Alliance/KEC Reporting Database

Version: 1.0

Table of Contents

1. Document Control

The document control section describes the revision history and summary of the changes made to the document.

1.1. Document Location

The source of the document will be found under the KEC ALLIANCE Documentation section.

1.2. Revision History

Revision Number	Revision Date	Summary of Changes Made	Changed By
1.	01 Dec 2016	Initial Draft 1.0	Ramiz Sarah
2.			
3.			
4.			

2. Executive Summary

The purpose of this document is to provide high level technical description and best practices for setting up a reporting database utilizing Oracle Streams (Downstream Capture) that will facilitate data replication for the KEC/Alliance applications. This can be used to offload reporting to another database server minimizing the production database(s) work load.

3. Data Replication

The Data Replication section provides a detailed description of the Oracle Replication process.

3.1. Introduction

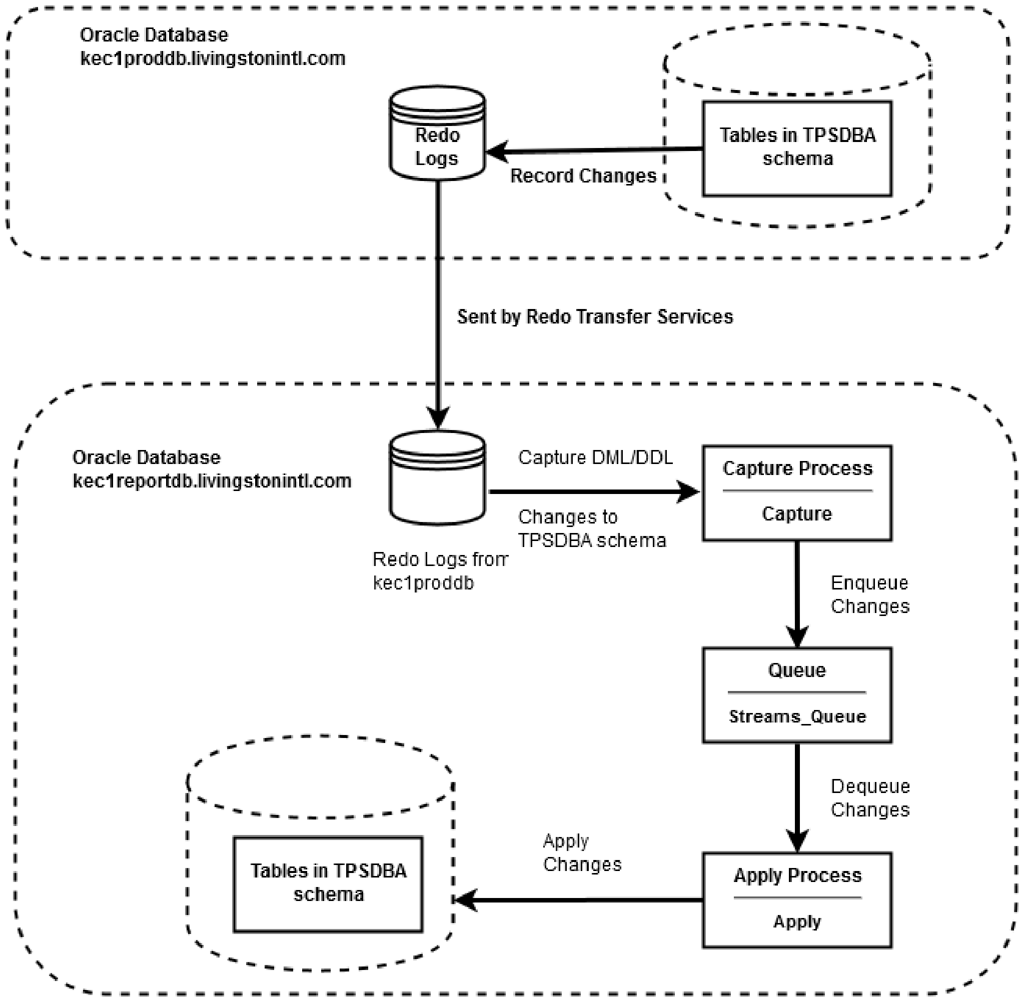
Oracle Streams enables the propagation and management of data, transactions, and events in a data stream either within a database, or from one database to another. Oracle Streams consists of three components: capture, propagation, and apply.

