Oracle Maximum
Availability Architecture

# Oracle Database Rolling Upgrades

## Using a Data Guard Physical Standby Database

ORACLE WHITE PAPER | AUGUST 2016

ORACLE

## Table of Contents



.

## 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 Oracle MAA recommended best practice for performing rolling database upgrades (e.g. upgrade to new patch set or new Oracle Database release) is to use the transient logical process available with Oracle Data Guard. This enables a database rolling upgrade to be performed using an existing Data Guard physical standby. A Bourne shell script named `physru`, which automates a majority of the operations of the rolling upgrade, is available for download by means of My Oracle Support Note 949322.1.

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

- It reduces downtime associated with database upgrade from potentially hours to seconds.
- 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 – substantially reducing the risk of upgrading to a new database version.
- When the upgrade process is complete, the primary database and physical standby database are both running the new Oracle release.
- Since the process is automated, the rolling database upgrade process using transient logical standby is relatively simple to deploy and use.

To maximize availability for other types of planned maintenance, 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, the database rolling upgrade process documented in this paper 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.

The database rolling upgrade process will require a brownout, albeit brief, to change the roles of the databases. If you require absolute zero down time then Oracle GoldenGate bi-directional replication must be used in place of the Data Guard database rolling upgrade process. Oracle GoldenGate provides additional flexibility and the ability to completely eliminate downtime for planned maintenance but it also requires a larger operational investment compared to using Data Guard. Refer to Oracle Database High Availability documentation for various rolling upgrade strategies for different scenarios.

## Rolling Database Upgrade Using Data Guard

Data Guard transient logical standby is the Oracle MAA best practice to execute database rolling upgrades with near-zero downtime when upgrading to new patch sets or major releases of the Oracle Database. The transient logical rolling upgrade process was first introduced in Oracle Database 11*g* Release 1 (11.1) to make it simpler to execute a database rolling upgrade using an existing physical standby database. 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.

The following high-level steps describe the transient logical database rolling upgrade process as depicted in Figure 1. Automation in Phases 2, 4, and 6 is provided by an Oracle maintained and supported script that is available from My Oracle Support. Additional details about the automation script and each step are included later in this document.

Phase 1: Prerequisites and preparation (manual):

- Review the prerequisites, limitations and best practices
- Install the new Oracle Home on all nodes in preparation for an out-of-place upgrade.
- (optional) Configure and install the database services role-change trigger and scripts

Phase 2: First physru execution

- Verifies that Data Guard Broker is disabled and FRA is configured.
- Creates a guaranteed restore point
- Converts the existing Physical Standby to a Logical Standby

Phase 3: Upgrade (manual)

- Manually perform the upgrade on the logical standby as per Oracle documentation.
- Run tests to validate the upgrade.

Phase 4: Second physru execution to switchover (APPLICATION BROWNOUT):

- Executes a switchover making the upgraded standby database the primary database.
- Executes a flashback of the original primary database to the guaranteed restore point from step 1 and shuts it down.

Phase 5: Mount old primary database with the new Oracle Home (manual):

Phase 6: Execute physru for the third and final time. This execution will:

- Start redo apply
- Prompt whether to switch back to original configuration
- Remove guaranteed restore points

