Oracle database upgrade using Logical Standby overview

I have used autoupgrade for DB upgrade (we can perform Manual upgrade also) and I have used physru script (oracle provided script My Oracle Support Note 949322.1) for logical standby conversion and other tasks to reduce the downtime. Also I have not included the basic prechecks that we do for 19c upgrade (like fixing invalid objects or any other items that preupgrade.jar reports)
Here are the high level steps

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 (autoupgrade/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)DOWNTIME

» Executes a switchover making the upgraded standby database as 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 (this switchover is optional if we can run the apps by pointing to the new primary)

» Remove guaranteed restore points

➔ 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), normal operations can be resumed by pointing your application services to new primary. Switchback to Original Primary is optional if we are ok for the second downtime we can do it else we can do this on later point of time

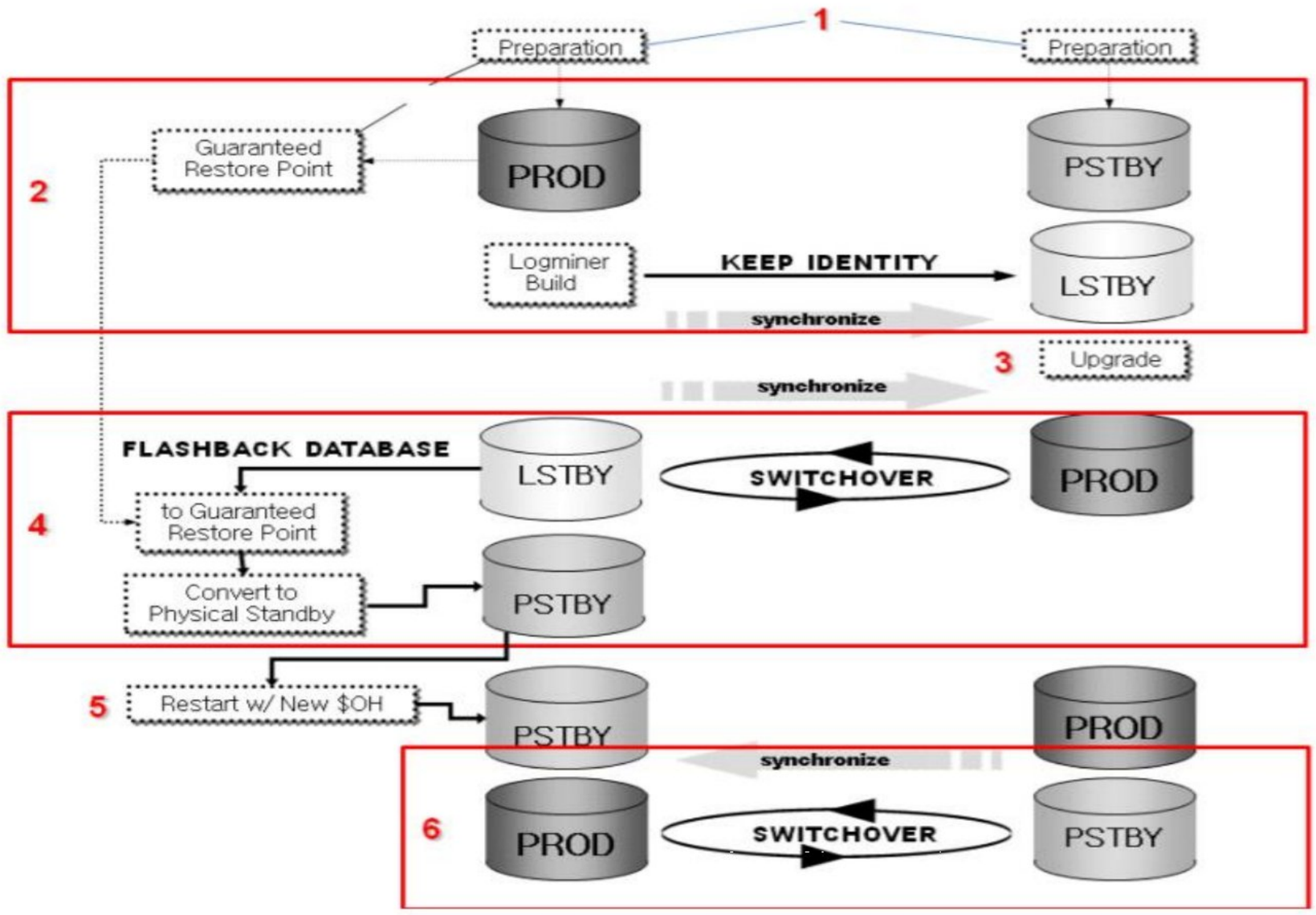

Figure 1: Transient Logical Rolling Upgrade Process

Let's first check if the Database is Qualified for Logical standby upgrade.

1. Find list of objects which are not supported by logical standby apply, we can use below query to check the unsupported object details

```
select distinct owner, table_name from dba_logstdby_unsupported order by owner,table_name;
```

If the Above query reports any tables, we need to find why they are marked as unsupported and see if we can make them supported.