Changes made to the source database objects are captured by the “capture” process, which formats the changes into Logical Change Records (LCRs) that can be propagated to destination databases and then applied by the Oracle Streams “apply” process.

There are a number of Streams implementations/configurations that can be achieved depending on the replication requirements (environment, number of target sites, data volume, the number of transactions occurring on the source database ...etc).

The following table lists some Streams configurations and their practical usage.

Configuration Name	**Usage**
Hub-and-Spoke Replication	Used to distribute information to multiple target databases and to consolidate information from multiple databases to a single database. A central database (hub) communicates with one or more secondary databases (spokes). The spokes do not communicate directly with each other.
Downstream Capture	Used to replicate a full/partial database, or to offload processing from the source production database to the target database The capture process runs on a remote database instead of the source database. Using downstream capture removes the capture workload from the production database.
Synchronous Capture	Used to replicate changes to tables with infrequent data changes in a highly active database or in situations where capturing changes from the redo logs is not possible.
N-Way Replication	Used in an environment with several peer databases and each database must replicate data with each of the other databases. Each database communicates directly with each other database in the environment. The changes made to replicated database objects at one database are captured and sent directly to each of the other databases in the environment, where they are applied.



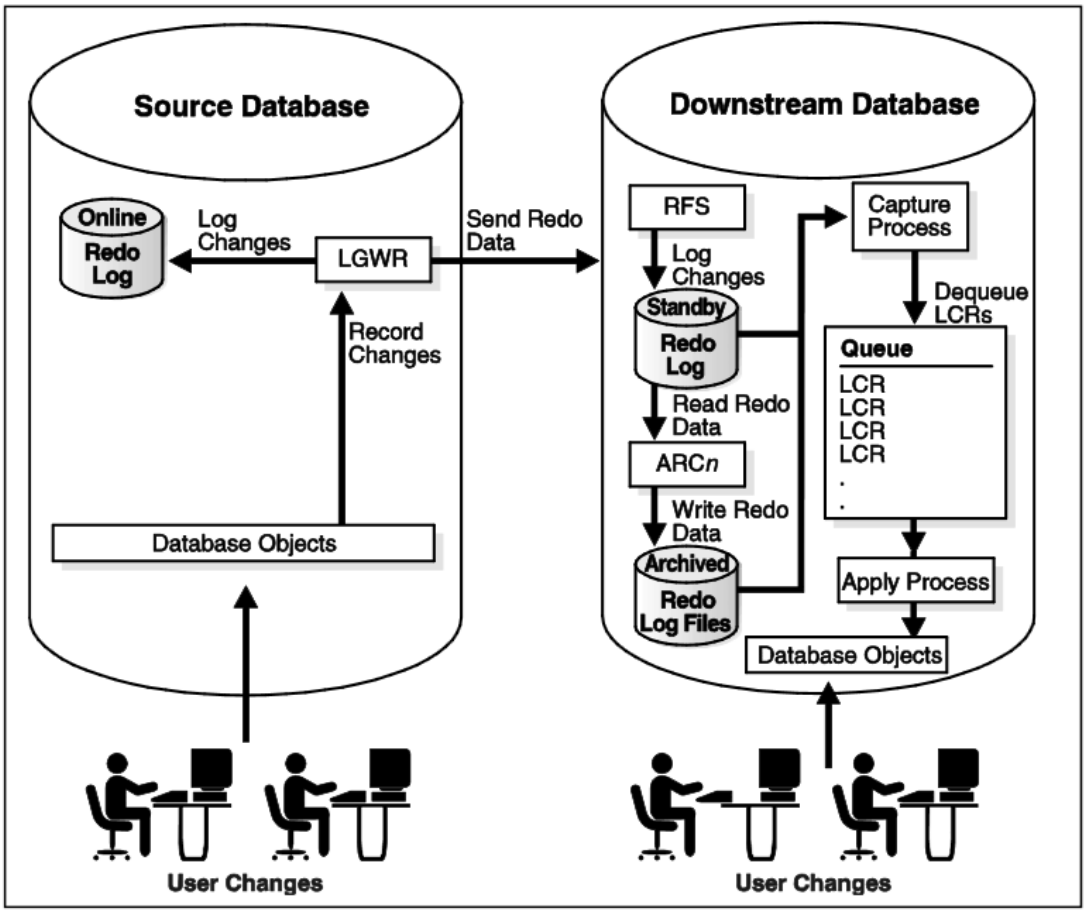
3.2. Oracle Streams (Downstream Capture)

