

ORACLE 10G: DATA REPLICATION MADE EASY

Kirtikumar Deshpande, Verizon Information Services

Introduction

Oracle Streams was introduced in Oracle9i Database Release 2 as a new solution for information sharing among applications and databases in today's distributed business environments. It provided a solid infrastructure for data replication using very flexible architecture to capture stage and propagate, and apply data modifications. Oracle Database 10g enhanced Oracle Streams and offered a number of additional features to simplify configuration, maintenance and monitoring. In addition to data replication, you can use Oracle Streams for event notification, loading data into a data warehouse, or keeping a copy of the data at a remote site.

This paper introduces you to Oracle 10g Streams. It shows you how to setup a simple Oracle10g Streams environment for data replication.

What Is Oracle Streams

Oracle Streams is the flow of information either within a single database or from one database to another. Oracle Streams can be setup in homogeneous (all Oracle databases) or heterogeneous (non-Oracle and Oracle databases) environments.

The Streams setup uses a set of processes and database objects to share data and messages. The database changes (DDL and DML) are captured at the source; those are then staged and propagated to one or more destination databases to be applied there. The unit of information put into the Streams is called an Event. Your application can also put information, or events, into the Streams to share it with other applications. These events are called User Defined events. These user defined events are typically different than the DDL and DML changes. Those may or may not be stored in database tables, but are simply passed between Applications. How Oracle Streams handles these events is beyond the scope of this paper.

How Streams Flow

The flow of the Oracle Streams starts by capturing database changes. As you know, all the database changes are recorded in the redo log files, with the exception of some operations that use 'nologging' option. Oracle Streams employs the Log Miner technology to capture the database changes from the redo logs. The captured information is then formatted into Logical Change Record (LCR) to put into the Streams queue. Oracle uses Advance Queuing mechanism to propagate the LCRs to another Streams queue on destination server. The Streams process on the destination server reads the LCRs and applies them to the destination tables.

Figure 1 below shows the basic flow of the information in Oracle Streams. There are three basic tasks, or elements, in the Oracle Streams setup, viz. Capture, Stage & Propagate, and Apply.



Figure 1. Oracle Streams flow

Capture

A Capture process is an Oracle background process with the name *cnnn*, where *nnn* is the process number. In Oracle10g the process number varies from 001 to 999. The Capture process reads the redo log files, and when necessary, the archived redo log files, to extract DDL and DML database changes. You must keep the archived redo logs available until all the changes within that log file are captured and applied. The captured changes are formatted into LCRs. There are two types of LCRs. All DML changes are formatted into Row LCRs (or DML LCRs), while DDL changes are in DDL LCRs. You can setup rules to specify what DML and DDL changes are to be captured into LCRs. You can choose to setup such rules to capture DDL and/or DML changes at database, schema or table level. The formatted LCRs are then put into Streams queue.

In Oracle 10g, the Streams queue resides in the Streams pool within the SGA. However, the streams pool must be configured using the new `STREAMS_POOL_SIZE` initialization parameter. If a separate streams pool is not configured, then the streams queue reside in the shared pool occupying up to 10% of the shared pool memory. When the in-memory streams queue gets full, the information is written to a queue table in the database.

Stage And Propagate

The Streams queue is used to stage events for propagation and consumption. The Source queue propagates the events to the Destination queue. The source and destination queues can have one-to-one, one-to-many, many-to-one or many-to-many relationship. The queue itself is an object of type `SYS.AnyData`. This is a new type of object that can contain data of any other type, such as `DATE`, `VARCHAR2`, and `NUMBER` etc. The queue can contain two types of events for staging and propagating. These are LCRs and user messages. LCRs are automatically captured database changes while user messages are custom messages that are created by the applications or user processes, such as event notification. The staged events can be consumed by local application or can be propagated to a streams queue.

The events remain on the source queue until the destination process consumes or dequeues them. You can define rules for propagation, and control what events are propagated. In addition, you can control the timing of propagation.

Apply

An Apply process is an Oracle background process with the name *annn*, where *nnn* is the process number. In Oracle10g the process number varies from 001 to 999. The Apply process dequeues the LCRs from the specified Streams queue and applies them to the local database or passes it as a parameter to user-defined procedure. At this point, the event is considered consumed, and removed from the source queue. This marks the termination of the streamed event.