2. Check the reason for un supported (pass the any one of the object name from above output)

```
select column_name,data_type from dba_logstdby_unsupported where owner='&OWNER' AND
TABLE_NAME='&TABLE_NAME';
```

Here are the top 3 reasons for unsupported objects

- ➔ Tables with missing Primary Keys – check with app if we can add primary key
- ➔ Tables/Columns names with more than 30 characters - LGSBY: 12.2 Tables with Long Identifier> 30 Char Is Showing in DBA_Logstdby_Unsupported (Doc ID 2536695.1)
- ➔ Tables with unsupported data types - (ROWID,UROWID)

3. **Find list of objects which doesn't have unique key row**

```
COL OWNER FOR A10
COL TABLE_NAME FOR A30
SET LINES 190
SELECT OWNER, TABLE_NAME,BAD_COLUMN FROM DBA_LOGSTDBY_NOT_UNIQUE;
```

```
---- sample output on test database ---
OWNER      TABLE_NAME                     B
---------- ------------------------------ -
KOTI       TEST                           N
```

Note – if The B column value say N it is fine to ignore and proceed, if it say 'Y' we must add unique key for those tables. In this case I have proceeded no issue observed.

What options available if we can't fix the unsupported object…?

If we cannot fix the unsupported objects we can try configuring other replication methods like Goldengate for these unsupported objects, we must accept this process requires an additional effort and licensing cost may involve.

We can try export/import if the size of the unsupported is small. The size is important here because we can only do export and import after the app stops modifying this objects and before initiating the switchover so this adds additional downtime(switchover time + export&import time), if it is taking more time than the offline upgrade it makes no sense to use this method

Or we can check with app team if they can hold the app to make the changes to only this unsupported objects.

What if we don't take any action and proceeded for upgrade..?

If we don't take any action for the unsupported objects and proceeded for the upgrade the data will not be replicated for the supported objects so **there will be data loss** from the moment we converted the physical standby to logical standby and until we do the upgrade of the logical standby and switchover (aprox 2 hrs).

Transient logical standby rolling upgrade process - Important Considerations

1) The "log miner" is the critical part of the entire process (The log miner will extract the SQL commands from the log files and apply it to the open logical standby database)

2) At the very beginning we create Guaranteed Restore Points (GRP) on both the primary database and the physical standby database that can be used to flashback to beginning of the process or any other intermediate steps along the way.

3) **Support for Data Types That Lack Native Redo-Based Support**

The Extended Datatype Support (EDS) feature provides a mechanism for logical standbys to support certain data types that lack native redo-based support. For example, tables with SDO_GEOMETRY columns can be replicated using EDS. (Source tables must have a primary key.)You can query the DBA_LOGSTDBY_EDS_SUPPORTED view to find out which tables are candidates for EDS.

If any table is reported under the above view we have to add EDS support for the tables using below after converting the physical standby to logical standby and we can remove EDS support post upgrade

```
begin
 execute immediate 'ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE INDEX)
COLUMNS';
 for i in (select owner, table_name from DBA_LOGSTDBY_EDS_SUPPORTED order by 1,2)
 loop
        DBMS_LOGSTDBY.EDS_ADD_TABLE(i.owner, i.table_name);
 end loop;
end;
```

--- To remove EDS support

```
declare vCnt Number := 0;
begin
 for i in (select owner, table_name from DBA_LOGSTDBY_EDS_SUPPORTED order by 1,2)
 loop
        DBMS_LOGSTDBY.EDS_REMOVE_TABLE(i.owner, i.table_name);
 end loop;
 select count(1) into vCnt from dba_capture ;
 if vCnt = 0 then
 begin
        execute immediate 'ALTER DATABASE DROP SUPPLEMENTAL LOG DATA (PRIMARY KEY, UNIQUE
INDEX) COLUMNS';
        execute immediate 'ALTER DATABASE DROP SUPPLEMENTAL LOG DATA';
 exception
        when others then
                null ;
 end;
 end if;
end;
/"
```

4) 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. Which means we need to define **a service migration trigger** to start the services on the new primary manually (including the golden gate service) otherwise application connections will fail.

----- Here is the sample code for Role change trigger ------

```
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;
/
```

Let's Run Prechecks with Autoupgrade utility – before running configure Autoupgrade

Configure Autoupgrade utility

Autoupgrade utility autoupgrade.jar file exists by default under $ORACLE_HOME/rdbms/admin directory from Oracle 19.3 release onwards, however Oracle strongly recommends to download the latest AutoUpgrade version before doing the upgrade.