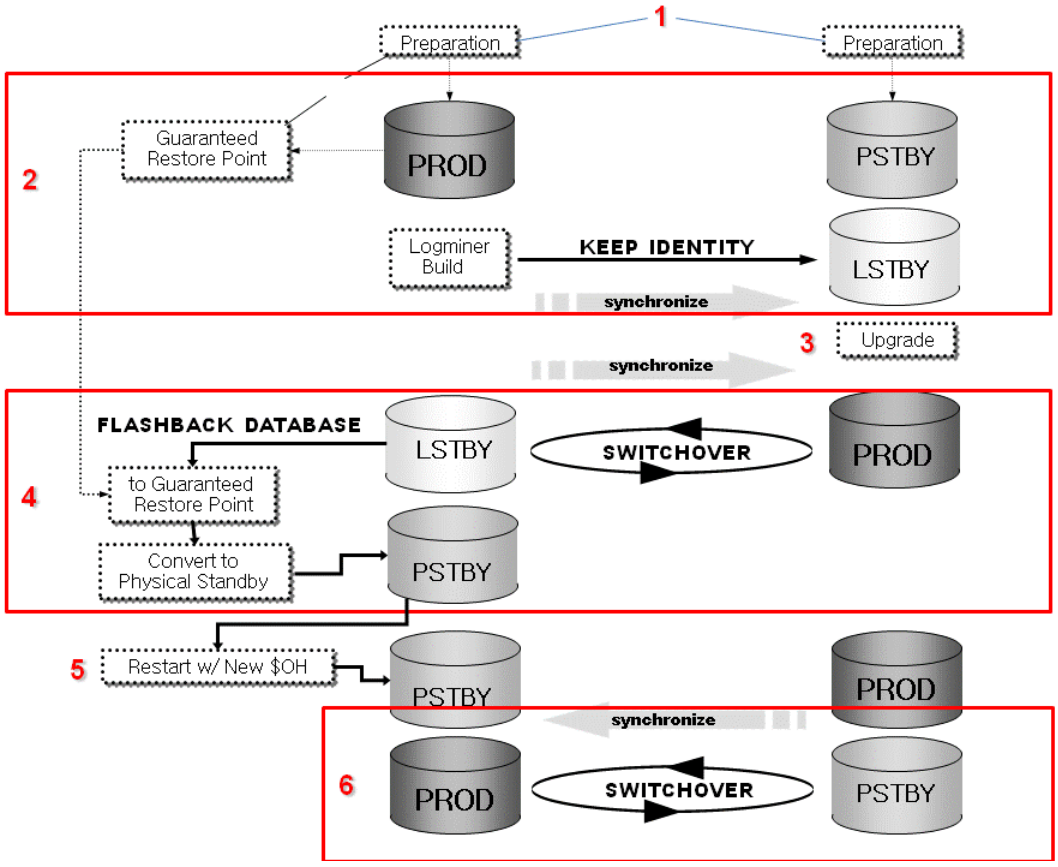



Figure 1: Transient Logical Rolling Upgrade Process

The transient logical rolling upgrade process will always result in a significant reduction in planned downtime compared to the conventional database upgrade method. Proof-of-concept tests using a transient logical standby at a large Oracle customer demonstrated that primary and standby databases can be upgraded to a new release with less than one minute of total database downtime.

*The generic upgrade steps for any Oracle database are detailed in the Oracle Database Upgrade Guide for each RDBMS version. Review this documentation before performing a rolling upgrade using a transient logical standby. The best practices provided in this paper complement the documentation, but the information in this white paper is not intended to duplicate information already provided in the documentation.*

*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` *(Phase 4 in Figure 1), normal operations can be resumed after the third and final execution of* `physru` *has completed (Phase 6 in Figure 1).*

*Note that production databases running at a minimum of Oracle Database 12.1.0.2 may utilize the new* `DBMS_ROLLING` *PL/SQL package to automate the rolling upgrade process in place of the* `physru` *script. Unlike the physru script that can be used with Data Guard,* `DBMS_ROLLING` *requires an Active Data Guard license.*

## Fast Database Switchover

While an approximate one minute brownout using the traditional switchover processing for database upgrades is acceptable for many applications, those with the most sensitive availability requirements must reduce the brownout time even further. Recent enhancements to the switchover process are available which testing has shown can reduce the brownout to 10 seconds.

### Requirements

The requirements to enable this functionality are as follows.

- Standby Redo Logs (SRLs) must be used at the standby database.
- There can be no logical bystander databases.
- Refer to MOS 949322.1 for additional requirements required for the improved logical switchover performance.

### Service Migration Trigger

Data Guard broker must be disabled during the rolling upgrade process therefore role-based services will not be automatically started on the new primary database after switchover. Manual stopping and starting of the service can add time to the application brownout. In order to achieve the fastest return to availability with the least user intervention, a script to start and stop the appropriate database service(s) on each database must be in place and called by a role change trigger. The trigger is designed so that after a database role change, the new primary database will START the application service while the new logical standby database will STOP the database service. Below are examples of the scripts which were used during testing along with the trigger that called them. The trigger must be installed in the primary database and applied through redo at the standby database prior to first execution of the physru script.

*Read-only services should not be migrated during this process because the standby database cannot be used as an Active Data Guard read-only source during this process.*

*Edit the ORACLE_HOME and service name in the scripts and the path to the script location in the trigger DDL prior to installation.*

**Script start_service.sh**

```
#!/bin/sh
export ORACLE_HOME=/u01/app/oracle/product/11.2.0.4/dbhome_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
$ORACLE_HOME/bin/srvctl start service -d ${1} -s oltpru
```

**Script stop_service.sh**

```
#!/bin/sh
export ORACLE_HOME=/u01/app/oracle/product/11.2.0.4/dbhome_1
export LD_LIBRARY_PATH=$ORACLE_HOME/lib
$ORACLE_HOME/bin/srvctl stop service -d ${1} -s oltpru
```