Downstream capture is a configuration in which the capture process runs on a database other than the source database. Using downstream capture removes the capture workload from the production database to the target (downstream) database.

The downstream capture is typically used to replicate a full database, to offload processing from the source database to the target database, or to reduce data loss with ASYNC or SYNC redo transport.

There are a number of advantages to using downstream capture configurations

- Capturing changes uses fewer resources at the source (production) database because the downstream database performs most of the required work.
- The ability to configure multiple capture processes that capture changes from a single source database provides more flexibility and can improve scalability.
- Providing additional protection against data loss as the redo logs data is shipped to a downstream database.



There are two possible configurations to downstream capture; these are: "Real-Time Capture" and "Archived-Log Capture".

The "Real-Time" downstream capture process captures changes from the standby redo log whenever possible and from the archived standby redo log files whenever necessary. A capture process can capture changes in the archived standby redo log files if it falls behind. When it catches up, it resumes capturing changes from the standby redo log.

The advantage of "Real-Time" downstream capture over "Archived-Log" downstream capture is that "Real-Time" downstream capture reduces the amount of time required to capture changes made at the source database. The time is reduced because the "Real-Time" downstream capture process does not need to wait for the redo log file to be archived before it can capture data from it.

3.3. Operational Requirements

The following are operational requirements for using downstream capture:

- The source (production) database must be running at least Oracle Database 10g and the target (downstream capture) database must be running the same version of Oracle as the source database or higher.
- The operating system on the source and downstream capture sites must be the same, but the operating system release does not need to be the same. In addition, the downstream sites can use a different directory structure than the source site.

- The hardware architecture on the source and downstream capture sites must be the same. For example, a downstream capture configuration with a source database on a 32-bit Sun system must have a downstream database that is configured on a 32-bit Sun system. Other hardware elements, such as the number of CPUs, memory size, and storage configuration, can be different between the source and downstream sites.

3.4. Downstream Capture Implementation

This section describes how to prepare the source database to ship redo to a downstream database, and to prepare the target database to receive and apply the received redo.

3.4.1. Prepare the Redo-Logs (Source and Target)

Configure the source and target databases in ARCHIVELOG mode.

The best practice for one-way Streams capture, is to configure both the source and target databases to run in ARCHIVELOG mode, in case media recovery is required for either of the databases.

Also, configure the local archive destination, **LOG_ARCHIVE_DEST_1**, parameter and do not use a flash recovery area. Streams downstream capture require online redo logs and archived redo logs. Do not place archived redo log files in the Flash Recovery area, because this is a fixed-size storage area and the files may be deleted if the amount of space becomes low.

3.4.2. Create Streams Tablespace (Source and Target)

Create a dedicated Streams tablespace to contain the Streams AQ tables and other objects related to Streams, and to allow for better space management for the Streams objects.