Download the latest autoupgrade.jar file from Oracle supports **MOS Document 2485457.1.**

Replace the autoupgrade.jar with the latest version downloaded

```
$ mv $ORACLE_HOME/rdbms/admin/autoupgrade.jar $ORACLE_HOME/rdbms/admin/autoupgrade.jar-bkp
$ cp /tmp/autoupgrade.jar $ORACLE_HOME/rdbms/admin/
$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/rdbms/admin/autoupgrade.jar -version
Sample Output:
==============
build.hash e84c9c2
build.version 19.10.0
build.date 2020/10/23 10:36:46
```

```
build.max_target_version 19
build.supported_target_versions 12.2,18,19
build.type production
```

Validate the Java version

Java version should be 1.**8** or later, which is available by default in Oracle Database homes from release 12.1.0.2 and latest.

```
$ $ORACLE_HOME/jdk/bin/java –version

Sample Output:
==============
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)
```

Create the configuration file

Create a directory to hold all upgrade config and log files.

```
$ mkdir /u01/19c-autoupg

$ cd /u01/19c-autoupg
```

Create the sample config file

```
cd /u01/19c-autoupg
export ORACLE_HOME=<19c db home>
export PATH=$PATH:$ORACLE_HOME/jdk/bin

$ORACLE_HOME/rdbms/admin/autoupgrade.jar -create_sample_file config
Sample output:
==============
Created sample configuration file /u01/19c-autoupg/sample_config.cfg
```

Copy the sample config file and make the necessary changes as per the database environment.

```
cd /u01/19c-autoupg
cp sample_config.cfg dbname_db_config.cfg
vi dbanme_db_config.cfg
```

Sample configuration file (For a single database):

```
$cat Dev_db_config.cfg
global.autoupg_log_dir=/u01/19c-autoupg/upg_logs
#
# Database cdbdev
#
upg1.dbname=dev
upg1.start_time=NOW
```

```
upg1.source_home=<12C db home>
upg1.target_home=<19c db home>
upg1.sid=dev
upg1.log_dir=/u01/19c-autoupg/upg_logs/dev
upg1.upgrade_node=new19c
upg1.target_version=19.12
upg1.run_utlrp=yes
upg1.timezone_upg=yes
```

Analyze the database (running pre upgrade jar with autoupgrade)

Autoupgrade Analyze mode checks your database to see if it is ready for the upgrade. This will reads data from the database and does not perform any updates.Execute autoupgrade in analyze mode with the below syntax,

```
export ORACLE_HOME=<19c DB Home>
export PATH=$PATH:$ORACLE_HOME/jdk/bin
cd <upgrade directory>
$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/rdbms/admin/autoupgrade.jar -config
dbname_db_config.cfg -mode ANALYZE
```

We can monitor, manage and control the jobs from the **autoupgrade console**.

Example:

lsj – to list the jobs
status – to show the job status
tasks – shows the tasks executing

All Analyze logs are created under **autoupg_log_dir** defined in configuration file

We can review the html file **(dbname_preupgrade.html)** which will list all precheck errors, warnings and recommendations.

Review Preupgrade.log

- ➔ No action items under required/mandatory action tab.
- ➔ No invalid objects under sys/system schema.
- ➔ Recyclebin should be empty.
- ➔ Execute the below if pre-upgrade warns Oracle Streams

--------- Some more tasks before starting the Upgrade ----------

- ➔ Validate Level 1 backup/archive backups are running fine.
- ➔ Run stats gather job and purge Recyclebin (to save time run it before 4 hrs of upgrade start time)

```
SQL> exec dbms_stats.gather_dictionary_stats;
```

```
SQL> purge dba_recyclebin;
```

Run the autoupgrade in fixup mode, it will run the preupgrade fixup script

```
export ORACLE_HOME=<19c DB Home>
export PATH=$PATH:$ORACLE_HOME/jdk/bin
cd <upgrade directory>
$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/rdbms/admin/autoupgrade.jar -config
dbname_db_config.cfg -mode fixups
```

Actual upgrade steps starts

➔ Create first restore point on Standby and then create on the primary side this GRP is optional just for the safe side, physru script will create the required GRP

---- On standby side ---

```
SQL> create restore point pre_upgrade_db19c_standby guarantee flashback database;
```

----on Primary side ----

```
SQL> create restore point pre_upgrade_db19c_Prim guarantee flashback database;
```

➔ Disable the Data Guard Broker configuration

If no lag is identified and if there are no pending redo to apply then Cancel MRP on Standby

==========

```
DGMGRL> show configuration verbose;
DGMGRL> DISABLE CONFIGURATION;
DGMGRL> SHOW CONFIGURATION verbose;
```

Both Primary and standby:

```
SQL> select name,open_mode,database_role from v$database;
SQL> show parameter DG_BROKER_START
SQL> ALTER SYSTEM SET DG_BROKER_START=FALSE scope=both sid='*';
SQL> show parameter DG_BROKER_START
```

Physru1 - CONVERT PHYSICAL STANDBY TO LOGICAL STANDBY

Call Physru script – 1st time (Run from Primary server)

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>

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

--- this is what I used to run --
$ physru_v3.sh sys testdb_Primaryhost testdb_standbyhost testdb_Primaryhost testdb_standbyhost 19.0.0.0.0

This step Converts Existing Physical standby to Transient logical standby - i.e Sets up Transient logical standby parameters for rolling upgrade

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.