**Role change trigger**

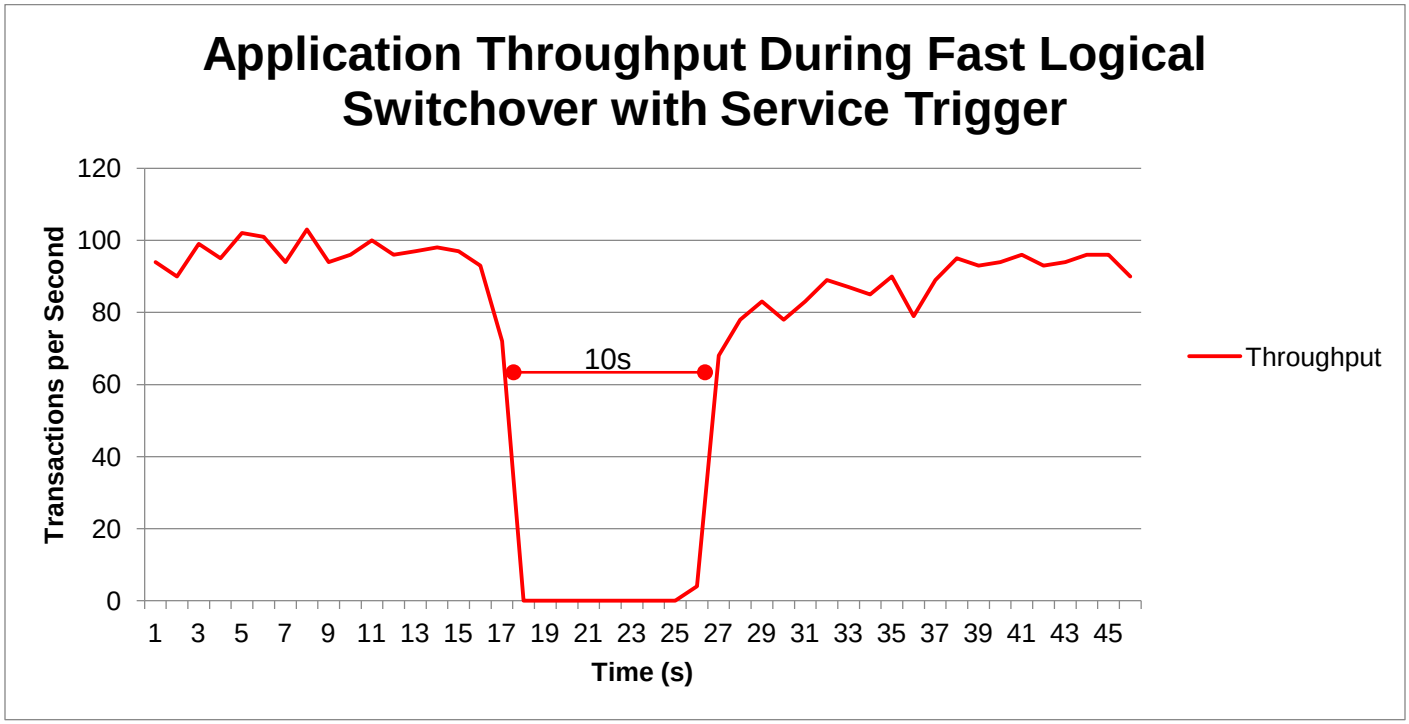
```
create or replace trigger MANAGE_SERVICE AFTER DB_ROLE_CHANGE ON DATABASE
DECLARE
role VARCHAR(30);
db_u_nm VARCHAR2(30);
BEGIN
SELECT DATABASE_ROLE INTO role FROM V$DATABASE;
select SYS_CONTEXT('USERENV','DB_UNIQUE_NAME') INTO db_u_nm FROM DUAL;
IF role = 'PRIMARY' THEN
 dbms_scheduler.create_job(
 job_name=>'publish_start',
 job_type=>'executable',
 job_action=>'<path to script>/start_service.sh',
 number_of_arguments=>1,
 enabled=>FALSE
 );
dbms_scheduler.set_attribute(name=>'publish_start',attribute=>'DATABASE_ROLE',v
  alue=>'PRIMARY');
 dbms_scheduler.set_job_argument_value('publish_start',1,db_u_nm);
 dbms_scheduler.enable('publish_start');
 dbms_scheduler.run_job('publish_start');
END IF;
IF role = 'LOGICAL STANDBY' THEN
 dbms_scheduler.create_job(
 job_name=>'publish_stop',
 job_type=>'executable',
 job_action=>'<path to script>/stop_service.sh',
 number_of_arguments=>1,
 enabled=>FALSE
 );
dbms_scheduler.set_attribute(name=>'publish_stop',attribute=>'DATABASE_ROLE',va
  lue=>'LOGICAL STANDBY');
 dbms_scheduler.set_job_argument_value('publish_stop',1,db_u_nm);
 dbms_scheduler.enable('publish_stop');
 dbms_scheduler.run_job('publish_stop');
END IF;
EXCEPTION
  WHEN OTHERS THEN
  RAISE_APPLICATION_ERROR (-20000, 'ERROR WITH TRIGGER!!!!!');
END;
/
```

*In the absence of Oracle Clusterware, the call to the scripts can be replaced with a call to DBMS_SERVICE.START_SERVICE and DBMS_SERVICE.STOP_SERVICE.*

**Performance with Trigger and Scripts**

The trigger, scripts and additional information included in MOS 949322.1 to accelerate switchover processing reduced application brownout during Oracle MAA testing to approximately 10 seconds. Timing was similar whether using RAC or single instance database type.

*Actual brownout times may vary due to a number of factors including but not limited to database activity at the time of switchover and system resources.*




## Prerequisites and Restrictions

Since this process utilizes a logical standby database, the logical standby restrictions apply. A list of the most commonly encountered restrictions follows. Please refer to Data Guard documentation for a complete list of logical standby prerequisites and restrictions (11.2|12.1).

- Data Guard Broker must be disabled.
- The 'processes' parameter must be greater than 'parallel_max_servers'+100 on the initial standby database.
- Data Guard protection mode must NOT be MAXIMUM PROTECTION.
- LOG_ARCHIVE_DEST_n for the standby database must be OPTIONAL.
- The COMPATIBLE initialization parameter must match the software release prior to the upgrade. That is, a rolling upgrade from release *X* to new release *Y* 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, *Y*. COMPATIBLE must be set to a minimum of 11.1.0 though to enable enhanced features a setting of 11.2 is required.

*Once the COMPATIBLE parameter has changed to the target database release, the database cannot be downgraded to an earlier release with flashback database nor the database downgrade procedure*

- 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* documentation appropriate for you version.

- Logical standby databases do not support Oracle Label Security.
- Logical standby databases do not fully support an Oracle E-Business Suite implementation because there are tables that contain unsupported data types.
- Data type restrictions (11.2): *
  - BFILE
  - Collections (including VARRAYS and nested tables)
  - Multimedia data types (including Spatial, Image, and Oracle Text)
  - ROWID, UROWID
  - User-defined types

*Extended Datatype Support can be utilized to mitigate data type restrictions. See My Oracle Support Note 949516.1*

- Data type restrictions (12.1) *:
  - BFILE
  - ROWID, UROWID
  - Collections (including VARRAYs and nested tables)
  - Objects with nested tables and REFs
  - The following Spatial types are not supported:
    - MDSYS.SDO_GEORASTER
    - MDSYS.SDO_TOPO_GEOMETRY
  - Identity columns
- If you have multiple standby databases, review the 'Multiple Standby Considerations' section prior to starting the rolling upgrade.
- Active Data Guard standby databases involved in the upgrade cannot be used as a read-only source during the execution of this process.