For downstream capture, we will create the Streams tablespace on the downstream capture database only.

```
CREATE TABLESPACE "STREAMS_TBS"
DATAFILE '/path/to/datafile/streams_01.dbf' SIZE 100m
AUTOEXTEND ON NEXT 10m MAXSIZE 8g
LOGGING ONLINE PERMANENT BLOCKSIZE 8192
EXTENT MANAGEMENT LOCAL AUTOALLOCATE DEFAULT
NOCOMPRESS SEGMENT SPACE MANAGEMENT AUTO;
```

3.4.3. Setup Log Miner (Target)

Instruct the Log Miner service to use the newly created tablespace:

```
exec DBMS_LOGMNR_D.SET_TABLESPACE ('STREAMS_TBS');
```

3.4.4. Create Streams Admin User (Source and Target)

Create the Streams administrator account that will be used to create and modify all Streams configurations and perform administrator functions.

For example, to create a new user named “strmadmin” with the default “STREAMS_TS” tablespace, enter the following SQL*Plus statement (as SYSDBA):

```
CREATE USER strmadmin IDENTIFIED BY strmadmin
DEFAULT TABLESPACE streams_tbs temporary tablespace temp
QUOTA UNLIMITED ON streams_tbs;
```

Grant all the necessary privileges:

```
GRANT DBA TO strmadmin;

BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE (
    grantee          => 'strmadmin',
    grant_privileges => TRUE);
END;
/
```

3.4.5. Create Database Link (Target Database)

Create the dedicated database link to link up the target to the source database. We need to create two separate database links, a global/public link and a private link under the “strmadmin” schema to avoid getting connection errors due to using GLOBAL_NAME.

The database link name has to be set to the database GLOBAL_NAME. You can use the following SQL to determine the database GLOBAL_NAME:

```
Select * from global_name;

Create public database link KEC1RPTDB.LIVINGSTONINTL.COM using
'KEC1DEVD.LIVINGSTONINTL.COM';

Conn STRMADMIN/STRMADMIN

Create database link KEC1RPTDB.LIVINGSTONINTL.COM connect to
strmadmin identified by "strmadmin";
```

Validate the database link is working by issuing the following query on the target database:

```
Select * from dual@kec1proddb.livingstonintl.com;
```

3.4.6. Initialization Parameters (Source and Target)

Set the following parameters on both the source and target databases:

a. Set **GLOBAL_NAMES** to TRUE on both sides:

   ```
   Alter system set global_names=TRUE scope=BOTH;
   ```

b. Set **UNDO_RETENTION** to a value of at least 3600, this setting **ONLY** applies to the target database running the capture process:

   ```
   alter system set undo_retention=3600 scope=both;
   ```

c. Set **LOG_ARCHIVE_CONFIG** to enable the database to send/receive redo. The DG_CONFIG attribute in this initialization parameter includes the DB_UNIQUE_NAME of the source and downstream databases.

```
ALTER SYSTEM SET
LOG_ARCHIVE_CONFIG='DG_CONFIG=(kec1proddb,kec1rptdb)'
SCOPE=SPFILE;
```

d. Do not set the **AQ_TM_PROCESSES** parameter to 0 or 10. Doing so could disable the queue monitoring processing and impact the Streams pool memory utilization. If this parameter has not been set, then the parameter will be auto-tuned by Oracle.

e. Set **TIMED_STATISTICS** to TRUE on each Streams database. This setting will allow performance statistics to be gathered for analysis of potential performance bottlenecks.

f. Set **SHARED_POOL_SIZE** to 256MB (min) for the Streams Target database. Streams uses a significant amount of PL/SQL and library cache components especially for the database running the “Apply” process.

g. Set **STREAMS_POOL_SIZE** parameter to a minimum of 256MB. Memory from the Streams pool is used by both the streams processes and the buffered queue, with the LCRs in the buffered queue the largest component. Use the **V$STREAMS_POOL_ADVICE** dynamic view to help size the buffered queue appropriately.

3.4.7. Configure Stand-by Redo-Logs (Target)

To enable real-time mining for capture at the downstream database, we must configure standby redo logs (SRLs) on the downstream (target) database to accept redo from the source database. The Streams capture process mines the standby redo logs and any archived redo logs created from the standby redo logs.