---- Output of first iteration ----

I have faced an issue, standby instance crashed with ora-600 due to a Bug, workaround is to disable the BCT, bug is fixed in 19.10 since we are upgrading to 19.12 we can enable BCT post upgrade

---- below bug -- work around disable BCT --

Standby MRP fails with ORA-00600: Internal Error Code, Arguments: [krccckp_scn_low] (Doc ID 2467729.1

```
## Initialize script to either start over or resume execution
Sep 09 08:05:52 2021 [0-1] Identifying rdbms software version
Sep 09 08:05:52 2021 [0-1] database testdb_Primaryhost is at version 12.2.0.1.0
Sep 09 08:05:52 2021 [0-1] database testdb_standbyhost is at version 12.2.0.1.0
Sep 09 08:05:53 2021 [0-1] verifying fast recovery area is enabled at testdb_Primaryhost and
testdb_standbyhost
Sep 09 08:05:53 2021 [0-1] verifying backup location at testdb_Primaryhost and
testdb_standbyhost
Sep 09 08:05:53 2021 [0-1] verifying available flashback restore points
Sep 09 08:05:54 2021 [0-1] verifying DG Broker is disabled
Sep 09 08:05:54 2021 [0-1] looking up prior execution history
Sep 09 08:05:54 2021 [0-1] last completed stage [2-1] using script version 0003

WARN: The last execution of this script either exited in error or at the
      user's request.  At this point, there are three available options:
```

```
        1) resume the rolling upgrade where the last execution left off
        2) restart the script from scratch
        3) exit the script

      Option (2) assumes the user has restored the primary and physical
      standby back to the original configuration as required by this script.

Enter your selection (1/2/3): 1

Sep 09 08:05:59 2021 [0-1] resuming execution of script
Sep 09 08:05:59 2021 [2-2] verifying RAC is disabled at testdb_standbyhost
Sep 09 08:06:00 2021 [2-2] stopping media recovery on testdb_standbyhost
Sep 09 08:06:01 2021 [2-2] starting media recovery on testdb_standbyhost
Sep 09 08:06:07 2021 [2-2] confirming media recovery is running
Sep 09 08:06:08 2021 [2-2] waiting for apply lag to fall under 30 seconds
Sep 09 08:06:15 2021 [2-2] apply lag measured at 7 seconds
Sep 09 08:06:16 2021 [2-2] stopping media recovery on testdb_standbyhost
Sep 09 08:06:17 2021 [2-2] executing dbms_logstdby.build on database testdb_Primaryhost
Sep 09 08:06:23 2021 [2-2] converting physical standby into transient logical standby
Sep 09 08:06:27 2021 [2-3] opening database testdb_standbyhost
Sep 09 08:06:31 2021 [2-4] configuring transient logical standby parameters for rolling
upgrade
Sep 09 08:06:31 2021 [2-4] starting logical standby on database testdb_standbyhost
Sep 09 08:06:38 2021 [2-4] enabling log archive destination to database testdb_standbyhost
Sep 09 08:06:38 2021 [2-4] waiting until logminer dictionary has fully loaded
Sep 09 08:06:49 2021 [2-4] dictionary load 24% complete
Sep 09 08:06:59 2021 [2-4] dictionary load 99% complete
Sep 09 08:07:30 2021 [2-4] dictionary load is complete
Sep 09 08:07:32 2021 [2-4] waiting for apply lag to fall under 30 seconds
Sep 09 08:07:48 2021 [2-4] apply lag measured at 15 seconds

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

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

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

```
set serveroutput on
execute dbms_logstdby.build;

PL/SQL procedure successfully completed. <----

SQL> SELECT * FROM V$LOGSTDBY_STATE WHERE STATE='LOADING DICTIONARY';

no rows selected
```

```
SQL>
```

Verify sync (try to insert into test table on primary and check from standby)

----- On Primary ----

```
testdb> insert into koti.test (customer_id) values (3);

1 row created.

testdb> commit;

Commit complete.

testdb> select * from koti.test;

CUSTOMER_ID
-----------
          1
          2
          3
```

---- On standby ----

```
testdb> select * from koti.test;

CUSTOMER_ID
-----------
          1
          2
          3

testdb> select guard_status from v$database;

GUARD_S
-------
ALL
```

The guard_status column protects the data from being changed. There are three values:

ALL - All users other than SYS are prevented from making changes to any data in the database.

STANDBY - All users other than SYS are prevented from making changes to any database object being maintained by logical standby.

NONE - Indicates normal security for all data in the database.

Upgrade Logical standby database using autoupgrade (we can do manual upgrade also)

Autoupgrade Deploy mode performs the actual upgrade of the database from preugrade source database analysis to post-upgrade checks.

Note: Before deploying the upgrade, you must have a backup plan in place still it will create GRP.

Execute the autoupgrade in DEPLOY mode using the below syntax

---- Run autoupgrade deploy to upgrade the database---

$ORACLE_HOME/jdk/bin/java -jar $ORACLE_HOME/rdbms/admin/autoupgrade.jar -config /u01/bin/oracle/upg1/sample_config.cfg -mode DEPLOY

