Flashback Considerations 13
Appendix A : Sample RAC physru Script Execution 14
Appendix B : Sample Single Instance physru Script Execution 20
References 24

Executive Overview

Oracle Maximum Availability Architecture (MAA) is the Oracle best practices blueprint for implementing Oracle high availability technologies. Starting with Oracle Database 11*g* release 1, the MAA recommended best practice for performing rolling database upgrades is to use the transient logical standby database feature available with Oracle Data Guard. This MAA best practice paper describes a Bourne shell script developed by Oracle to automate database rolling upgrades to new Oracle patch sets or full database releases beginning with Oracle Database 11*g* Release 1 and higher. The database rolling upgrade is performed using an existing Data Guard physical standby database and the transient logical standby rolling upgrade process. The Bourne shell script, named `physru`, is available for download by means of My Oracle Support Note 949322.1.

Oracle MAA best practices recommend using Online Patching for database one-off patches, Oracle Real Application Cluster (Oracle RAC) rolling upgrade for database patches, patchset updates (PSUs) and critical patch updates (CPUs), Cluster Ready Services (CRS) rolling upgrade, Oracle Automatic Storage Management (Oracle ASM) rolling upgrade and Data Guard Standby-First patch apply (refer to My Oracle Support note 1265700.1) whenever possible. Even in these cases, however, a Data Guard database rolling upgrade has the advantage of applying and testing the change on a completely separate system and database prior to switching the production application and clients over to the upgraded system. Refer to Oracle Database High Availability documentation for various rolling upgrade strategies for different scenarios.

The transient logical standby rolling upgrade process is attractive for several reasons:

- It greatly improves availability by eliminating planned downtime required for many of the usual tasks associated with a conventional database upgrade, including PL/SQL recompilation.
- It employs existing physical standby databases for database rolling upgrades; there is no additional storage or effort required to deploy a separate logical standby database for the sole purpose of a rolling upgrade.
- It requires executing only a single catalog upgrade to migrate both primary and standby databases to a new Oracle release.
- It allows for additional validation of the upgrade and the system before switching applications and clients to the new environment.
- When the upgrade process is complete, the primary database and physical standby database are both running the new Oracle release.

The `physru` upgrade script greatly reduces the complexity of executing a rolling database upgrade by automating most of the upgrade steps, leaving only the following steps for the user to perform:

- Calling DBUA or `catupgrd.sql` to upgrade the standby database
- Starting the upgraded standby database in the new Oracle home
- Starting the former primary database in the new Oracle home

Rolling Database Upgrade Using Transient Logical Standby

Using transient logical standby is the best practice to execute database rolling upgrades with near-zero downtime when upgrading to new patch sets or major releases of the Oracle Database. Such an upgrade is performed using the transient logical rolling upgrade process introduced in Oracle Database 11*g* Release 1 (11.1). Although the upgrade begins with a physical standby database, the transient logical standby process uses SQL Apply to take redo generated by a database running a lower Oracle release, and apply the redo to a standby database running a higher Oracle release. When the upgrade process is complete, both the primary database and its physical standby database are operating at the new Oracle Database release.

Note: Although the ability to perform a database rolling upgrade with a physical standby database was first introduced in Oracle Database 11*g* Release 1 (11.1), the basic elements of the upgrade process were first introduced for Data Guard logical standby databases beginning with Oracle Database 10*g* Release 1 (10.1). Thus, the new capability is based upon components that have been maturing over several major Oracle Database releases.

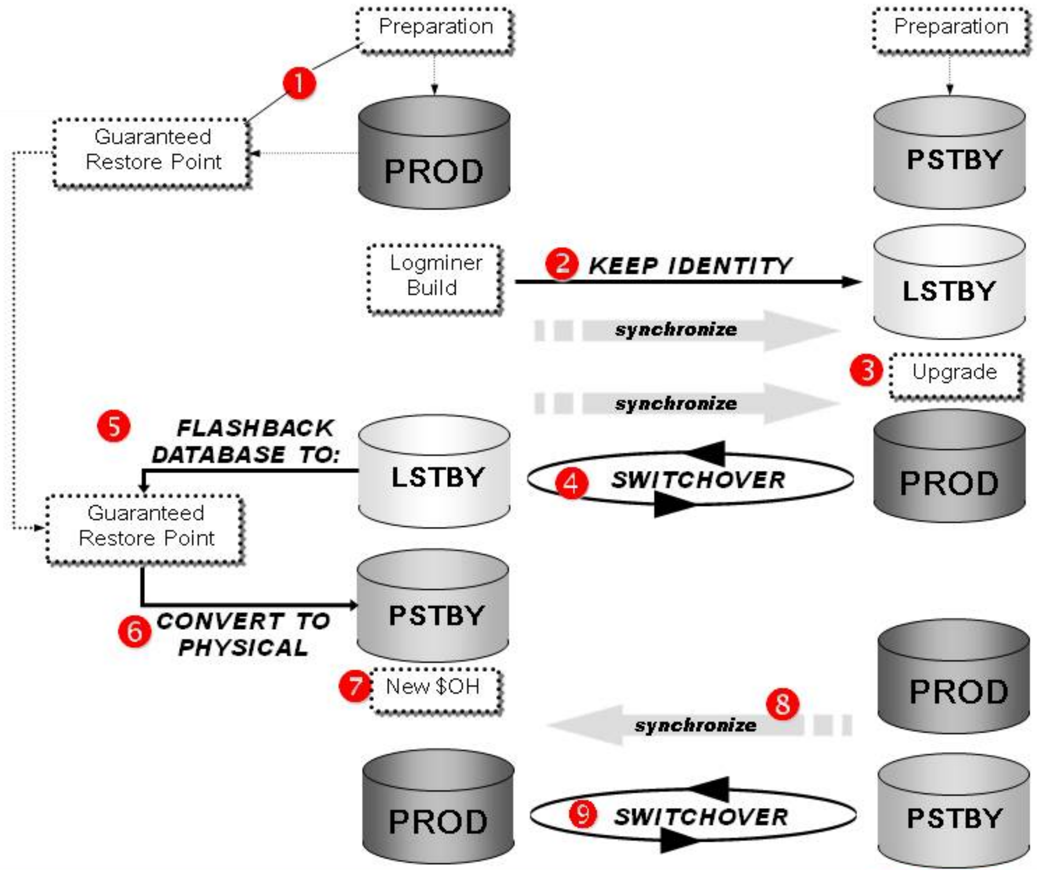

Figure 1: Transient Logical Rolling Upgrade Process

Figure 1 depicts the process flow for executing a transient logical standby rolling upgrade. While the process is described in step-by-step detail in the MAA best practices paper, "Database Rolling Upgrades using Transient Logical Standby" [6], the good news is that Oracle now provides the `physru` upgrade script that automates most of the steps required to perform the upgrade process. The `physru` script directly handles all steps of the transient logical rolling upgrade process except Step 3, "Upgrade Transient Logical Standby" and Step 7, "Open Physical Standby with new `$ORACLE_HOME`". The upgrade script is available for download through My Oracle Support Note 949322.1.

Note: The transient logical standby rolling upgrade process should be considered a planned maintenance operation and therefore performed during non-peak hours. Application downtime is incurred starting with the second execution of `physru` (Step 4 in Figure 1), normal operations can be resumed after the third and final execution of `physru` has completed (Step 9 in Figure 1).

Prerequisites and Limitations

While the `physru` script automates upgrade tasks, it is subject to the same prerequisites and limitations as the manual process for executing a transient logical database rolling upgrade. For a complete understanding of these requirements, see the 'General Restrictions' and 'Confirm Data Type Support' sections in the MAA best practices paper: Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11g [6].