Additionally, this process requires the following prerequisites to ensure a successful execution.

- Existing physical standby database.
- Forced logging enabled. To ensure there are no unrecoverable blocks. The following query should return no rows Refer to MOS 290161.1 for additional details regarding nologging operations:

```
SQL> select NAME from V$DATAFILE where UNRECOVERABLE_CHANGE#>0;
no rows selected
```

*Note: If UNRECOVERABLE_CHANGE# is >0 for any datafile, compare the value in the primary to the value at the standby. If they are the same the datafile was copied after the unrecoverable change and no action is necessary.*

- Fast Recovery Area (FRA) configured. (for Guaranteed Restore Points (GRP) use)
- Log archive destination must be configured on each database to facilitate a switchover.
  If broker is regularly configured, standby will not be configured with a log archive destinations when the broker is disabled and should be done so manually
- fal_server parameter set for all databases.
- STANDBY_FILE_MANAGEMENT=AUTO

- DB_FILE_NAME_CONVERT set accordingly. **This is especially important for local standby databases so that files are not overwritten**.
- Static services are required to one instance of each database so that the physru script can start the databases throughout the process of database rolling maintenance. My Oracle Support Note 1387859.1, Oracle Data Guard Broker and Static Service Registration[1], can be used to assist in setting up static services if not already defined.
- Oracle Net service names which use the static service.
- For CDBs TNS services must point to the root container.
- Any existing restore points will be dropped by this process. Ensure this is acceptable for your application.
- Data Vault is not supported for upgrade. Follow MOS 1085051.1 for steps on how to handle Data Vault during and upgrade

## The Rolling Upgrade Process

The transient logical process gets its name from the fact that it begins and ends with a physical standby database while temporarily being converted to a logical standby database for the upgrade. The process can also be used to reduce downtime for other types of planned maintenance, such as partitioning a table, or encrypting data with Transparent Data Encryption, etc. A script provided by Oracle named `physru` automates many of the manual steps along with additional levels of validation that occur automatically throughout the process.

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 ru_static rustby_static ru rustby 12.1.0.2.0

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

**Static Listeners**

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. In case of RAC these need only be created for one instance and you should ensure that the node from which you execute `physru` has a valid TNS alias created to connect to these instances. The following example shows the primary database's `LISTENER.ORA` file for instance `ru1` of the Oracle RAC database `ru`:

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC =
        (SID_NAME = ru1)
        (ORACLE_HOME = /u01/app/oracle/product/11.2.0/11.2.0.4/db)
     )
  )
```

The following example shows the physical standby database's `LISTENER.ORA` file for instance `rustby1` of the Oracle RAC database `rustby`:

[1] https://support.oracle.com/epmos/faces/DocumentDisplay?id=1387859.1

```
SID_LIST_LISTENER =
  (SID_LIST =
     (SID_DESC =
        (SID_NAME = rustby1)
        (ORACLE_HOME = /u01/app/oracle/product/11.2.0/11.2.0.4/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>
```

**tnsnames.ora**

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:

```
RU_STATIC =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = <host VIP>)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SID = rustby1)
    )
 )
```

**Install software**

Install the software in a new home on all nodes of the environment.

**Prerequisites and Limitations**

Review all prerequisites and limitations listed above and take action accordingly

**Service Migration Trigger (optional)**

Modify and create the service trigger and scripts described in section ‘Fast Logical Role Transitions’

## 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.

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.

## Detailed Upgrade Steps

1. Run the "physru" Upgrade Script to completion. This first execution does the following:

```
$ physru.sh sys ru_static rustby_static ru rustby 12.1.0.2.0
```

   - Validates the environment before proceeding with the remainder of the script.
   - Creates control file backups for both the primary and the target physical standby database.
   - 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. This conversion requires that the standby is in single instance mode. The script will prompt the user to set CLUSTER_DATABASE=FALSE and restart the standby database if necessary.

During this first execution, you may receive the following warning if there are unsupported data types in the database:

```
WARN: Objects have been identified on the primary database which will not be
replicated on the transient logical standby. The complete list of objects and
their associated unsupported datatypes can be found in the
dba_logstdby_unsupported view. For convenience, this script has written the
contents of this view to a file - physru_unsupported.log.

Various options exist to deal with these objects such as:

 - disabling applications that modify these objects
 - manually resolving these objects after the upgrade
 - extending support to these objects (see My Oracle Support Note: 559353.1)

If you need time to review these options, you should enter 'n' to exit
the script.  Otherwise, you should enter 'y' to continue with the rolling
upgrade.