```
upg> lsj
+----+-------+---------+---------+-------+--------------+--------+-------------+
|Job#|DB_NAME|    STAGE|OPERATION| STATUS|    START_TIME| UPDATED|      MESSAGE|
+----+-------+---------+---------+-------+--------------+--------+-------------+
| 102|testdb|DBUPGRADE|EXECUTING|RUNNING|21/09/08 06:05|06:14:47|21%Upgraded |
+----+-------+---------+---------+-------+--------------+--------+-------------+
```

Once the upgrade process is started consider monitoring the logs to see the progress of the upgrade. Autoupgrade logs are available under,

```
/bb/bin/oracle/cfgtoollogs/autoupgrade/cfgtoollogs/upgrade/auto/status/status.html
/bb/bin/oracle/cfgtoollogs/autoupgrade/cfgtoollogs/upgrade/auto/status/status.log
```

Monitor the **upg_summary.log** for the status of Upgrade activity.

Timezone file upgrade and database recompilation has already completed by the autoupgrade utility as the below values are adjusted as “yes” in the config file,

```
upg1.run_utlrp=yes =yes  # yes(default) to run utlrp as part of upgrade
upg1.timezone_upg=yes # yes(default) to upgrade timezone if needed
```

Post Upgrade validations (on logical standby)

- ➔ Check invalid objects and Datapatch status
- ➔ **Check the Timezone version and validate the DB upgrade**

Start SQL Apply on upgraded Logical standby and validate the sync status

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

Database altered.

testdb> !ps -ef | grep lsp
oracle   114953      1 25 06:52 ?        00:00:02 ora_lsp0_testdb
oracle   115005 113835  0 06:52 pts/0    00:00:00 /usr/bin/ksh -c ps -ef | grep lsp
oracle   115007 115005  0 06:52 pts/0    00:00:00 grep lsp
```

---- update test table on primary ---

```
testdb> insert into koti.test (customer_id) values (4);

1 row created.

testdb> commit;

Commit complete.

testdb> insert into koti.test (customer_id) values (5);

1 row created.

testdb> commit;

Commit complete.
```

----- onstandby ---

```
testdb> select * from koti.test;

CUSTOMER_ID
-----------
           1
           2
           3
           5
           4
testdb> select sequence#,first_change#,next_change#,timestamp,applied,blocks,block_size from
dba_logstdby_log;

 SEQUENCE# FIRST_CHANGE# NEXT_CHANGE# TIMESTAMP         APPLIED      BLOCKS BLOCK_SIZE
---------- ------------- ------------ ------------------ --------- ---------- ----------
         14       3210766      3211131 08-SEP-21       YES            35590          512
         15       3211131      3211148 08-SEP-21       YES               19          512
         16       3211148      3216408 08-SEP-21       YES             3264          512
         17       3216408      3217152 08-SEP-21       YES              489          512
         18       3217152      3217423 08-SEP-21       YES              155          512
         19       3217423      3220958 08-SEP-21       YES             9130          512
         20       3220958      3227089 08-SEP-21       YES             3442          512
         21       3227089      3227609 08-SEP-21       YES              278          512
         22       3227609      3227647 08-SEP-21       YES               19          512

9 rows selected.

testdb> select name from v$datafile where status='OFFLINE';

no rows selected
```

```
testdb> SELECT SYSDATE, APPLIED_TIME FROM V$LOGSTDBY_PROGRESS;

SYSDATE     APPLIED_TIME
------------------ ------------------
08-SEP-21   08-SEP-21

testdb> ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';

Session altered.

testdb> SELECT SYSDATE, APPLIED_TIME FROM V$LOGSTDBY_PROGRESS;

SYSDATE                  APPLIED_TIME
---------------------------- ----------------------------
08-SEP-2021 07:05:56             08-SEP-2021 07:05:55
```

Physru2 - Downtime - SWITCHOVER PRIMARY DATABASE TO LOGICAL STANDBY

The "physru" script automatically resumes from the point it left off, and during the second execution, the script performs the following tasks:

If the sync is less than 30 sec then only script will proceed for switchover and it will ask for confirmation before doing the switchover

- ➔ 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 and Shuts down the new physical standby database.