In addition to the general prerequisites for a transient logical rolling database upgrade, the following additional prerequisites apply to the `physru` upgrade script:

- Ensure table rows in the primary database can be uniquely identified. See 'Prerequisite Conditions for Creating a Logical Standby Database' in the *Oracle Data Guard Concepts and Administration Guide* 11*g* Release 1 and 11*g* Release 2 documentation.
- Enable Flashback Database on the databases directly involved in the rolling upgrade.
- If you are using Data Guard broker to manage your configuration, then disable the broker for the duration of the upgrade process. This includes stopping the broker background process (by setting the initialization parameter `DG_BROKER_START=FALSE`).
- Ensure that the log transport (initialization parameter `LOG_ARCHIVE_DEST_`*`n`*) is correctly configured to perform a switchover from the primary database to the target physical standby database and back.
- Define a static database listener entry for the databases directly involved the rolling upgrade process for the appropriate listener/listener.ora. This allows the script to connect to a database instance that has been shut down.
- If you have multiple standby databases, review the 'Multiple Standby Considerations' section prior to starting the rolling upgrade.

Overview of Upgrades Made Easy (Who does What)

This section describes the responsibilities of the administrator and a high-level summary of the tasks that the `physru` script will perform.

Administrator: Prepare for Upgrade

MAA best practices recommend testing your application on the target release and testing standard upgrade procedures using the *Oracle Database Upgrade Guide* and the relevant Upgrade Companion notes (My Oracle Support Notes 601807.1 and 785351.1)

If you are planning to apply an Oracle patch or patch set upgrade, the best practice is to perform an out of place upgrade by cloning a new `ORACLE_HOME` and applying the patch/patch set to the cloned home on both the primary and standby systems. Having a separate `ORACLE_HOME` allows for easy fallback if necessary. Details for this method can be found in

'Appendix A – Cloning the Oracle Home to Apply a Patch Set' of the Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11*g* white paper.

It is also recommended that you review any role change triggers you may have already implemented in your environment. The `physru` script performs two role transitions (switchovers) that will cause the execution of your role change triggers. If you do not want these triggers to fire, you will need to disable them before beginning this process

When you have completed the necessary preparation, you are ready to begin executing the `physru` upgrade script.

`physru` Script: Pre-Upgrade Tasks

The `physru` script performs the following pre-upgrade tasks:

- Creates Guaranteed Restore Points (GRP) on both the primary database and the physical standby database that enable fallback to the beginning of the process or to intermediate steps along the way.
- Converts a physical standby into a transient logical standby database.

The script will exit after the physical standby database has been converted to a transient logical to allow you to perform the upgrade of the transient logical database.

Administrator: Standby Database Upgrade

The administrator performs the usual tasks for upgrading a database to a new Oracle Database release at the transient logical standby database.

`physru` Script: Post-Upgrade Tasks

After you complete the standby database upgrade, execute the `physru` script again (using the same command line parameters). The script automatically resumes from the point it left off, and during the second execution, the script performs the following tasks:

- Performs a switchover to the upgraded transient logical standby; the standby database becomes the primary. The script will not attempt the switchover until:
  - SQL Apply on the transient logical standby has become current with the primary database
  - Redo Apply on the standby database lags the primary database by 30 seconds or less
- Performs a flashback on the original primary database and converts the original primary into a physical standby of the new primary and shuts down the physical standby database

After the physical standby has been shut down, the script will exit to allow you to mount the physical standby using the new Oracle Home.

Administrator: Prepare Original Primary Database for Upgrade

If separate Oracle Homes were used for the old and new versions, the administrator mounts the new physical standby in the upgraded Oracle Home. If Oracle Homes are being patched in-place, then the administrator must upgrade the original primary system's Oracle Home and then mount the new physical standby database (original primary) using the patched binary.

`physru`: Final Upgrade Tasks

The final execution of the `physru` script (again, using the same command line parameters) will perform the following:

- Starts Redo Apply on the new physical standby database (the original primary database) to apply all redo that has been generated during this process, including any SQL statements that have been executed on the transient logical standby as part of the upgrade.
- Waits until the physical standby database has been synchronized with the primary database, providing a periodic status of the progress.
- When synchronized, the script offers the option of performing a final switchover to return the databases to their original roles, now on the updated software. This script will not attempt the switchover until:
  - Redo Apply starts successfully and applies all redo through the upgrade process
  - Redo Apply on the standby database lags the primary database by 30 seconds or less
- Removes all Guaranteed Restore Points created by `physru`.

Note: If you decide to postpone the final switchover to a later time, the switchover must be performed manually (for example, without the aid of `physru)`. So regardless of your selection, all Guaranteed Restore Points created by `physru` are removed as part of this final execution.

Details of the physru Upgrade Script

The `physru` script requires six parameters:

`$./physru <`*`sysdba user`*`> <`*`primary TNS alias`*`> <`*`physical standby TNS alias`*`> <`*`primary db unique name`*`> <`*`physical standby db unique name`*`> <`*`target version`*`>`

For example: `$./physru sys racprm1 racsby1_upgr racprm racsby 11.2.0.1.0`

The script prompts for the `SYSDBA` password at the beginning of each execution of the script. You can execute the script from any Unix/Linux node as long as that node has access to SQL*Plus and SQL*Net connectivity to the databases involved in the rolling upgrade.

Pre-Upgrade Tasks

Before executing the script, ensure that static `LISTENER.ORA` entries have been created for both the primary and physical standby databases involved in the rolling upgrade. These need only be created for one instance of each of the Oracle RAC databases, and you should ensure that whatever node from which you execute `physru` has a valid TNS alias created to connect to this instance. The following example shows the primary database's `LISTENER.ORA` file for instance `racprm1` of the Oracle RAC database `racprm`:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC =
        (SID_NAME = racprm1)
        (ORACLE_HOME = /u01/app/oracle/product/11.1.0/11.1.0.7/db)
     )
  )
```

The following example shows the physical standby database's `LISTENER.ORA` file for instance `racsby1` of the Oracle RAC database `racsby`:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC =
        (SID_NAME = racsby1)
        (ORACLE_HOME = /u01/app/oracle/product/11.1.0/11.1.0.7/db)
     )
  )
```

These static entries allow the script to connect to and restart the database instances after they have been shut down. After changing the `LISTENER.ORA` file, reload the listener process:

```
$lsnrctl reload <listener_name>
```

For each Oracle RAC database, ensure that a TNS alias has been defined to allow access directly to a single instance of the Oracle RAC database. These aliases should be used in the appropriate parameter when executing `physru`. If the physical standby database is an Oracle RAC database, this TNS alias must point to the instance running media recovery as `physru` starts and stops media recovery multiple times during execution. Stopping Redo Apply can only be performed on the instance where media recovery is running. For example:

```
TRUSBY1_UPGR =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <host VIP>)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = racsby1)
    )
  )
```

Detailed Upgrade Steps

1. Perform the initial execution of the `physru` script. The script will:

   a. Validate the environment before proceeding with the remainder of the script.

   b. Create control file backups for both the primary and the target physical standby database.

   c. Create a Guaranteed Restore Point on both the primary database and the physical standby database. This serves two purposes:

      i. Provides a fallback option to back out all changes that occurred as part of the rolling upgrade

      ii. Provides the script with the necessary starting point for converting the original primary database into a physical standby database

   d. Convert the physical standby database into a transient logical standby.

   The conversion of a physical standby database into a logical standby database requires that the physical standby be a single-instance database. Thus, Oracle RAC must be disabled. If the physical standby database is an Oracle RAC database, you will be directed to manually:

      i. Shut down all but one instance of the physical standby database.

      ii. Set initialization parameter `CLUSTER_DATABASE=FALSE` in the spfile of the physical standby database.

      iii. Shut down and mount the remaining instance.

   After the conversion to a transient logical standby has completed, you are directed to manually:

      i. Set initialization parameter `CLUSTER_DATABASE=TRUE` in the spfile of the transient logical standby database.

      ii. Shut down and startup this instance.

ii. Startup all remaining instances of the transient logical standby database.

The script processing ends and returns to the command prompt.

2. Upgrade the transient logical standby. You can perform the upgrade procedure using upgrade scripts or by using the Database Upgrade Assistant (DBUA). Review the *Oracle Database Upgrade Guide* for the release and/or the Database Readme before any upgrade. Do not change the `COMPATIBLE` initialization parameter as part of the upgrade.

   Note: While the transient logical standby is running in `OPEN MIGRATE` mode, it rejects all redo logs the primary is attempting to transmit. These rejected logs will be resolved on the next start of SQL Apply.