We can use the following dynamic SQL to generate the standby redo log(s):

```
SELECT 'alter database add standby logfile '''
  || regexp_substr(MEMBER,'/.+/')
  ||'stdby_'
  || regexp_replace(member,regexp_substr(MEMBER,'/.+/'),'')
  || ''' size '
  ||bytes
  ||';' "Create Standby redo"
FROM v$logfile lf ,
  v$log l
WHERE l.group# = lf.group#
UNION ALL
SELECT 'alter database add standby logfile '''
  || regexp_substr(MEMBER,'/.+/')
  ||'stdby_redo0'
  ||
  (SELECT MAX(group#)+1 FROM v$log
  )
  ||'.rdo'
  || ''' size '
  ||bytes
  ||';' "Create Standby redo"
```

```
FROM v$logfile lf ,
  v$log l
WHERE l.group# = lf.group#
AND rownum <=2
```

Query the V$STANDBY_LOG view to validate the log groups and their status:

```
SELECT GROUP#, SEQUENCE#, STATUS FROM V$STANDBY_LOG;
```

As redo data is received by the downstream database, the redo is written to disk. In synchronous configurations, the disk write I/O must occur prior to sending acknowledgment back to the primary database that the redo has been received. Therefore it is important to optimize I/O on the standby database.

To improve I/O performance on the standby database, employ the following best practices:

- Ensure that Oracle is able to use ASYNC I/O. Note that (by default) the Oracle database is configured for asynchronous I/O. However, the operating system, and storage array must also be configured properly.

- The number of standby redo log file groups must be at least one more than the number of online redo log file groups on the source database.

- The stand-by redo log file size must be exactly the same as the source database file size.

- Stand-by redo logs should not be multiplexed.

- Avoid using RAID5 configuration for Stand-by redo logs as I/O writes are slower than when the controllers are configured with mirroring.

3.4.8. Setup Redo Archiving (Source)

The primary/source/production database is configured so that the transport services use the log writer process (LGWR) at the source database to send redo data to the downstream database (either synchronously or asynchronously). At the same time, the LGWR records redo data in the online redo log at the source database.

Enable Shipping of online redo log data from Source to Downstream database:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2='SERVICE=kec1rptdb LGWR ASYNC
NOREGISTER
VALID_FOR=(ONLINE_LOGFILES,PRIMARY_ROLE)
DB_UNIQUE_NAME=kec2rptdb'
SCOPE=SPFILE;
```

SERVICE - Specify the identifier of the downstream database (from TNSNAMES.ora entry.

LGWR ASYNC or **LGWR SYNC** - Specify a redo transport mode.
The advantage of specifying LGWR SYNC is that redo data is sent to the downstream database faster than when LGWR ASYNC is specified. Having said that, using **LGWR SYNC** can slow down the source database commit as it has to write across the network.

Use **LGWR ASYNC** redo transport mode so that the log writer process writes to the remote (downstream) database asynchronously.

NOREGISTER - Specify this attribute so that the location of the archived redo log files is not recorded in the downstream database control file.

VALID_FOR - Specify either (ONLINE_LOGFILE, PRIMARY_ROLE) or (ONLINE_LOGFILE, ALL_ROLES).

DB_UNIQUE_NAME - The value of DB_UNIQUE_NAME of the downstream database.

Set the archive logs format for the source database:

```
ALTER SYSTEM SET log_archive_format = 'kec1proddb_%t_%s_%r.arc'
SCOPE=SPFILE;
```

Finally, enable both archive destinations:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2= ENABLE SCOPE=SPFILE;

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = ENABLE SCOPE=SPFILE;
```

3.4.9. Setup Redo Archiving (Target)

Configure the downstream database to receive redo data from the source database:

```
Alter system set LOG_ARCHIVE_DEST_2='LOCATION=/kec1fra/
kec1rptdb/archivelog/kec1prod_archive
VALID_FOR=(STANDBY_LOGFILE,PRIMARY_ROLE)' scope=both;
```

LOCATION – Directory where archived-logs will be written to coming from the source database.
VALID FOR - Specify either (STANDBY_LOGFILE, PRIMARY_ROLE) or (STANDBY_LOGFILE, ALL_ROLES).