Are you ready to proceed with the rolling upgrade? (y/n):
```

In this case review the physru_unsupported.log file from the execution directory for details and resolve issues accordingly.

The script processing ends and returns to the command prompt.

2. Upgrade Logical Standby to Target Release.

   You can perform the upgrade procedure using upgrade scripts or by using the Database Upgrade Assistant (DBUA). Review the *Oracle Database Upgrade Guide* (12.1|11.2) for the release and/or the Database Readme before any upgrade.

*Do not change the `COMPATIBLE` initialization parameter at this point as it will remove the ability to downgrade.*

   After the upgrade to the transient logical standby database has completed, the upgraded standby database can be used to perform testing. Doing so before the switchover occurs isolates the testing from the active primary database with the added benefit of testing against production data. Use this time to ensure the critical processes will continue to work after the upgrade. Create a guaranteed restore point (GRP) prior to testing and

flashback to this GRP once testing is complete. This will ensure SQL Apply resumes at the proper point in time when all testing is finished.

*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.*

Once the upgrade and testing are complete, 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;
```

*Starting SQL Apply in this step is critical to achieving the fastest switchover possible.*

3. Next, 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 = rustby1)
        (ORACLE_HOME =
/u01/app/oracle/product/12.1.0/12.1.0.2/db)
     )
  )

$ lsnrctl reload listener
```

If Database Upgrade Assistant (DBUA) was not used for the upgrade 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. For cases where clusterware is installed, execute srvctl upgrade database -d <dbname> -o <new Oracle home> and verify the changes described have also been completed.

4. (Start of Planned Maintenance) After you complete the standby database upgrade and perform the desired testing and verification, 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:

```
$ physru.sh sys ru_static rustby_static ru rustby 12.1.0.2.0
```

- Ensures the transient logical standby database has been upgraded to the target version and is not started in `OPEN MIGRATE` mode. This could take some time, monitor the DBA_LOGSTBY_LOG view for progress
- Ensures 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.
- Performs a switchover to the upgraded transient logical standby database. The script will not attempt the switchover until SQL Apply on the standby database lags the primary database by 30 seconds or less.
    - After the switchover completes the database services can be started and the applications can reconnect. If the scripts and trigger described in 'Fast Transient Logical Switchovers' are in place, the services will be restarted automatically.
- Performs a flashback of the original primary database to the initial Guaranteed Restore Point that was created in step 1. 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.
- Converts the original primary database into a physical standby database.
- Shuts down the new physical standby database.

The script processing ends and returns to the command prompt.

5. Now prepare the new physical standby

   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. (A manual copy of the password file to the new home is still required as well as potential updates to the tnsnames.ora and listener.ora if used out of the ORACLE_HOME)

   ```
   $ srvctl upgrade database -d ru -o /u01/app/oracle/product/12.1.0/12.1.0.2/db
   ```

   Then mount the new standby database.

   Otherwise, manually mount the database with the upgraded Oracle Home that was installed in step 1. Since this is an out of place upgrade and the software has already been installed, simply change the oratab to the new Oracle home, copy the init.ora/spfile and password file to the correct location, reset the environment and mount the database. Changes to the listener.ora for the static service and tnsnames.ora may also be required depending on the configuration.

6. Lastly, the final execution of `physru` will synchronize the new standby (original primary) with all redo generated by the new primary. This includes all application activity that occurred at the original primary database while the transient logical database was being upgraded, plus all redo directly associated with the upgrade itself, and all transactions that have occurred at the new primary after the initial switchover.

   ```
   $ physru.sh sys ru_static rustby_static ru rustby 12.1.0.2.0
   ```

   This final execution of physru:

   - 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. If the script times out simply restart the script
   - 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 that you may see the following message in the alert log if you are monitoring it.*

```
ORA-19906: recovery target incarnation changed during recovery
Managed Standby Recovery not using Real Time Apply
Recovery Slave PR00 previously exited with exception 19906
```

*This is due to the remote log archive destination on the new primary timing out. You can ignore this and reconnection will eventually occur or you can set the log_archive_dest_state_n=enable on the new primary to force a reconnection.*

### Post Execution Considerations

After completion of the process review if any of the following are necessary for your environment.

» Drop the role-change trigger if it was created.

```
SQL>  drop trigger MANAGE_SERVICE;
```

» Drop the jobs that the trigger creates.

```
SQL> execute DBMS_SCHEDULER.drop_job(job_name => 'PUBLISH_STOP',
force => true);
SQL> execute DBMS_SCHEDULER.drop_job(job_name => 'PUBLISH_START',
force => true);
```

» Verify clusterware start options of each database if necessary.

» Re-enable Data Guard broker configuration if necessary.

» Remove static entries from listener.ora which were added for the process.

» Remove tnsnames.ora entries which added for the process.

## Multiple Standby Database Considerations

There are cases where you may have existing multiple standby database environment for additional data protection, HA and disaster recovery or to support an Active Data Guard reader farm. You can still use transient logical standby procedures to execute database rolling upgrade; however follow these additional steps and considerations.

**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;

*Oracle Database 12c does not require the USING CURRENT LOGFILE clause*

2. Install the new Oracle Home at each bystander standby
3. Perform the upgrade using physru as documented In the Detailed Upgrade Steps section
4. 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;

## Fallback Best Practices

**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.

**Guaranteed Restore Points**

If for any reason you need to perform a flashback before the second execution of the physru script, additional actions are required to convert the transient logical standby database back to a physical standby due to a known bug to be resolved in a future patch.

Flashback the transient logical standby to restore point PRU_0000_0002

```
SQL> STARTUP MOUNT FORCE

SQL> FLASHBACK DATABASE TO RESTORE POINT PRU_0000_0002;

SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;

SQL> SHUTDOWN ABORT
```

If the transient logical standby database had been using a new version $ORACLE_HOME, reset your environment and mount the database using the original $ORACLE_HOME.

After the database has been mounted, issue the following as SYSDBA:

```
SQL> ALTER SYSTEM SET _TRANSIENT_LOGICAL_CLEAR_HOLD_MRP_BIT=TRUE;
```

Start Redo Apply:

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

At some future time, remove the Guaranteed Restore Points created by the physru script.

**Downgrade**

To run the downgrade procedure, use the manual downgrade steps documented in the appropriate upgrade guide for the version you are upgrading to. 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 .

## 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 first GRP generated. 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.

*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.*

## Conclusion

The Transient Logical Rolling Upgrade Process can greatly reduce the impact to application availability. By converting an existing physical standby database to a logical standby database with the help of an Oracle maintained and supported script, the upgrade of both databases can be completed easily with a single catalog upgrade. With the recent further enhancements to the switchover process explained in this paper, application brownout during switchover of production to the new database version can be further reduced to meet even the most aggressive Service Level Agreements.

## Appendix A- Using EDS with a Transient Logical Standby

For an overview of Extended Datatype Support (EDS), see the MAA white paper Extended Datatype Support. EDS is useful if your database contains data types not natively supported by the SQL Apply process used by the database rolling upgrade process to replicate changes from a lower database version to a higher database version.

For details and examples of using EDS to support data types that are not natively supported by SQL Apply, see the following Support notes:

- Pre 11.2 - My Oracle Support Note 559353.1
- 11.2+ - My Oracle Support Note 949516.1

Using EDS with a transient logical is intended for use while the standby is a logical standby and once the standby is converted back to a physical standby EDS is no longer necessary.

**Preparing EDS**

Prior to converting the physical standby to a logical standby perform the following steps to setup EDS. These steps are performed on the primary database for each table that requires EDS support.

1. On the primary database, for each schema that contains a table that requires EDS support, grant execute privilege on the `DBMS_LOGSTDBY` package.

   ```
   e.g. (where OBJUSER is the schema)
   SQL> connect system/<password>
   SQL> grant execute on dbms_logstdby to OBJUSER;
   ```

2. Create the logging table on the primary database
   e.g. OBJUSER.PLAYERS_LOG

3. Create the base table trigger on the primary database
   e.g. OBJUSER.PLAYERS_PRI_TRG

4. Create the logging table trigger on the primary database
   e.g. OBJUSER.PLAYERS_STBY_TRG

*If the starting release is Oracle Database 11g Release 1 (11.1.0.6) then create all logging table triggers disabled. For example:*

```
SQL> create trigger PLAYERS_STBY_TRG disable ...
```

In Oracle Database 11g Release 1 (11.1.0.6), logging table triggers will generate warnings during compilation and will impact operations on the base table if they are not disabled. Invalid logging table triggers will be successfully compiled and enabled on the logical standby database once it has been upgraded to the later release.

**Configuring EDS Logging Table Triggers**

After the physical standby has become a logical standby and the logical standby has been upgraded configure the logging table triggers. These commands are run on the upgraded logical standby database for each logging table trigger used by EDS.

The following steps must be done prior to starting SQL Apply for the first time. If SQL Apply is started before these steps are performed, updates performed on the primary database to EDS-supported tables during the upgrade will not be applied to the upgraded logical standby database.

1. As SYS, temporarily disable GUARD, then enable and compile logging table triggers

```
SQL> alter session disable guard;
SQL> alter trigger OBJUSER.PLAYERS_STBY_TRG compile;
SQL> alter trigger OBJUSER.PLAYERS_STBY_TRG enable;
SQL> alter session enable guard;
```

2. Set the FIRE_ONCE trigger firing property for all logging table triggers to FALSE

```
SQL> execute dbms_ddl.set_trigger_firing_property('OBJUSER',
'PLAYERS_STBY_TRG', fire_once => FALSE);
```

**Removing EDS**

Once the switchover takes place and the upgraded logical standby becomes the new primary, drop all the EDS objects on the primary.

```
SQL> drop trigger OBJUSER.PLAYERS_STBY_TRG;
SQL> drop trigger OBJUSER.PLAYERS_PRI_TRG;
SQL> drop table OBJUSER.PLAYERS_LOG;
```

## Appendix B: Sample output of the first execution of the physru script

```
### Initialize script to either start over or resume execution
Jul 21 16:07:18 2016 [0-1] Identifying rdbms software version
Jul 21 16:07:18 2016 [0-1] database ru is at version 11.2.0.4.0
Jul 21 16:07:18 2016 [0-1] database rustby is at version 11.2.0.4.0
Jul 21 16:07:18 2016 [0-1] verifying fast recovery area is enabled at ru and rustby
Jul 21 16:07:18 2016 [0-1] verifying available flashback restore points
Jul 21 16:07:18 2016 [0-1] verifying DG Broker is disabled
Jul 21 16:07:19 2016 [0-1] looking up prior execution history
Jul 21 16:07:19 2016 [0-1] purging script execution state from database ru
Jul 21 16:07:19 2016 [0-1] purging script execution state from database rustby
Jul 21 16:07:20 2016 [0-1] starting new execution of script

### Stage 1: Backup user environment in case rolling upgrade is aborted
Jul 21 16:07:20 2016 [1-1] stopping media recovery on rustby
Jul 21 16:07:21 2016 [1-1] creating restore point PRU_0000_0002 on database rustby
Jul 21 16:07:35 2016 [1-1] backing up current control file on rustby
Jul 21 16:07:35 2016 [1-1] created backup control file
  /u01/app/oracle/product/11.2.0.4_160119DBPSU/dbhome_1/dbs/PRU_0002_rustby_f.f
Jul 21 16:07:35 2016 [1-1] creating restore point PRU_0000_0002 on database ru
Jul 21 16:07:37 2016 [1-1] backing up current control file on ru
Jul 21 16:07:38 2016 [1-1] created backup control file
  /u01/app/oracle/product/11.2.0.4_160119DBPSU/dbhome_1/dbs/PRU_0002_ru_f.f