   After the upgrade to the transient logical standby database has completed, you should start SQL Apply**.** To manually start SQL Apply, using SQL*Plus log in as `SYSDBA` to the instance of the transient logical standby database you have been using with the `physru` script and issue:

   ```
   SQL> ALTER DATABASE START LOGICAL STANDBY APPLY IMMEDIATE;
   ```

   Note: If you expect to run with the transient logical standby database in place for an extended time prior to the next execution of the `phyrsu` script, you should tune SQL Apply to ensure the transient logical standby database performance characteristics can satisfy your requirements. Use the guidelines in SQL Apply Best Practices - Data Guard 11g white paper for SQL Apply recommendations.

3. Modify the `LISTENER.ORA` entry for the upgraded transient logical standby database to point to the new `ORACLE_HOME` and reload the listener. For example:

   ```
   SID_LIST_LISTENER =
     (SID_LIST =
        (SID_DESC =
           (SID_NAME = racsby1)
           (ORACLE_HOME = /u01/app/oracle/product/11.2.0/11.2.0.1/db)
        )
     )
   ```

   ```
   $ lsnrctl reload listener
   ```

4. Perform the second execution of the `physru` script. This will:

   a. Ensure the transient logical standby database has been upgraded to the target version and is not started in `OPEN MIGRATE` mode.

      Note: Once the transient logical standby database is no longer in `OPEN MIGRATE` mode, Data Guard automatically initiates redo gap fetching. Depending on the amount of redo generated, this process could take some time. Monitor the `DBA_LOGSTBY_LOG` view to determine the progress of SQL Apply and redo gap fetching.

   b. Ensure the transient logical standby database is current with the primary. This includes applying all changes that occurred to the primary database while the transient logical standby was being upgraded.

   c. Perform a switchover to the upgraded transient logical standby database.

   d. Flashback the original primary database to the initial Guaranteed Restore Point that was created in Step 1.

      Note: If the original primary database is an Oracle RAC database, all but one instance of the original primary database must be shut down prior to performing the flashback operation. Execution of the `physru` script will pause and direct you to perform the shutdown at the appropriate time.

   e. Convert the original primary database into a physical standby database.

   f. Shut down the physical standby database.

   The script processing ends and returns to the command prompt.

5. On the original primary database system, reset your environment and mount the physical standby database using the `$ORACLE_HOME` of the new release. You should also modify the `LISTENER.ORA` static entry for this database to point to the new `$ORACLE_HOME` and reload the listener, for example:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC =
        (SID_NAME = racprm)
        (ORACLE_HOME = /u01/app/oracle/product/11.2.0/11.2.0.1/db)
     )
  )
```

```
$ lsnrctl reload listener
```

   You should also ensure that the new `$ORACLE_HOME` has been configured to allow seamless starting of the database instance. This includes, but is not limited to, creating appropriate initialization files (`init.ora`) and Oracle password files in the `$ORACLE_HOME/dbs` directory and providing access to a current `TNSNAMES.ORA` file. You should also update the `/etc/oratab` entry for the database instance being upgraded.

6. If your original primary database definition is stored in the Oracle Cluster Registry (OCR), update the (OCR) to store the new `ORACLE_HOME` for the database. You should remove the OCR definition and re-add with the new information. For example, to update the OCR for an Oracle RAC database:

```
$ srvctl remove database –d racprm
$ srvctl add database –d racprm –o
/u01/app/oracle/product/11.2.0/11.2.0.1/db –o open
$ srvctl add inst –d racprm –i racprm1 –n node1
$ srvctl add inst –d racprm –i racprm2 –n node2
```

   Note: Removing the database from the OCR drops other dependent items such as services defined for the database, you must recreate these items after the database has been redefined in the OCR.

7. On the current upgraded primary database, enable the `LOG_ARCHIVE_DEST_STATE_n` from the upgraded primary database to the physical standby database to ensure prompt transmission of redo. For example:

```
SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
```

8. Perform the final execution of the `physru` script. This will:

   a. Ensure the physical standby database has been mounted with the correct software version.

   b. Start Redo Apply to bring the physical standby database current with the upgraded primary database. This recovery session re-applies all application activity that has occurred to the original primary database while the transient logical database was being upgraded, any changes directly associated with the upgrade and all transactions that occurred after the initial switchover to the transient logical standby.

   c. Optionally perform a switchover after the physical standby has become current with the primary database. If you choose to perform the switchover and the current primary database is an Oracle RAC database, you must manually shutdown all but one instance of the current primary. `physru` execution will pause and direct you to perform the shutdown at the appropriate time. After the switchover completes, the databases have been returned to their original roles.

d. Cleanup from the execution of the script. This entails removing the various GRPs created and producing statistics.

Recovery from Errors

The `physru` script is engineered to be able to restart from a point-of-failure. Should an error occur while executing the script, you have the option of resolving the issue and restarting `physru` to continue. The script determines the last completed stage and resumes execution from that point.

Note: If the last completed stage was not a planned exit point (for example, full completion of an execution phase), upon restart the script displays a menu offering the option to continue from the last execution point or start over from the beginning. If you choose to start over, the script assumes that you have manually performed all necessary cleanup steps and reconfigured the environment back to the original primary and physical standby database configurations.

Multiple Standby Database Considerations

It is possible to perform the transient logical standby rolling upgrade process with bystander physical standby databases, bystander logical standby databases and/or Oracle Streams databases in place. Bystanders are any additional standby databases that are in the same Data Guard configuration as the physical standby database used by the `physru` upgrade script to execute the upgrade. This section describes the steps to handle each type of bystander database.

Bystander Physical Standby Database Considerations

The `physru` script does not directly manage bystander physical standby databases during the upgrade process. However, they can be manually upgraded by executing the following steps.

1. Prior to starting the script for the first time (Step 1), create a Guaranteed Restore Point on each bystander physical standby database that you wish to retain in the Data Guard configuration after the upgrade has completed. Perform the following steps to create the GRP:

   a. Stop Redo Apply on the bystander physical standby database:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

   b. As the `SYSDBA` user, create the GRP:

   ```
   SQL> CREATE RESTORE POINT PRE_UPGRADE GUARANTEE FLASHBACK DATABASE;
   ```

   c. Start Redo Apply:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT
   LOGFILE DISCONNECT;
   ```

2. Clone the Oracle Home at each bystander standby and upgrade the cloned Oracle Home to the new version.

3. After the final execution of the `physru` script (Step 8):

   a. Stop Redo Apply on the bystander physical standby database:

   ```
   SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;
   ```

   b. If the bystander database is an Oracle RAC database, shut down all but one database instance:

```
$ srvctl stop instance –d <bystander database name> –i <bystander
instance> –o abort
```

c. Flash back the bystander physical standby database to the `PRE_UPGRADE` GRP:

```
SQL> FLASHBACK DATABASE TO RESTORE POINT PRE_UPGRADE;
```

d. Shut down the bystander standby database:

e. Mount the bystander physical standby database using the upgraded Oracle Home.

f. Start Redo Apply on the bystander physical standby database:

```
SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT
LOGFILE DISCONNECT;
```

g. If the bystander physical standby database is an Oracle RAC database, start the remaining database instances:

```
$ srvctl start instance –d <bystander database name> –i <bystander
instance> –o <startup option>
```

h. Drop the Guaranteed Restore Point:

```
DROP RESTORE POINT PRE_UPGRADE;
```

Bystander Logical Standby Database Considerations

The `physru` script does not directly manage bystander logical standby databases during the upgrade process. Use the following process to maintain bystander logical standby databases as part of the transient logical standby rolling upgrade:

1. Before beginning the transient logical rolling upgrade process:

   a. All bystander logical standby databases must be manually upgraded to the target Oracle RDBMS version.

   b. On each bystander logical standby database create a database link that points to the physical standby database that will take on the role of transient logical standby during the upgrade process. The `SYSTEM` userid should be explicitly specified in the `CONNECT TO` clause.