Also, set the local archiving destination for the target database:

```
Alter system set
LOG_ARCHIVE_DEST_1='LOCATION=/kec1fra/kec1rptdb/archivelog
VALID_FOR=(ONLINE_LOGFILE,PRIMARY_ROLE)' scope=both;
```

Please note that a real-time downstream capture configuration should keep archived standby redo log files separate from archived online redo log files generated by the downstream database. Specify ONLINE_LOGFILE instead of ALL_LOGFILES for the redo log type in the VALID_FOR attribute to accomplish this.

Finally, enable both archive destinations:

```
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2= ENABLE SCOPE=SPFILE;

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = ENABLE SCOPE=SPFILE;
```

3.4.10. Setup the Capture/Apply Queue (Target)

Run the SET_UP_QUEUE procedure to create the queue that will be used by both the capture and apply processes:

```
BEGIN
  DBMS_STREAMS_ADM.SET_UP_QUEUE(
    queue_table => 'strmadmin.streams_queue_tab',
    queue_name  => 'strmadmin.streams_queue',
    queue_user  => 'STRMADMIN');
END;
/
```

Using a single queue provides the best efficiency (and practice) for Streams where both capture and apply are configured on the same database as it eliminates the redundant propagation (queue to queue) transfer.

Check the queue through the following SQL:

```
select name, queue_table from dba_queues;
```

3.4.11. Setup the Capture Process (Target)

Run the CREATE_CAPTURE procedure to create the capture process:

```
BEGIN
  DBMS_CAPTURE_ADM.CREATE_CAPTURE(
    queue_name         => 'strmadmin.streams_queue',
    capture_name       => 'real_time_capture',
    rule_set_name      => NULL,
    start_scn          => NULL,
    source_database    => 'KEC1PRODDB.LIVINGSTONINTL.COM',
    use_database_link  => TRUE,
    first_scn          => NULL,
    logfile_assignment => 'implicit'
  );
END;
/
```

The *logfile_assignment* parameter is relevant for Down Streams capture configuration. If set to IMPLICIT (default value) then, the Capture process at the Downstream database scans all redo log files added by log transport services or manually from the source database to the Down Streams database.

Use the following SQL to check the capture information:

```
SELECT capture_name, status from dba_capture;

SELECT parameter, value, set_by_user FROM DBA_CAPTURE_PARAMETERS;
```

3.4.12. Setup the Apply Process (Target)

Run the CREATE_APPLY procedure to create the apply process at the downstream site

```
BEGIN
   DBMS_APPLY_ADM.CREATE_APPLY(
      queue_name => 'strmadmin.streams_queue',
      apply_name => 'real_time_apply',
      apply_captured => TRUE
   );
END;
/
```

Use the following SQL to check the capture information:

```
SELECT apply_name, status, queue_name FROM DBA_APPLY;

SELECT parameter, value, set_by_user
FROM DBA_APPLY_PARAMETERS
WHERE apply_name = 'REAL_TIME_APPLY';
```

Some “Apply” process parameters to consider:

Parameter	**Default Value**	**Recommended Value**	**Comment**
DISABLE_ON_ERROR	Y	N	If Y, then the apply process is disabled on the first unresolved error, even if the error is not fatal. If N, then the apply process continues regardless of unresolved errors.
PARALLELISM	1	4	Parallelism, configures the number of apply servers available to the apply process for performing user transactions. This value can be increased by multiples of 4 as needed.
TXN_LCR_SPILL_THRESHOLD	10,000	10,000	Enables you to specify that an apply process begins to spill messages for a transaction from memory to disk when the number of messages in memory for a particular transaction exceeds the specified number. Setting this parameter to a value that is higher than the default to try to stage everything in memory must be done carefully so that queue spilling is not increased
_DYNAMIC_STMTS	N	Y	If set to "Y", then for UPDATE statements, the apply process will optimize the generation of SQL statements based on required columns.
_HASH_TABLE_SIZE	80*parallelism	1000000	Set the size of the hash table used to calculate transaction dependencies to 1 million.