--- we execute the same command again from primary server ---

physru_v3.sh sys testdb_Primaryhost testdb_standbyhost testdb_Primaryhost testdb_standbyhost 19.0.0.0.0

---On primary ---

```
physru_v3.sh sys testdb_Primaryhost testdb_standbyhost testdb_Primaryhost testdb_standbyhost
19.0.0.0.0
```

```
### Initialize script to either start over or resume execution
Sep 09 10:35:59 2021 [0-1] Identifying rdbms software version
Sep 09 10:36:00 2021 [0-1] database testdb_Primaryhost is at version 12.2.0.1.0
Sep 09 10:36:00 2021 [0-1] database testdb_standbyhost is at version 19.0.0.0.0
Sep 09 10:36:01 2021 [0-1] verifying fast recovery area is enabled at testdb_Primaryhost and
testdb_standbyhost
Sep 09 10:36:01 2021 [0-1] verifying backup location at testdb_Primaryhost and
testdb_standbyhost
Sep 09 10:36:01 2021 [0-1] verifying available flashback restore points
Sep 09 10:36:01 2021 [0-1] verifying DG Broker is disabled
Sep 09 10:36:01 2021 [0-1] looking up prior execution history
Sep 09 10:36:02 2021 [0-1] last completed stage [3-1] using script version 0003

WARN: The last execution of this script either exited in error or at the
      user's request.  At this point, there are three available options:

        1) resume the rolling upgrade where the last execution left off
        2) restart the script from scratch
        3) exit the script

      Option (2) assumes the user has restored the primary and physical
      standby back to the original configuration as required by this script.

Enter your selection (1/2/3): 1

Sep 09 10:36:06 2021 [0-1] resuming execution of script

### Stage 4: Switch the transient logical standby to be the new primary
Sep 09 10:36:07 2021 [4-1] waiting for testdb_standbyhost to catch up (this could take a
while)
Sep 09 10:36:07 2021 [4-1] waiting for apply lag to fall under 30 seconds
Sep 09 10:36:11 2021 [4-1] apply lag measured at 4 seconds
Sep 09 10:36:12 2021 [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 testdb_standbyhost will become the new primary database, and database
testdb_Primaryhost
      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

Sep 09 10:36:42 2021 [4-2] continuing
Sep 09 10:36:42 2021 [4-2] switching testdb_Primaryhost to become a logical standby
Sep 09 10:36:47 2021 [4-2] testdb_Primaryhost is now a logical standby
Sep 09 10:36:47 2021 [4-2] waiting for standby testdb_standbyhost to process end-of-redo from
primary
Sep 09 10:36:49 2021 [4-2] switching testdb_standbyhost to become the new primary
Sep 09 10:36:49 2021 [4-2] testdb_standbyhost is now the new primary

### Stage 5: Flashback former primary to pre-upgrade restore point and convert to physical
Sep 09 10:36:50 2021 [5-1] shutting down database testdb_Primaryhost
Sep 09 10:37:15 2021 [5-1] mounting database testdb_Primaryhost
Sep 09 10:37:31 2021 [5-2] flashing back database testdb_Primaryhost to restore point
PRU_0000_0003
Sep 09 10:37:34 2021 [5-3] converting testdb_Primaryhost into physical standby
Sep 09 10:37:34 2021 [5-4] shutting down database testdb_Primaryhost

NOTE: Database testdb_Primaryhost 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.
```

################## DOWN TIME ENDS ###################

Sync/upgrade OLD/actual PRIMARY

Note – we can just mount the standby in 19c home and call the physru script for third time it will start MRP and sync the standby and then does the switchover again (here I have manually started MRP and synched standby before calling the script to have more control)

Update oratab pointing to new home

Copy spfile/passwordfile/make changes to listener/tnsnames pointing to new 19c home

```
testdb> STARTUP NOMOUNT;
ORACLE instance started.

Total System Global Area 1073737792 bytes
Fixed Size                  8904768 bytes
Variable Size             276824064 bytes
Database Buffers    780140544 bytes
Redo Buffers                7868416 bytes
testdb> ALTER DATABASE MOUNT STANDBY DATABASE;

Database altered.
```

Start MRP and sync

```
testdb> select process,status,sequence#,thread# from v$managed_standby where process like
'MRP%';

no rows selected

testdb> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;
Database altered.

testdb> select process,status,sequence#,thread# from v$managed_standby where process like
'MRP%';

no rows selected
```

MRP failed to start with unpublished bug workaround is to set the parameter - Doc ID 2069325.1

```
testdb> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE cancel;

Database altered.
```

```
testdb> select process,status,sequence#,thread# from v$managed_standby where process like
'MRP%';

no rows selected

testdb> alter system set "_transient_logical_clear_hold_mrp_bit"=TRUE SCOPE=BOTH;

System altered.

testdb> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;

Database altered.

testdb> select process,status,sequence#,thread# from v$managed_standby where process like
'MRP%';

PROCESS   STATUS        SEQUENCE#    THREAD#
--------- ------------ ---------- ----------
MRP0      WAIT_FOR_LOG         27          1
```

We Need to set the archive dest2 on the new primary to ship the archives to this standby database