   ```
   SQL> CREATE DATABASE LINK <link name> CONNECT TO SYSTEM IDENTIFIED
   BY <password> USING <tns alias of transient logical standby>;
   ```

   c. On the physical standby database that will take on the role of transient logical standby during the upgrade process, ensure there is a `LOG_ARCHIVE_DEST_n` parameter setting for each bystander logical standby database. These should be defined with `VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)` and have `LOG_ARCHIVE_DEST_STATE_n=ENABLE`.

   ```
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_n=’SERVICE=<bystander logical
   service> LGWR ASYNC VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
   DB_UNIQUE_NAME=<bystander logical db unique name>;
   ```

   ```
   SQL> ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_n=ENABLE;
   ```

2. During the transient logical upgrade process:

   a. SQL Apply should continue running on the bystander logical standby databases.

   b. Ensure the bystander logical standby database does not have a large lag behind the primary database. You can use a query similar to the following to determine SQL Apply progress. See ‘Managing a Logical

Standby Database' in the *Oracle Data Guard Concepts and Administration Guide* 11*g* Release 1 and 11*g* Release 2 documentation for more information on monitoring SQL Apply.

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
SQL> SELECT APPLIED_TIME, APPLIED_SCN, MINING_TIME, MINING_SCN -
> FROM V$LOGSTDBY_PROGRESS;

APPLIED_TIME            APPLIED_SCN   MINING_TIME           MINING_SCN
--------------------    -----------   --------------------  -----------
29-JUL-2011 12:00:05    346791023     29-JUL-2011 12:10:05  3468810134
```

c. After the second execution of the script (Step 4f of the Detailed Upgrade Steps section), the transient logical standby database has assumed the primary role. At this time, on each bystander logical standby database:

   i. Check that the bystander logical standby has received and processed the End-of-Redo (EOR) marker from the switchover. The following query should return `SWITCHOVER` in the `VALUE` column:

```
SQL> SELECT NAME, VALUE FROM SYSTEM.LOGSTDBY$PARAMETERS WHERE
NAME='COMPLETED SESSION';

NAME                      VALUE
------------------        -------------------------------------
COMPLETED_SESSION         SWITCHOVER
```

   Note: When the switchover operation from the primary database to the transient logical standby database is encountered on the bystander logical standby database via the EOR marker in the redo stream, the SQL Apply process on the bystander logical standby will stop.

   ii. The bystander logical standby database must now be informed that it will be applying redo from a new primary database. Issue the following command on the bystander logical standby:

```
SQL> ALTER DATABASE START LOGICAL STANDBY APPLY NEW PRIMARY
<db link name created in step 1b of this section>;
```

   Note: Starting SQL Apply with the `NEW PRIMARY` clause will not start SQL Apply in Real Time Apply (RTA) mode. This version of the command will accept the `IMMEDIATE` clause but will ignore it. To enable RTA, after starting SQL Apply with the `NEW PRIMARY` clause, stop SQL Apply and restart it with the `IMMEDIATE` clause. Do not specify the `NEW PRIMARY` clause on this or any subsequent start of SQL Apply as part of the transient logical standby rolling upgrade process. When you switch back to your original configuration, the bystander logical standby databases will follow to the new primary database with no additional action.

d. When all of the bystander logical standby databases have been changed to recognize redo from the new primary database, continue on with the rest of the transient logical rolling upgrade process (Step 5 of the Detailed Upgrade Steps section).

e. After completing the transient logical upgrade process, on each bystander logical standby database optionally drop the database link created in Step 1b of this section. The database link can be retained to use in future transient logical rolling upgrade operations if desired.

```
SQL> DROP DATABASE LINK <link name>;
```

Oracle Streams Database Considerations

The `physru` script does not directly manage Oracle Streams databases during the upgrade process. Use the following process to maintain Oracle Streams databases as part of the transient logical upgrade process.

1. Prior to starting the rolling upgrade process stop the Streams capture on the primary database. The capture process will remain stopped for the duration of the rolling upgrade.
2. During the second execution of the `physru` script, a switchover is performed to transition the transient logical standby database to the primary role. When this second execution of the script completes, start the Streams capture process on the new primary database. Capture will begin mining redo logs that were generated while the capture process was stopped.

Fallback Options

The fallback options when using the `physru` script are the same as those documented in the 'Fallback Best Practices' section of the Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11g [6] white paper.

Flashback Considerations

Although the `physru` script creates multiple Guaranteed Restores Points during execution, these GRPs are mainly used by the script to maintain progress points. The only GRP created by `physru` that you should manually use is the GRP named `PRU_0000_0001`. This GRP is created prior to any work being done and can be used safely for flashback to back out all changes. Attempts to use any other GRP created by `physru` as a flashback point can lead to undesirable results.

Note: If you need to abandon the upgrade, after performing your selected fallback option, ensure all Guaranteed Restore Points generated by the script have been dropped.

To determine which GRPs have been created:

```
SQL>SELECT NAME FROM V$RESTORE_POINT;
```

To drop the restore point(s);

```
SQL> DROP RESTORE POINT <GRP_NAME>;
```

Appendix A : Sample RAC physru Script Execution

In this example, the Data Guard configuration consists of two Oracle RAC databases , the primary database, `racprm`, and the target physical standby database `racsby`. The databases are initially running Oracle Database 11*g* release 1 (11.1.0.7) and will be upgraded to release 11.2.0.1. The configuration is managed via SQL*Plus, and the Data Guard broker is not in use.

1. The initial execution of `physru` validates the environment for the rolling upgrade and prepares `racsby` for rolling upgrade by converting it to a transient logical standby database. The aliases being used connect to instance 1 (e.g. `racprm1` and `racsby1`) of each database. Note the following items :
   - You are prompted for the `SYSDBA` password when the script begins execution. The password is not maintained or saved by the script, so every execution will include this prompt.
   - In Stage 2, `physru` has determined that `racsby` is an Oracle RAC database. The conversion of a physical standby database into a logical standby database requires that the Oracle RAC option be disabled in the physical standby database (`CLUSTER_DATABASE=FALSE`). Prior to continuing the script execution, you must :
     1. Manually stop all secondary instances
     2. Issue the `ALTER SYSTEM SET CLUSTER_DATABASE=FALSE SCOPE=SPFILE` command in the remaining instance.
     3. Restart that instance for the parameter change to take effect.
   - After completion of Stage 2 (the end of the initial execution of `physru`), you should re-enable the Oracle RAC option in the standby database before performing the upgrade.

```
$ ./physru sys racprm1 racsby1_upgr racprm racsby 11.2.0.1.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Jun 02 08:37:54 2010 [0-1] Identifying rdbms software version
Jun 02 08:37:55 2010 [0-1] database racprm is at version 11.1.0.7.0
Jun 02 08:37:55 2010 [0-1] database racsby is at version 11.1.0.7.0
Jun 02 08:37:56 2010 [0-1] verifying flashback database is enabled at racprm and racsby
Jun 02 08:37:56 2010 [0-1] verifying available flashback restore points
Jun 02 08:37:56 2010 [0-1] verifying DG Broker is disabled
Jun 02 08:37:57 2010 [0-1] looking up prior execution history
Jun 02 08:37:57 2010 [0-1] purging script execution state from database racprm
Jun 02 08:37:57 2010 [0-1] purging script execution state from database racsby
Jun 02 08:37:57 2010 [0-1] starting new execution of script

### Stage 1: Backup user environment in case rolling upgrade is aborted
Jun 02 08:37:58 2010 [1-1] stopping media recovery on racsby
Jun 02 08:37:58 2010 [1-1] creating restore point PRU_0000_0001 on database pstby
Jun 02 08:37:58 2010 [1-1] backing up current control file on racsby
Jun 02 08:38:02 2010 [1-1] created backup control file
/u01/app/oracle/product/11.1.0/11.1.0.7/db/dbs/PRU_0001_racsby_f.f
Jun 02 08:38:02 2010 [1-1] creating restore point PRU_0000_0001 on database racprm
Jun 02 08:38:02 2010 [1-1] backing up current control file on racprm
Jun 02 08:38:04 2010 [1-1] created backup control file
/u01/app/oracle/product/11.1.0/11.1.0.7/db/dbs/PRU_0001_racprm_f.f