3.4.13. Create the Capture Rule Set (Target)

Create the positive rule set for the capture process and add a rule to it:

```
BEGIN
   DBMS_STREAMS_ADM.ADD_SCHEMA_RULES(
      schema_name         =>'TPSDBA',
      streams_type        =>'CAPTURE',
      streams_name        =>'REAL_TIME_CAPTURE',
      queue_name          =>'STRMADMIN.STREAMS_QUEUE',
      include_dml         =>TRUE,
      include_ddl         =>TRUE,
      source_database     =>'KEC1PRODDB.LIVINGSTONINTL.COM'
   );
```

```
END;
/
```

Check the created rule using the following SQL:

```
SELECT rule_name, rule_condition
FROM DBA_STREAMS_SCHEMA_RULES
WHERE streams_name = 'REAL_TIME_CAPTURE'
AND streams_type = 'CAPTURE';
```

Some “Capture” process parameters to consider:

Parameter	Default Value	Recommended Value	Comment
PARALLELISM	1	1	Number of parallel execution servers for the capture process.
_CHECKPOINT_FREQUENCY	500	1000	Modify the frequency of log miner checkpoints especially in a database with significant LOB or DDL activity. Log miner checkpoints are not the same as database checkpoints. Availability of log miner checkpoints impacts the time required to recover/restart the capture process after database restart. A log miner checkpoint is requested by default every 10Mb of redo mined. If the value is set to 500, a log miner checkpoint is requested after every 500Mb of redo mined. Increasing the value of this parameter is recommended for active databases with significant redo generated per hour.
_SGA_SIZE	10	10	Amount of memory available from the streams pool for log miner processing. The default amount of streams_pool memory allocated to log miner is 10Mb. Only increase this value when the ORA-1341 error is encountered (where large LOBs are processed.)

3.4.14. Set Real-Time Capture (Target)

Set the downstream REAL_TIME_MINE capture process parameter to Y. This procedure will configure the Streams capture process to perform real-time mining of the redo log that is shipped from the source database.

```
BEGIN
  DBMS_CAPTURE_ADM.SET_PARAMETER(
    capture_name => 'REAL_TIME_CAPTURE',
    parameter    => 'DOWNSTREAM_REAL_TIME_MINE',
    value        => 'Y');
END;
/
```

Once the parameter has been set for the capture process, you can monitor the ALERT.LOG on the downstream database to watch Log Miner mining the standby redo log files.

Go to the source database and archive the current logfile (as SYSDBA). This will start the real time mining of the source database redo log.

```
ALTER SYSTEM ARCHIVE LOG CURRENT;
```

Check that the status of the stand-by redo logs on the target database has changed from UNASSIGNED to ACTIVE:

```
SELECT GROUP#, THREAD#, SEQUENCE#, ARCHIVED, STATUS FROM
V$STANDBY_LOG;
```

3.4.15. Start the Apply and Capture Processes (Target)

Connect to the target instance as the Streams admin user and startup the Apply and Capture processes:

```
conn strmadmin/strmadmin

exec DBMS_APPLY_ADM.START_APPLY(apply_name => 'REAL_TIME_APPLY');
```

Check The Apply process status using the following SQL:

```
select apply_name, status from dba_apply;

APPLY_NAME                     STATUS
------------------------------ --------
REAL_TIME_APPLY                ENABLED

exec DBMS_CAPTURE_ADM.START_CAPTURE(capture_name =>
'REAL_TIME_CAPTURE');
```

Check The Capture process status using the following SQL:

```
select capture_name, status from dba_capture;

CAPTURE_NAME                   STATUS
------------------------------ --------
REAL_TIME_CAPTURE              ENABLED
```

3.4.16. Set Checkpoint Retention Time Parameter (Target)

Set the CHECKPOINT_RETENTION_TIME capture parameter to specify the number of days of checkpoints the capture process will retain. The default value for this parameter is 60 days but the recommended initial setting is 7 days.