NOTE: Restore point PRU_0000_0002 and backup control file PRU_0002_rustby_f.f
      can be used to restore rustby 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
Jul 21 16:07:39 2016 [2-1] verifying RAC is disabled at rustby

WARN: rustby 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 rustby2 in mounted mode.  This can be accomplished
      with the following steps:

        1) Shutdown all instances other than instance rustby2.
           eg: srvctl stop instance -d rustby -i rustby1 -o abort

        2) On instance rustby2, set the cluster_database parameter to FALSE.
           eg: SQL> alter system set cluster_database=false scope=spfile;

        3) Shutdown instance rustby2.
           eg: SQL> shutdown abort;

        4) Startup instance rustby2 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):

Jul 21 16:08:14 2016 [2-1] continuing
Jul 21 16:08:14 2016 [2-1] verifying RAC is disabled at rustby
Jul 21 16:08:14 2016 [2-1] verifying database roles
Jul 21 16:08:14 2016 [2-1] verifying physical standby is mounted
Jul 21 16:08:14 2016 [2-1] verifying database protection mode
Jul 21 16:08:15 2016 [2-1] verifying transient logical standby datatype support
Jul 21 16:08:16 2016 [2-2] starting media recovery on rustby
Jul 21 16:08:22 2016 [2-2] confirming media recovery is running
Jul 21 16:08:22 2016 [2-2] waiting for apply lag to fall under 30 seconds
Jul 21 16:08:28 2016 [2-2] apply lag measured at 6 seconds
Jul 21 16:08:28 2016 [2-2] stopping media recovery on rustby
Jul 21 16:08:29 2016 [2-2] executing dbms_logstdby.build on database ru
Jul 21 16:08:56 2016 [2-2] converting physical standby into transient logical standby
Jul 21 16:09:01 2016 [2-3] opening database rustby
Jul 21 16:09:02 2016 [2-4] configuring transient logical standby parameters for rolling upgrade
Jul 21 16:09:03 2016 [2-4] starting logical standby on database rustby
Jul 21 16:09:10 2016 [2-4] enabling log archive destination to database rustby
```

```
Jul 21 16:09:10 2016 [2-4] waiting until logminer dictionary has fully loaded
Jul 21 16:11:52 2016 [2-4] dictionary load 33% complete
Jul 21 16:12:02 2016 [2-4] dictionary load 62% complete
Jul 21 16:12:12 2016 [2-4] dictionary load 75% complete
Jul 21 16:13:12 2016 [2-4] dictionary load is complete
Jul 21 16:13:13 2016 [2-4] waiting for apply lag to fall under 30 seconds
Jul 21 16:13:20 2016 [2-4] apply lag measured at 7 seconds

NOTE: Database rustby 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 rustby 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 rustby2, set the cluster_database parameter to TRUE.
          eg: SQL> alter system set cluster_database=true scope=spfile;

          2) Shutdown instance rustby2.
          eg: SQL> shutdown abort;

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

## Appendix C: Sample output of the second execution of the physru script

```
### Initialize script to either start over or resume execution
Jul 21 16:16:45 2016 [0-1] Identifying rdbms software version
Jul 21 16:16:45 2016 [0-1] database ru is at version 11.2.0.4.0
Jul 21 16:16:45 2016 [0-1] database rustby is at version 12.1.0.2.0
Jul 21 16:16:46 2016 [0-1] verifying fast recovery area is enabled at ru and rustby
Jul 21 16:16:46 2016 [0-1] verifying available flashback restore points
Jul 21 16:16:46 2016 [0-1] verifying DG Broker is disabled
Jul 21 16:16:46 2016 [0-1] looking up prior execution history
Jul 21 16:16:46 2016 [0-1] last completed stage [2-4] using script version 0002
Jul 21 16:16:46 2016 [0-1] resuming execution of script

### Stage 3: Validate upgraded transient logical standby
Jul 21 16:16:46 2016 [3-1] database rustby is no longer in OPEN MIGRATE mode
Jul 21 16:16:46 2016 [3-1] database rustby is at version 12.1.0.2.0

### Stage 4: Switch the transient logical standby to be the new primary
Jul 21 16:16:48 2016 [4-1] waiting for rustby to catch up (this could take a while)
Jul 21 16:16:49 2016 [4-1] starting logical standby on database rustby
Jul 21 16:16:54 2016 [4-1] waiting for apply lag to fall under 30 seconds
Jul 21 16:16:58 2016 [4-1] apply lag measured at 4 seconds
Jul 21 16:17:00 2016 [4-2] using fast switchover optimizations
NOTE: A switchover is about to be performed which will incur a brief outage
      of the primary database.  If you answer 'y' to the question below,
      database rustby will become the new primary database, and database ru
      will be converted into a standby in preparation for upgrade.  If you answer
      'n' to the question below, the script will exit, leaving the databases in
      their current roles.
Are you ready to proceed with the switchover? (y/n): y

Jul 21 16:17:16 2016 [4-2] continuing
Jul 21 16:17:00 2016 [4-2] switching ru to become a logical standby
Jul 21 16:17:16 2016 [4-2] ru is now a logical standby
Jul 21 16:17:16 2016 [4-2] waiting for standby rustby to process end-of-redo from primary
Jul 21 16:17:16 2016 [4-2] switching rustby to become the new primary
Jul 21 16:17:18 2016 [4-2] rustby is now the new primary

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to physical
Jul 21 16:17:20 2016 [5-1] verifying instance ru1 is the only active instance

WARN: ru 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 ru1.
           eg: srvctl stop instance -d ru -i ru2 -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):

Jul 21 16:17:46 2016 [5-1] continuing
Jul 21 16:17:46 2016 [5-1] verifying instance ru1 is the only active instance
Jul 21 16:17:46 2016 [5-1] shutting down database ru
Jul 21 16:18:16 2016 [5-1] mounting database ru
Jul 21 16:18:28 2016 [5-2] flashing back database ru to restore point PRU_0000_0002
Jul 21 16:18:35 2016 [5-3] converting ru into physical standby
Jul 21 16:18:37 2016 [5-4] shutting down database ru

NOTE: Database ru 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 ru 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 ru
```

```
eg: srvctl start database -d ru -o mount
```