--- on current primary database---

```
ALTER SYSTEM SET log_archive_dest_2='service="testdb_Primaryhost" SYNC AFFIRM delay=0 optional
compression=disable max_failure=0 max_connections=1 reopen=300
db_unique_name="testdb_Primaryhost" net_timeout=10 valid_for=(online_logfile,all_roles)'
scope=both sid='*';
```

```
testdb> alter system set log_archive_dest_2='service="testdb_Primaryhost" SYNC AFFIRM delay=0
optional compression=disable max_failure=0 max_connections=1 reopen=300
db_unique_name="testdb_Primaryhost" net_timeout=10 valid_for=(online_logfile,all_roles)'
scope=both sid='*';

System altered.

testdb> select DEST_ID,DEST_NAME,DESTINATION,TARGET,STATUS,ERROR from v$archive_dest where
dest_id=2;

   DEST_ID
----------
DEST_NAME
--------------------------------------------------------------------------------
--------------------------------------
DESTINATION
--------------------------------------------------------------------------------
--------------------------------------
TARGET              STATUS    ERROR
---------------- --------- ----------------------------------------------------------------
           2
LOG_ARCHIVE_DEST_2
testdb_Primaryhost
STANDBY   VALID
```

DEST status should be valid if it report any ERROR like TNS fix the issue

---- Verify the sync status ------

--- on current primary ----

```
testdb> SELECT ARCH.THREAD# "Thread", ARCH.SEQUENCE# "Last Sequence Received", APPL.SEQUENCE#
"Last Sequence Applied", (ARCH.SEQUENCE# - APPL.SEQUENCE#) "Difference" FROM (SELECT THREAD#
,SEQUENCE# FROM V$ARCHIVED_LOG WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME)
FROM V$ARCHIVED_LOG GROUP BY THREAD#)) ARCH,(SELECT THREAD# ,SEQUENCE# FROM V$LOG_HISTORY
WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME) FROM V$LOG_HISTORY GROUP BY
THREAD#)) APPL WHERE ARCH.THREAD# = APPL.THREAD# ORDER BY 1;

    Thread Last Sequence Received Last Sequence Applied Difference
---------- ---------------------- ---------------------- ----------
         1                     56                     56          0
         1                     56                     56          0
```

---- on standby -----

```
testdb> SELECT ARCH.THREAD# "Thread", ARCH.SEQUENCE# "Last Sequence Received", APPL.SEQUENCE#
"Last Sequence Applied", (ARCH.SEQUENCE# - APPL.SEQUENCE#) "Difference" FROM (SELECT THREAD#
,SEQUENCE# FROM V$ARCHIVED_LOG WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME)
FROM V$ARCHIVED_LOG GROUP BY THREAD#)) ARCH,(SELECT THREAD# ,SEQUENCE# FROM V$LOG_HISTORY
WHERE (THREAD#,FIRST_TIME ) IN (SELECT THREAD#,MAX(FIRST_TIME) FROM V$LOG_HISTORY GROUP BY
THREAD#)) APPL WHERE ARCH.THREAD# = APPL.THREAD# ORDER BY 1;

    Thread Last Sequence Received Last Sequence Applied Difference
---------- ---------------------- ---------------------- ----------
         1                     56                     56          0
```

Physru3 - DOWNTIME 2 – (Optional) Put the Primary back in place.

SWITCHOVER NEW PRIMARY TO PHYSICAL STANDBY

Check the sync once again before doing the switchover and call the script for the third time, it will ask you for confirmation before doing the switchover.

Script lost the connection after it has converted the primary to standby, I have just restarted and it completed successfully. Script report shows second switchover time as 9 mins due to timeout nut actually it completed the switchover in 2 mins.