NOTE: Restore point PRU_0000_0001 and backup control file PRU_0001_racsby_f.f
      can be used to restore racsby back to its original state as a
      physical standby, in case the rolling upgrade operation needs to be aborted
      prior to the first switchover done in Stage 4.
```

```
### Stage 2: Create transient logical standby from existing physical standby
Jun 02 08:38:05 2010 [2-1] verifying RAC is disabled at racsby

WARN: racsby is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance, disable the RAC, and
      restart instance racsby1 in mounted mode.  This can be accomplished
      with the following steps:

        1) Shutdown all instances other than instance racsby1.
           eg: srvctl stop instance -d racsby -i racsby2 -o abort

        2) On instance racsby1, set the cluster_database parameter to FALSE.
           eg: SQL> alter system set cluster_database=false scope=spfile;

        3) Shutdown instance racsby1.
           eg: SQL> shutdown abort;

        4) Startup instance racsby1 in mounted mode.
           eg: SQL> startup mount;

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n): y

Jun 02 08:39:16 2010 [2-1] continuing
Jun 02 08:39:16 2010 [2-1] verifying RAC is disabled at racsby
Jun 02 08:39:17 2010 [2-1] verifying database roles
Jun 02 08:39:17 2010 [2-1] verifying physical standby is mounted
Jun 02 08:39:17 2010 [2-1] verifying database protection mode
Jun 02 08:39:17 2010 [2-1] verifying transient logical standby datatype support
Jun 02 08:39:20 2010 [2-2] starting media recovery on racsby
Jun 02 08:39:26 2010 [2-2] confirming media recovery is running
Jun 02 08:39:27 2010 [2-2] waiting for v$dataguard_stats view to initialize
Jun 02 08:39:28 2010 [2-2] waiting for apply lag on racsby to fall below 30 seconds
Jun 02 08:40:00 2010 [2-2] apply lag is now less than 30 seconds
Jun 02 08:40:00 2010 [2-2] stopping media recovery on racsby
Jun 02 08:40:01 2010 [2-2] executing dbms_logstdby.build on database racprm
Jun 02 08:40:26 2010 [2-2] converting physical standby into transient logical standby
Jun 02 08:40:33 2010 [2-3] shutting down database racsby
Jun 02 08:40:44 2010 [2-3] mounting database racsby
Jun 02 08:40:55 2010 [2-3] opening database racsby
Jun 02 08:41:02 2010 [2-4] configuring transient logical standby parameters for rolling upgrade
Jun 02 08:41:02 2010 [2-4] starting logical standby on database racsby
Jun 02 08:41:09 2010 [2-4] waiting until logminer dictionary has fully loaded
Jun 02 08:41:20 2010 [2-4] dictionary load 39% complete
Jun 02 08:41:30 2010 [2-4] dictionary load 75% complete
Jun 02 08:41:40 2010 [2-4] dictionary load 99% complete
Jun 02 08:41:50 2010 [2-4] dictionary load is complete
Jun 02 08:41:53 2010 [2-4] waiting for v$dataguard_stats view to initialize
Jun 02 08:42:12 2010 [2-4] waiting for apply lag on racsby to fall below 30 seconds
Jun 02 08:42:13 2010 [2-4] apply lag is now less than 30 seconds

NOTE: Database racsby is now ready to be upgraded.  This script has left the
      database open in case you want to perform any further tasks before
      upgrading the database.  Once the upgrade is complete, the database must
      opened in READ WRITE mode before this script can be called to resume the
      rolling upgrade.

NOTE: Database racsby may be reverted back to a RAC database upon completion
      of the rdbms upgrade.  This can be accomplished by performing the
      following steps:

          1) On instance racsby1, set the cluster_database parameter to TRUE.
          eg: SQL> alter system set cluster_database=true scope=spfile;
```

```
2) Shutdown instance racsby1.
eg: SQL> shutdown abort;

3) Startup and open all instances for database racsby.
eg: srvctl start database -d racsby
```

2. At this point, `racsby` has been upgraded from release 11.1.0.7 to release 11.2.0.1 using the Database Upgrade Assistant (DBUA).

   Note: When using DBUA to perform your upgrade, be aware there is a prompt to disable both Flashback Database and archive log mode. Do not disable these features.

   After completing the upgrade, if the location of the `ORACLE_HOME` has changed, you must reset the static `LISTENER.ORA` entry for the transient logical standby database. One of the parameters in the static entry is `ORACLE_HOME`, which must match the new location. After modifying the entry, reload the listener.

   When the upgrade is complete, the script is executed for the second time. This execution:

   - Validates that the transient logical standby database (`racsby`) has been upgraded to the target release
   - Ensures `racsby` database is current
   - Performs a switchover operation, making `racsby` the primary database and the original primary (`racprm`) a logical standby
   - Prompts you to shut down all but one instance of the original primary if it is an Oracle RAC database
   - Performs a flashback operation on `racprm` to the initial GRP created by the script
   - Converts the `prim` into a physical standby database
   - Shuts down `prim` in preparation for running on the upgraded binary

   Note: During this execution of the script, the original primary database (`racprm`) will be started in a new `ORACLE_HOME` if you are performing an out-of-place upgrade. The best practices is:

     o To set up the new `ORACLE_HOME` on all nodes to allow seamless starting of the original primary by ensuring the initialization parameter file (`init.ora`) and password files are in place in the `$ORACLE_`HOME`/dbs` directory

     o To update the `/etc/oratab` entry for the database to point to the new `ORACLE_HOME`

     o To ensure the `TNSNAMES.ORA` file is accessible to all instances of the database

```
$ ./physru sys racprm1 racsby1_upgr racprm racsby 11.2.0.1.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Jun 02 09:22:22 2010 [0-1] Identifying rdbms software version
Jun 02 09:22:22 2010 [0-1] database racprm is at version 11.1.0.7.0
Jun 02 09:22:22 2010 [0-1] database racsby is at version 11.2.0.1.0
Jun 02 09:22:23 2010 [0-1] verifying flashback database is enabled at racprm and racsby
Jun 02 09:22:24 2010 [0-1] verifying available flashback restore points
Jun 02 09:22:24 2010 [0-1] verifying DG Broker is disabled
Jun 02 09:22:24 2010 [0-1] looking up prior execution history
Jun 02 09:22:24 2010 [0-1] last completed stage [2-4] using script version 0001
Jun 02 09:22:24 2010 [0-1] resuming execution of script

### Stage 3: Validate upgraded transient logical standby
Jun 02 09:22:25 2010 [3-1] database racsby is no longer in OPEN MIGRATE mode
Jun 02 09:22:25 2010 [3-1] database racsby is at version 11.2.0.1.0
```

```
### Stage 4: Switch the transient logical standby to be the new primary
Jun 02 09:22:26 2010 [4-1] waiting for racsby to catch up (this could take a while)
Jun 02 09:22:27 2010 [4-1] starting logical standby on database racsby
Jun 02 09:22:33 2010 [4-1] waiting for v$dataguard_stats view to initialize
Jun 02 09:22:34 2010 [4-1] waiting for apply lag on racsby to fall below 30 seconds
Jun 02 09:23:05 2010 [4-1] current apply lag: 266
Jun 02 09:24:06 2010 [4-1] current apply lag: 326
Jun 02 09:24:36 2010 [4-1] apply lag is now less than 30 seconds
Jun 02 09:24:37 2010 [4-2] switching racprm to become a logical standby
Jun 02 09:25:13 2010 [4-2] racprm is now a logical standby
Jun 02 09:25:13 2010 [4-3] waiting for standby racsby to process end-of-redo from primary
Jun 02 09:25:14 2010 [4-4] switching racsby to become the new primary
Jun 02 09:25:27 2010 [4-4] racsby is now the new primary

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to physical
Jun 02 09:25:28 2010 [5-1] verifying instance racprm1 is the only active instance

WARN: racprm is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:

        1) Shutdown all instances other than instance racprm1.
           eg: srvctl stop instance -d racprm -i racprm2 -o abort

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n): y

Jun 02 09:25:56 2010 [5-1] continuing
Jun 02 09:25:56 2010 [5-1] verifying instance racprm1 is the only active instance
Jun 02 09:29:45 2010 [5-1] shutting down database racprm
Jun 02 09:29:55 2010 [5-1] mounting database racprm
Jun 02 09:30:17 2010 [5-2] flashing back database racprm to restore point PRU_0000_0001
Jun 02 09:30:26 2010 [5-3] converting racprm into physical standby
Jun 02 09:30:30 2010 [5-4] shutting down database racprm

NOTE: Database racprm has been shutdown, and is now ready to be started
      using the newer version Oracle binary.  This script requires the
      database to be mounted (on all active instances, if RAC) before calling
      this script to resume the rolling upgrade.