Changing this parameter to 7 days can reduce the volume of checkpoint information that the capture process purges, and therefore improves the overall performance of the capture process.

```
BEGIN
   DBMS_CAPTURE_ADM.ALTER_CAPTURE (
      capture_name => 'REAL_TIME_CAPTURE',
      checkpoint_retention_time => 7);
END;
/
```

3.4.17. Instantiating the replicated objects (Target)

There are a number of ways to instantiate the schema objects for a Streams environment. We can use exp/imp (with datapump) to move the initial data across, or we can utilize the current running data guard setup.

3.4.18. Set Apply Parallelism (Target)

The “Apply Parallelism” parameter specifies the number of transactions that can be applied by Oracle Streams simultaneously. As a starting point, set the degree of parallelism for the apply process to “4”.

```
BEGIN
   DBMS_APPLY_ADM.SET_PARAMETER
('REAL_TIME_APPLY','PARALLELISM','4');
END;
/
```

3.5. Supported Data Types

The data manipulation language (DML) and data types supported by the "Capture" process do not always match the DML types and data types supported by the Streams "Apply" process.

Capture process captures the following types of DML changes made to tables: INSERT, UPDATE, DELETE, MERGE and Piecewise updates to LOBs.

A capture process cannot capture DML changes made to the following:

- Temporary tables or object tables.
- Tables in the flashback data archive.
- External tables.

We can run the following SQL query to identify objects that are not compatible with capture process:

```
Select owner, table_name from dba_streams_unsupported where
auto_filtered='YES';
```

Capture process does not capture the results of DML changes to columns of the following data types:

- BFILE
- ROWID
- User-defined types (including object types, REFs, varrays, and nested tables).
- XMLType stored object relationally or as binary XML.
- The following Oracle-supplied types: Any types, URI types, spatial types, and media types.

As for the "Apply" process, it does not handle DML calls, it handles Logical Change Records (LCRs) and/or user-messages, so the restrictions are applicable only at the data type layer.

- BFILE
- ROWID
- User-defined types (including object types, REFs, varrays, and nested tables).
- The following Oracle-supplied types: Any types, URI types, spatial types, and media types.

3.6. Monitoring Streams Performance

Just like any other process, performance can become a concern with Oracle Streams. This section highlights the performance considerations when deploying applications for Streams replication.

Monitoring Streams is a must in any configuration implementation as just like any other process, performance can become a concern. If there are errors in the process, then those errors need to be identified and resolved quickly. If errors are not identified in a timely manner, data inconsistencies can occur in the tables that are replicated.

The Streams Health Check script provides information about the current configuration of the Streams environment. The script itself and instruction(s) on how to run the script is available on My Oracle Support (MOS Note 273674.1).

3.7. Known Issues

There is a known issue with Oracle Streams where change data capture does not capture changes for tables with large objects (LOBs) greater than 3964 bytes. This is confirmed through bug 14135425.

3.8. Licensing

Oracle Streams is included in Oracle Enterprise Edition (EE), hence, no extra licensing is required.

According to Oracle, Streams will continue to be supported in Oracle 12c, but it will not be actively enhanced. Future releases of the Oracle RDBMS software will support Streams' current functionality.

New replication features in relation to the capture of additional data types, etc. will not be supported by Streams and if there is a need to use any of those features then we will have to use (and license) Oracle Golden Gate.

4. Assumptions and Constraints

The following table provides the assumptions made in preparing this document:

#	Relates to	Description
A1	Environment	It is assumed that the Target instance name is “kec1rptdb”.
A2	Schedule	All required business area testers are available to test business/application level reports once the setup is done within the planned timeframe set at the beginning of the project.
A3	Licenses	Oracle RDBMS Enterprise Edition licenses are in place and that we’re not in violation of any Oracle and/or other licenses requirements.
A4	Software	Client Reports changes will be covered by development and/or the application team.
A5	Scope	Only In-Scope items will be performed to ensure a successful Streams solution implementations.
A6	Network	Network changes including firewall ports changes will need to be evaluated and addressed by the network team to ensure sufficient firewall rules are in place to allow appropriate access.
A7	File System	It is assumed that the file system structure, OS version, and platform for the reporting database are the same as the primary database.