## Appendix D: Sample output of the final execution of the physru script

```
### Initialize script to either start over or resume execution
Jul 21 16:23:00 2016 [0-1] Identifying rdbms software version
Jul 21 16:23:00 2016 [0-1] database ru is at version 12.1.0.2.0
Jul 21 16:23:00 2016 [0-1] database rustby is at version 12.1.0.2.0
Jul 21 16:23:00 2016 [0-1] verifying fast recovery area is enabled at ru and rustby
Jul 21 16:23:01 2016 [0-1] verifying available flashback restore points
Jul 21 16:23:01 2016 [0-1] verifying DG Broker is disabled
Jul 21 16:23:01 2016 [0-1] looking up prior execution history
Jul 21 16:23:01 2016 [0-1] last completed stage [5-4] using script version 0002
Jul 21 16:23:01 2016 [0-1] resuming execution of script

### Stage 6: Run media recovery through upgrade redo
Jul 21 16:23:01 2016 [6-1] upgrade redo region identified as scn range [4180883, 4205367]
Jul 21 16:23:01 2016 [6-1] enabling log archive destination to database ru
Jul 21 16:23:02 2016 [6-1] starting media recovery on ru
Jul 21 16:23:08 2016 [6-1] confirming media recovery is running
Jul 21 16:23:18 2016 [6-1] waiting for media recovery to initialize v$recovery_progress
Jul 21 16:24:40 2016 [6-1] monitoring media recovery's progress
Jul 21 16:24:42 2016 [6-2] last applied scn 4170971 is approaching upgrade redo start scn 4180883
Jul 21 16:24:58 2016 [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 ru back to a primary database and
     rustby back to a physical standby database.  If you answer 'n'
     to the question below, ru will remain a physical standby
     database and rustby will remain a primary database.

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

Jul 21 16:33:02 2016 [7-1] continuing
Jul 21 16:33:02 2016 [7-2] verifying instance rustby2 is the only active instance

WARN: rustby 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 rustby2.
           eg: srvctl stop instance -d rustby -i rustby1

      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):

Jul 21 16:33:54 2016 [7-2] continuing
Jul 21 16:33:54 2016 [7-2] verifying instance rustby2 is the only active instance
Jul 21 16:33:54 2016 [7-2] waiting for apply lag to fall under 30 seconds
Jul 21 16:34:04 2016 [7-2] apply lag measured at 10 seconds
Jul 21 16:34:04 2016 [7-3] switching rustby to become a physical standby
Jul 21 16:34:07 2016 [7-3] rustby is now a physical standby
Jul 21 16:34:07 2016 [7-3] shutting down database rustby
Jul 21 16:34:07 2016 [7-3] mounting database rustby
Jul 21 16:34:18 2016 [7-3] starting media recovery on rustby
Jul 21 16:34:25 2016 [7-3] confirming media recovery is running
Jul 21 16:34:25 2016 [7-3] waiting for standby ru to process end-of-redo from primary
Jul 21 16:34:25 2016 [7-3] switching ru to become the new primary
Jul 21 16:34:26 2016 [7-3] ru is now the new primary
Jul 21 16:34:26 2016 [7-3] opening database ru

NOTE: Database ru has completed the switchover to the primary role, but
      instance ru1 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 ru1.
        eg: srvctl stop instance -d ru -i ru2

        2) Startup and open all inactive instances for database ru.
```

```
        eg: srvctl start database -d ru

NOTE: Database rustby 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 rustby
        eg: srvctl start database -d rustby -o mount

### Stage 8: Statistics
script start time:                                    21-Jul-16 16:07:21
script finish time:                                   21-Jul-16 16:34:41
total script execution time:                                +00 00:27:20
wait time for user upgrade:                                 +00 00:03:26
active script execution time:                               +00 00:23:54
transient logical creation start time:                21-Jul-16 16:08:15
transient logical creation finish time:               21-Jul-16 16:09:01
primary to logical switchover start time:             21-Jul-16 16:16:58
logical to primary switchover finish time:            21-Jul-16 16:17:19
primary services offline for:                               +00 00:00:21
total time former primary in physical role:                 +00 00:14:13
time to reach upgrade redo:
time to recover upgrade redo:
primary to physical switchover start time:            21-Jul-16 16:33:00
physical to primary switchover finish time:           21-Jul-16 16:34:41
primary services offline for:                               +00 00:01:41

SUCCESS: The physical rolling upgrade is complete
```

**Oracle Corporation, World Headquarters**
500 Oracle Parkway
Redwood Shores, CA 94065, USA

**Worldwide Inquiries**
Phone: +1.650.506.7000
Fax: +1.650.506.7200

ORACLE

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Integrated Cloud Applications & Platform Services



Oracle Database Rolling Upgrades
August 2016