NOTE: Database racprm is no longer limited to single instance operation since
      the database has been successfully converted into a physical standby.
      For increased availability, Oracle recommends starting all instances in
      the RAC on the newer binary by performing the following step:

        1) Startup and mount all instances for database racprm
        eg: srvctl start database -d racprm -o mount
```

3. `racprm` is now started in mount mode using the release 11.2.0.1 binary. The script is executed one last time to :

   - Start Redo Apply on `racprm.`
   - Verify that `racprm` becomes current with `racsby`.
   - Prompt to perform a switchover, returning `racprm` to the primary role and `racsby` to the physical standby role. If the current primary is an Oracle RAC database, you will be prompted to shut down all but one instance of the current primary database prior to the switchover.
   - Clean up from the script execution by dropping all GRPs that were created as a part of the process.

> **Note :** While `racprim` is shut down, the `LOG_ARCHIVE_DEST_STATE_`*`n`* on `racsby` pointing to `racprm` is marked invalid. Upon mounting `racprm`, the invalid state of the destination should automatically clear quickly, but to reduce the delay, manually enable the destination on `racsby` (`ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_`*`n`*`=ENABLE`).

```
$ ./physru sys racprm1 racsby1_upgr racprm racsby 11.2.0.1.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Jun 02 09:33:29 2010 [0-1] Identifying rdbms software version
Jun 02 09:33:29 2010 [0-1] database racprm is at version 11.2.0.1.0
Jun 02 09:33:30 2010 [0-1] database racsby is at version 11.2.0.1.0
Jun 02 09:33:30 2010 [0-1] verifying flashback database is enabled at racprm and racsby
Jun 02 09:33:31 2010 [0-1] verifying available flashback restore points
Jun 02 09:33:31 2010 [0-1] verifying DG Broker is disabled
Jun 02 09:33:31 2010 [0-1] looking up prior execution history
Jun 02 09:33:32 2010 [0-1] last completed stage [5-4] using script version 0001
Jun 02 09:33:32 2010 [0-1] resuming execution of script

### Stage 6: Run media recovery through upgrade redo
Jun 02 09:33:33 2010 [6-1] upgrade redo region identified as scn range [1682893769, 1683191330]
Jun 02 09:33:36 2010 [6-1] starting media recovery on racprm
Jun 02 09:35:37 2010 [6-1] confirming media recovery is running
Jun 02 09:37:20 2010 [6-1] waiting for media recovery to initialize v$recovery_progress
Jun 02 09:38:22 2010 [6-1] monitoring media recovery's progress
Jun 02 09:38:23 2010 [6-1] failed to determine the last applied scn by media recovery.
Jun 02 09:38:23 2010 [6-1] confirming media recovery is running
Jun 02 09:38:39 2010 [6-3] recovery of upgrade redo at 06% - estimated complete at Jun 02
09:43:04
Jun 02 09:38:54 2010 [6-3] recovery of upgrade redo at 07% - estimated complete at Jun 02
09:46:13
Jun 02 09:39:10 2010 [6-3] recovery of upgrade redo at 32% - estimated complete at Jun 02
09:40:53
Jun 02 09:39:25 2010 [6-3] recovery of upgrade redo at 58% - estimated complete at Jun 02
09:40:10
Jun 02 09:39:40 2010 [6-3] recovery of upgrade redo at 78% - estimated complete at Jun 02
09:40:01
Jun 02 09:39:55 2010 [6-3] recovery of upgrade redo at 88% - estimated complete at Jun 02
09:40:07
Jun 02 09:40:10 2010 [6-3] recovery of upgrade redo at 96% - estimated complete at Jun 02
09:40:13
Jun 02 09:40:26 2010 [6-4] media recovery has finished recovering through upgrade

### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover
     which will restore racprm back to a primary database and
     racsby back to a physical standby database.  If you answer 'n'
     to the question below, racprm will remain a physical standby
     database and racsby will remain a primary database.

Do you want to perform a switchover? (y/n): y

Jun 02 09:41:53 2010 [7-2] continuing
Jun 02 09:41:53 2010 [7-2] verifying instance racsby1 is the only active instance

WARN: racprm is a RAC database.  Before this script can continue, you
      must manually reduce the RAC to a single instance.  This can be
      accomplished with the following step:

        1) Shutdown all instances other than instance racsby1.
           eg: srvctl stop instance -d racsby -i racsby2

      Once these steps have been performed, enter 'y' to continue the script.
      If desired, you may enter 'n' to exit the script to perform the required
      steps, and recall the script to resume from this point.

Are you ready to continue? (y/n): y

Jun 02 09:42:40 2010 [7-2] continuing
Jun 02 09:42:40 2010 [7-2] verifying instance racsby1 is the only active instance
Jun 02 09:42:44 2010 [7-2] waiting for v$dataguard_stats view to initialize
Jun 02 09:42:44 2010 [7-2] waiting for apply lag on racprm to fall below 30 seconds
Jun 02 09:42:45 2010 [7-2] apply lag is now less than 30 seconds
Jun 02 09:42:45 2010 [7-3] switching racsby to become a physical standby
Jun 02 09:43:09 2010 [7-3] racsby is now a physical standby
Jun 02 09:43:09 2010 [7-3] shutting down database racsby
Jun 02 09:43:17 2010 [7-3] mounting database racsby
Jun 02 09:43:34 2010 [7-4] waiting for standby racprm to process end-of-redo from primary
Jun 02 09:43:35 2010 [7-5] switching racprm to become the new primary
Jun 02 09:43:37 2010 [7-5] racprm is now the new primary
Jun 02 09:43:37 2010 [7-5] opening database racprm
Jun 02 09:43:44 2010 [7-6] starting media recovery on racsby
Jun 02 09:43:51 2010 [7-6] confirming media recovery is running

NOTE: Database racprm has completed the switchover to the primary role, but
      instance racprm1 is the only open instance.  For increased availability,
      Oracle recommends opening the remaining active instances which are
      currently in mounted mode by performing the following steps:

        1) Shutdown all instances other than instance racprm1.
        eg: srvctl stop instance -d racprm -i racprm2

        2) Startup and open all inactive instances for database racprm.
        eg: srvctl start database -d racprm

NOTE: Database racsby is no longer limited to single instance operation since
      it has completed the switchover to the physical standby role.  For
      increased availability, Oracle recommends starting the inactive
      instances in the RAC by performing the following step:

        1) Startup and mount inactive instances for database racsby
        eg: srvctl start database -d racsby -o mount

### Stage 8: Statistics
script start time:                                        02-Jun-10 08:37:58
script finish time:                                       02-Jun-10 09:44:46
total script execution time:                                    +00 01:06:48
wait time for user upgrade:                                     +00 00:40:12
active script execution time:                                   +00 00:26:36
transient logical creation start time:                    02-Jun-10 08:39:19
transient logical creation finish time:                   02-Jun-10 08:40:33
primary to logical switchover start time:                 02-Jun-10 09:24:37
logical to primary switchover finish time:                02-Jun-10 09:25:27
primary services offline for:                                   +00 00:00:50
total time former primary in physical role:                     +00 00:10:42
time to reach upgrade redo:
time to recover upgrade redo:                                   +00 00:01:48
primary to physical switchover start time:                02-Jun-10 09:41:21
physical to primary switchover finish time:               02-Jun-10 09:43:44
primary services offline for:                                   +00 00:02:23