If we are not doing the switchback then just configure the dataguard broker

```
[oracle@Primaryhost] /u01/bin/oracle/upg1: testdb_Primaryhost testdb_standbyhost 19.0.0.0.0
<
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Sep 09 11:35:07 2021 [0-1] Identifying rdbms software version
Sep 09 11:35:07 2021 [0-1] database testdb_Primaryhost is at version 19.0.0.0.0
Sep 09 11:35:07 2021 [0-1] database testdb_standbyhost is at version 19.0.0.0.0
```

```
Sep 09 11:35:08 2021 [0-1] verifying fast recovery area is enabled at testdb_Primaryhost and
testdb_standbyhost
Sep 09 11:35:08 2021 [0-1] verifying backup location at testdb_Primaryhost and
testdb_standbyhost
Sep 09 11:35:08 2021 [0-1] verifying available flashback restore points
Sep 09 11:35:09 2021 [0-1] verifying DG Broker is disabled
Sep 09 11:35:09 2021 [0-1] looking up prior execution history
Sep 09 11:35:09 2021 [0-1] last completed stage [7-1] using script version 0003

WARN: The last execution of this script either exited in error or at the
      user's request.  At this point, there are three available options:

        1) resume the rolling upgrade where the last execution left off
        2) restart the script from scratch
        3) exit the script

      Option (2) assumes the user has restored the primary and physical
      standby back to the original configuration as required by this script.

Enter your selection (1/2/3): 1

Sep 09 11:35:12 2021 [0-1] resuming execution of script
Sep 09 11:35:13 2021 [7-2] waiting for apply lag to fall under 30 seconds
Sep 09 11:35:13 2021 [7-2] apply lag measured at 0 seconds
Sep 09 11:35:14 2021 [7-3] switching testdb_standbyhost to become a physical standby

### The following error was encountered:
ERROR:
ORA-01034: ORACLE not available
Process ID: 8948
Session ID: 286 Serial number: 34957

### The offending sql code in its entirety:
set pagesize 0 feedback off verify off heading off echo off tab off
whenever sqlerror exit sql.sqlcode
alter database commit to switchover to physical standby with session shutdown;
exit;

Sep 09 11:35:18 2021 [7-3] ERROR: failed to switchover to physical standby
[oracle@Primaryhost] /u01/bin/oracle/upg1: physru_v3.sh sys testdb_Primaryhost
testdb_standbyhost testdb_Primaryhost testdb_standbyhost 1>
Please enter the sysdba password:

### Initialize script to either start over or resume execution
Sep 09 11:36:46 2021 [0-1] Identifying rdbms software version
Sep 09 11:36:46 2021 [0-1] database testdb_Primaryhost is at version 19.0.0.0.0
Sep 09 11:36:47 2021 [0-1] database testdb_standbyhost is at version 19.0.0.0.0
Sep 09 11:36:47 2021 [0-1] verifying fast recovery area is enabled at testdb_Primaryhost and
testdb_standbyhost
Sep 09 11:36:47 2021 [0-1] verifying backup location at testdb_Primaryhost and
testdb_standbyhost
Sep 09 11:36:47 2021 [0-1] verifying available flashback restore points
Sep 09 11:36:48 2021 [0-1] verifying DG Broker is disabled
Sep 09 11:36:48 2021 [0-1] looking up prior execution history
Sep 09 11:36:48 2021 [0-1] last completed stage [7-2] using script version 0003

WARN: The last execution of this script either exited in error or at the
      user's request.  At this point, there are three available options:

        1) resume the rolling upgrade where the last execution left off
        2) restart the script from scratch
```

```
        3) exit the script

      Option (2) assumes the user has restored the primary and physical
      standby back to the original configuration as required by this script.

Enter your selection (1/2/3): 1

Sep 09 11:36:50 2021 [0-1] resuming execution of script
Sep 09 11:36:51 2021 [7-3] testdb_standbyhost is already a physical standby
Sep 09 11:36:51 2021 [7-3] shutting down database testdb_standbyhost
Sep 09 11:37:18 2021 [7-3] mounting database testdb_standbyhost
Sep 09 11:37:32 2021 [7-3] starting media recovery on testdb_standbyhost
Sep 09 11:37:39 2021 [7-3] confirming media recovery is running
Sep 09 11:37:39 2021 [7-3] waiting for standby testdb_Primaryhost to process end-of-redo from
primary
Sep 09 11:37:39 2021 [7-3] switching testdb_Primaryhost to become the new primary
Sep 09 11:37:41 2021 [7-3] testdb_Primaryhost is now the new primary
Sep 09 11:37:41 2021 [7-3] opening database testdb_Primaryhost
```

```
### Stage 8: Statistics
script start time:                                  09-Sep-21 07:59:57
script finish time:                                 09-Sep-21 11:37:47
total script execution time:                              +00 03:37:50
wait time for user upgrade:                               +00 01:30:36
active script execution time:                             +00 02:07:14
transient logical creation start time:              09-Sep-21 08:00:01
transient logical creation finish time:             09-Sep-21 08:06:27
primary to logical switchover start time:           09-Sep-21 10:36:12
logical to primary switchover finish time:          09-Sep-21 10:36:50
primary services offline for:                             +00 00:00:38
total time former primary in physical role:               +00 00:50:16
time to reach upgrade redo:
time to recover upgrade redo:                             +00 00:14:02
primary to physical switchover start time:          09-Sep-21 11:28:15
physical to primary switchover finish time:         09-Sep-21 11:37:46
primary services offline for:                             +00 00:09:31

SUCCESS: The physical rolling upgrade is complete
```

----- set DG Broker parameter and enable configuration -----

```
testdb> alter system set dg_broker_start=true scope=both sid='*';

System altered.
```

```
DGMGRL> enable configuration;
Enabled.
DGMGRL> show configuration;

Configuration - testdb_dg

  Protection Mode: MaxPerformance
  Members:
  testdb_Primaryhost - Primary database
    testdb_standbyhost - Physical standby database
```

```
Fast-Start Failover:  Disabled

Configuration Status:
WARNING   (status updated 24 seconds ago)

DGMGRL> show database testdb_standbyhost

Database - testdb_standbyhost

  Role:               PHYSICAL STANDBY
  Intended State:     APPLY-ON
  Transport Lag:      0 seconds (computed 1 second ago)
  Apply Lag:          0 seconds (computed 1 second ago)
  Average Apply Rate: 632.00 KByte/s
  Real Time Query:    OFF
  Instance(s):
    testdb

Database Status:
SUCCESS
```

-------- UPGRADE DONE ----------