SUCCESS: The physical rolling upgrade is complete
```

Appendix B : Sample Single Instance physru Script Execution

In this example, the Data Guard configuration consists of two single-instance databases: the primary database `prim` and the target physical standby database `pstby`. The databases are initially running on Oracle Database 11*g* release 1 (11.1.0.7.0) and are to be upgraded to release 11.2.0.1.0. The configuration is managed via SQL*Plus, and the Data Guard broker is not in use.

1. The initial execution of `physru` validates the environment for the rolling upgrade and prepares `pstby` for rolling upgrade by converting it to a transient logical standby database. For example :

```
$ ./physru sys prim pstby_upgr prim pstby 11.2.0.1.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Jun 07 07:04:35 2010 [0-1] Identifying rdbms software version
Jun 07 07:04:35 2010 [0-1] database prim is at version 11.1.0.7.0
Jun 07 07:04:36 2010 [0-1] database pstby is at version 11.1.0.7.0
Jun 07 07:04:36 2010 [0-1] verifying flashback database is enabled at prim and pstby
Jun 07 07:04:37 2010 [0-1] verifying available flashback restore points
Jun 07 07:04:37 2010 [0-1] verifying DG Broker is disabled
Jun 07 07:04:37 2010 [0-1] looking up prior execution history
Jun 07 07:04:37 2010 [0-1] purging script execution state from database prim
Jun 07 07:04:38 2010 [0-1] purging script execution state from database pstby
Jun 07 07:04:38 2010 [0-1] starting new execution of script

### Stage 1: Backup user environment in case rolling upgrade is aborted
Jun 07 07:04:38 2010 [1-1] stopping media recovery on pstby
Jun 07 07:04:39 2010 [1-1] creating restore point PRU_0000_0001 on database pstby
Jun 07 07:04:39 2010 [1-1] backing up current control file on pstby
Jun 07 07:04:40 2010 [1-1] created backup control file
/u01/app/oracle/product/11.1.0/11.1.0.7/db/dbs/PRU_0001_pstby_f.f
Jun 07 07:04:40 2010 [1-1] creating restore point PRU_0000_0001 on database prim
Jun 07 07:04:41 2010 [1-1] backing up current control file on prim
Jun 07 07:04:43 2010 [1-1] created backup control file
/u01/app/oracle/product/11.1.0/11.1.0.7/db/dbs/PRU_0001_prim_f.f

NOTE: Restore point PRU_0000_0001 and backup control file PRU_0001_pstby_f.f
      can be used to restore pstby back to its original state as a
      physical standby, in case the rolling upgrade operation needs to be aborted
      prior to the first switchover done in Stage 4.

### Stage 2: Create transient logical standby from existing physical standby
Jun 07 07:04:44 2010 [2-1] verifying RAC is disabled at pstby
Jun 07 07:04:44 2010 [2-1] verifying database roles
Jun 07 07:04:44 2010 [2-1] verifying physical standby is mounted
Jun 07 07:04:44 2010 [2-1] verifying database protection mode
Jun 07 07:04:44 2010 [2-1] verifying transient logical standby datatype support
Jun 07 07:04:46 2010 [2-2] starting media recovery on pstby
Jun 07 07:04:53 2010 [2-2] confirming media recovery is running
Jun 07 07:04:56 2010 [2-2] waiting for v$dataguard_stats view to initialize
Jun 07 07:04:56 2010 [2-2] waiting for apply lag on pstby to fall below 30 seconds
Jun 07 07:04:56 2010 [2-2] apply lag is now less than 30 seconds
Jun 07 07:04:57 2010 [2-2] stopping media recovery on pstby
Jun 07 07:04:57 2010 [2-2] executing dbms_logstdby.build on database prim
Jun 07 07:05:26 2010 [2-2] converting physical standby into transient logical standby
Jun 07 07:05:33 2010 [2-3] shutting down database pstby
Jun 07 07:05:44 2010 [2-3] mounting database pstby
Jun 07 07:05:56 2010 [2-3] opening database pstby
Jun 07 07:06:04 2010 [2-4] configuring transient logical standby parameters for rolling upgrade
Jun 07 07:06:04 2010 [2-4] starting logical standby on database pstby
Jun 07 07:06:11 2010 [2-4] waiting until logminer dictionary has fully loaded
Jun 07 07:06:21 2010 [2-4] dictionary load 70% complete
Jun 07 07:06:32 2010 [2-4] dictionary load 75% complete
Jun 07 07:06:42 2010 [2-4] dictionary load is complete
Jun 07 07:06:43 2010 [2-4] waiting for v$dataguard_stats view to initialize
Jun 07 07:07:12 2010 [2-4] waiting for apply lag on pstby to fall below 30 seconds
Jun 07 07:07:12 2010 [2-4] apply lag is now less than 30 seconds

NOTE: Database pstby is now ready to be upgraded.  This script has left the
```

```
      database open in case you want to perform any further tasks before
      upgrading the database.  Once the upgrade is complete, the database must
      opened in READ WRITE mode before this script can be called to resume the
      rolling upgrade.

NOTE: If pstby was previously a RAC database that was disabled, it may be
      reverted back to a RAC database upon completion of the rdbms upgrade.
      This can be accomplished by performing the following steps:

          1) On instance pstby, set the cluster_database parameter to TRUE.
          eg: SQL> alter system set cluster_database=true scope=spfile;

          2) Shutdown instance pstby.
          eg: SQL> shutdown abort;

          3) Startup and open all instances for database pstby.
          eg: srvctl start database -d pstby
```

2. At this point, `pstby` is upgraded from release 11.1.0.7 to release 11.2.0.1 using the Database Upgrade Assistant (DBUA).

   Note: When using DBUA to perform your upgrade, be aware there is a prompt to disable both Flashback Database and archive log mode. Do not disable these features.

   After completing the upgrade, if the location of `ORACLE_HOME` has changed, then you must reset the static `LISTENER.ORA` entry for the transient logical standby database. One of the parameters in the static entry is `ORACLE_HOME`, which must match the new location. After modifying the entry, reload the listener.

   When the upgrade is complete, the script is executed for the second time. This execution:

   - Validates that the transient logical standby database (`pstby`) has been upgraded to the target release
   - Ensures `pstby` database is current
   - Performs a switchover operation, making `pstby` the primary database and the original primary (`prim`) a logical standby
   - Performs a flashback operation on `prim` to the initial GRP created by the script
   - Converts `prim` into a physical standby database
   - Shuts down `prim` in preparation for running on the upgraded binaries

   Note : During this execution of the script, the original primary database (`prim`) will be started in a new Oracle Home if you are performing an out-of-place upgrade. The best practice is:

     - To set up the new `ORACLE_HOME` to allow seamless starting of the original primary by ensuring the initialization parameter file (`init.ora`) and password files are in place in the `$ORACLE_`HOME`/dbs` directory
     - To update the `/etc/oratab` entry for the database to point to the new `ORACLE_HOME`
     - To ensure the `TNSNAMES.ORA` file is accessible to all instances of the database

```
$ ./physru sys prim pstby_upgr prim pstby 11.2.0.1.0
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Jun 07 07:43:22 2010 [0-1] Identifying rdbms software version
Jun 07 07:43:22 2010 [0-1] database prim is at version 11.1.0.7.0
Jun 07 07:43:22 2010 [0-1] database pstby is at version 11.2.0.1.0
Jun 07 07:43:23 2010 [0-1] verifying flashback database is enabled at prim and pstby
Jun 07 07:43:23 2010 [0-1] verifying available flashback restore points
Jun 07 07:43:24 2010 [0-1] verifying DG Broker is disabled
Jun 07 07:43:24 2010 [0-1] looking up prior execution history
Jun 07 07:43:24 2010 [0-1] last completed stage [2-4] using script version 0001
Jun 07 07:43:24 2010 [0-1] resuming execution of script

### Stage 3: Validate upgraded transient logical standby
Jun 07 07:43:25 2010 [3-1] database pstby is no longer in OPEN MIGRATE mode
Jun 07 07:43:25 2010 [3-1] database pstby is at version 11.2.0.1.0

### Stage 4: Switch the transient logical standby to be the new primary
Jun 07 07:43:26 2010 [4-1] waiting for pstby to catch up (this could take a while)
Jun 07 07:43:26 2010 [4-1] starting logical standby on database pstby
Jun 07 07:43:33 2010 [4-1] waiting for v$dataguard_stats view to initialize
Jun 07 07:43:33 2010 [4-1] waiting for apply lag on pstby to fall below 30 seconds
Jun 07 07:44:03 2010 [4-1] apply lag is now less than 30 seconds
Jun 07 07:44:04 2010 [4-2] switching prim to become a logical standby
Jun 07 07:44:22 2010 [4-2] prim is now a logical standby
Jun 07 07:44:22 2010 [4-3] waiting for standby pstby to process end-of-redo from primary
Jun 07 07:44:23 2010 [4-4] switching pstby to become the new primary
Jun 07 07:44:30 2010 [4-4] pstby is now the new primary

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to physical
Jun 07 07:44:31 2010 [5-1] verifying instance pstby is the only active instance
Jun 07 07:44:31 2010 [5-1] shutting down database prim
Jun 07 07:45:01 2010 [5-1] mounting database prim
Jun 07 07:45:13 2010 [5-2] flashing back database prim to restore point PRU_0000_0001
Jun 07 07:45:20 2010 [5-3] converting prim into physical standby
Jun 07 07:45:21 2010 [5-4] shutting down database prim

NOTE: Database prim has been shutdown, and is now ready to be started
      using the newer version Oracle binary.  This script requires the
      database to be mounted (on all active instances, if RAC) before calling
      this script to resume the rolling upgrade.
```

4. `prim` is now started in mount mode using the release 11.2.0.1 binary. The script is executed one last time to :

- Start Redo Apply on `prim.`
- Verify that `prim` becomes current with `pstby.`
- Perform a switchover, returning `prim` to the primary role and `pstby` to the physical standby role. In this example, the switchover was performed immediately instead of waiting to do it manually at a later time.
- Clean up from the script execution by dropping all GRPs created as part of the process.

  Note : While `prim` is shut down, the `LOG_ARCHIVE_DEST_STATE_`*`n`* initialization parameter on `pstby` pointing to `prim` will become marked invalid. Upon mounting `prim`, the invalid state of the destination should automatically clear quickly, but to reduce the delay, manually enable the destination on `pstby` (`ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_`*`n`*`=ENABLE`).

```
$ ./physru sys prim pstby_upgr prim pstby 11.2.0.1.0
Please enter the sysdba password:
### Initialize script to either start over or resume execution
Jun 07 07:48:46 2010 [0-1] Identifying rdbms software version
Jun 07 07:48:47 2010 [0-1] database prim is at version 11.2.0.1.0
Jun 07 07:48:47 2010 [0-1] database pstby is at version 11.2.0.1.0
Jun 07 07:48:48 2010 [0-1] verifying flashback database is enabled at prim and pstby
Jun 07 07:48:48 2010 [0-1] verifying available flashback restore points
Jun 07 07:48:48 2010 [0-1] verifying DG Broker is disabled
Jun 07 07:48:48 2010 [0-1] looking up prior execution history
Jun 07 07:48:49 2010 [0-1] last completed stage [5-4] using script version 0001
Jun 07 07:48:49 2010 [0-1] resuming execution of script

### Stage 6: Run media recovery through upgrade redo
Jun 07 07:48:50 2010 [6-1] upgrade redo region identified as scn range [1683261616, 1683536323]
Jun 07 07:48:53 2010 [6-1] starting media recovery on prim
Jun 07 07:48:59 2010 [6-1] confirming media recovery is running
Jun 07 07:49:09 2010 [6-1] waiting for media recovery to initialize v$recovery_progress
Jun 07 07:50:12 2010 [6-1] monitoring media recovery's progress
Jun 07 07:50:13 2010 [6-2] last applied scn 1683250353 is approaching upgrade redo start scn
1683261616
```

```
Jun 07 07:50:28 2010 [6-3] recovery of upgrade redo at 14% - estimated complete at Jun 07
07:52:54
Jun 07 07:50:43 2010 [6-3] recovery of upgrade redo at 49% - estimated complete at Jun 07
07:51:26
Jun 07 07:50:59 2010 [6-3] recovery of upgrade redo at 73% - estimated complete at Jun 07
07:51:20
Jun 07 07:51:14 2010 [6-3] recovery of upgrade redo at 82% - estimated complete at Jun 07
07:51:29
Jun 07 07:51:29 2010 [6-3] recovery of upgrade redo at 96% - estimated complete at Jun 07
07:51:32
Jun 07 07:51:45 2010 [6-4] media recovery has finished recovering through upgrade

### Stage 7: Switch back to the original roles prior to the rolling upgrade

NOTE: At this point, you have the option to perform a switchover
     which will restore prim back to a primary database and
     pstby back to a physical standby database.  If you answer 'n'
     to the question below, prim will remain a physical standby
     database and pstby will remain a primary database.

Do you want to perform a switchover? (y/n): y

Jun 07 07:51:58 2010 [7-1] continuing
Jun 07 07:52:02 2010 [7-2] waiting for v$dataguard_stats view to initialize
Jun 07 07:52:03 2010 [7-2] waiting for apply lag on prim to fall below 30 seconds
Jun 07 07:52:03 2010 [7-2] apply lag is now less than 30 seconds
Jun 07 07:52:03 2010 [7-3] switching pstby to become a physical standby
Jun 07 07:52:26 2010 [7-3] pstby is now a physical standby
Jun 07 07:52:26 2010 [7-3] shutting down database pstby
Jun 07 07:52:35 2010 [7-3] mounting database pstby
Jun 07 07:52:51 2010 [7-4] waiting for standby prim to process end-of-redo from primary
Jun 07 07:52:52 2010 [7-5] switching prim to become the new primary
Jun 07 07:52:54 2010 [7-5] prim is now the new primary
Jun 07 07:52:54 2010 [7-5] opening database prim
Jun 07 07:52:58 2010 [7-6] starting media recovery on pstby
Jun 07 07:53:05 2010 [7-6] confirming media recovery is running

### Stage 8: Statistics
script start time:                                           07-Jun-10 07:04:39
script finish time:                                          07-Jun-10 07:53:31
total script execution time:                                       +00 00:48:52
wait time for user upgrade:                                        +00 00:36:12
active script execution time:                                      +00 00:12:40
transient logical creation start time:                       07-Jun-10 07:04:46
transient logical creation finish time:                      07-Jun-10 07:05:33
primary to logical switchover start time:                    07-Jun-10 07:44:04
logical to primary switchover finish time:                   07-Jun-10 07:44:30
primary services offline for:                                      +00 00:00:26
total time former primary in physical role:                        +00 00:06:30
time to reach upgrade redo:                                        +00 00:00:16
time to recover upgrade redo:                                      +00 00:01:17
primary to physical switchover start time:                   07-Jun-10 07:51:58
physical to primary switchover finish time:                  07-Jun-10 07:52:58
primary services offline for:                                      +00 00:01:00

SUCCESS: The physical rolling upgrade is complete
```

References

1. Oracle Maximum Availability Architecture Web site
   http://www.otn.oracle.com/goto/maa
2. *Oracle Database High Availability Overview (Part #B14210)*
   http://www.oracle.com/pls/db111/db111.to_toc?partno=b28281
3. *Oracle Database High Availability Best Practices (Part B25159)*
   http://www.oracle.com/pls/db111/db111.to_toc?partno=b28282
4. *Oracle Data Guard Concepts and Administration (Part B28294)*
   http://www.oracle.com/pls/db111/db111.to_toc?partno=b28294
5. *Oracle Data Guard Broker (Part B28295)*
   http://www.oracle.com/pls/db111/db111.to_toc?partno=b28295
6. *Database Rolling Upgrade Using Transient Logical Standby: Oracle Data Guard 11g*
   http://www.oracle.com/technology/deploy/availability/pdf/maa_wp_11g_transientlogicalrollingupgrade.pdf

ORACLE

Database Rolling Upgrades Made Easy by
Using a Data Guard Physical Standby Database
October 2011
Author: Frank Kobylanski
Contributing Authors: MAA team
Editor: Viv Schupmann

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

Oracle is committed to developing practices and products that help protect the environment

Copyright © 2010, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

0109