You can define rules for the Apply process to control how to apply, or ignore, the event. The rule can be setup at the database, schema or table level.

The Apply process can also detect data conflicts. For update conflicts it can use an update conflict handler procedure if you defined one. Oracle Streams offers a variety of pre-built conflict handlers that can assist you in defining conflict handler routines to satisfy your application requirements. You can also write your own conflict resolution handlers.

Configuring Streams

There are two ways to configure Oracle Streams: using the Streams Wizard in Oracle Enterprise Manager, or using the command line instructions.

In Oracle 10g Release 2, OEM Database Control, click on Maintenance, and then on Streams Setup. In the Streams Setup Options, start the Streams Wizard by clicking on 'Streams Global, Schema, Table and Subset Replication Wizard' link, and then follow the instructions.

This paper shows how to use the command line instructions to setup streams. There are several steps in configuring a database for implementing Oracle Streams. These steps are described below.

In this example setup, the source database name is DBXT and the destination database name is DBXP.

The DBXT database captures and propagates DDL and DML changes to DBXP to be applied there. Data replication will be setup for a table named TEST under STRMDEMO schema in both these databases.

Set Initialization Parameters

Various initialization parameters must be set for Capture, Propagation and Apply process. Database restart will be required after changing the init.ora file.

Following list shows these parameters with a comment specifying which streams process the parameter applies to.

The values specified for these parameters are only for illustration purpose. The actual values for these parameters in your environment may be different.

A brief description of the parameters follows the list.

- | | | |
|-----------------------------------|----------|---|
| 1. AQ_TM_PROCESSES | = > 1 | # Capture, Apply (ONLY for User Events) |
| 2. COMPATIBLE | = 10.2.0 | # Capture, Propagation, Apply |
| 3. GLOBAL_NAMES | = TRUE | # Capture, Propagation, Apply |
| 4. JOB_QUEUE_PROCESSES | = 2 | # Propagation |
| 5. LOGMNR_MAX_PERSISTENT_SESSIONS | = > 1 | # Capture (Obsolete in Oracle 10g R2) |
| 6. OPEN_LINKS | = 4 | # Propagation |
| 7. PARALLEL_MAX_SERVERS | = 8 | # Capture, Apply |
| 8. PROCESSES | = 100 | # Capture, Propagation, Apply |
| 9. SHARED_POOL_SIZE | = 200M | # Capture (Min 100 MB, +10 MB/capture) |
| 10. STREAMS_POOL_SIZE | = 500M | # Capture, Propagation, Apply |
| 11. TIMED_STATISTICS | = TRUE | # Capture, Apply |

AQ_TM_PROCESSES: If User defined events will be put into Streams then set this parameter to a value of 1. Else, do not use this parameter at all. This parameter establishes queue monitor processes. The value denotes the number of queue monitor processes to start. These processes are responsible for managing time-based operations of messages such as delay and expiration, cleaning up the messages after specified retention time.

COMPATIBLE: To use Oracle10g Streams this parameter must be set to 10.1.0 or higher.

GLOBAL_NAMES: For Streams, this parameter must be set to TRUE. It specifies if a database link is required to have the same name as the database to which it connects.

JOB_QUEUE_PROCESSES: It specifies the number of job queue processes for each instance (J000 to J999). If multiple propagations on a database are to run simultaneously, then set this parameter to be twice the number of simultaneous propagation desired. Minimum value of 2 works well in most cases.

LOGMNR_MAX_PERSISTENT_SESSIONS: It specifies the maximum number of persistent Log Miner sessions to mine the redo logs. If you want to run multiple Capture processes in the

database, then set this parameter to a value equal to or greater than the number of Capture processes. This parameter is obsolete as of Oracle 10g R2. It will be set to a value of 1 internally.

OPEN_LINKS: It specifies the number of concurrent open connections to remote databases from one session. You may need to increase the setting of this parameter to allow enough open connections for the propagation jobs to connect to remote databases.

PARALLEL_MAX_SERVERS: The maximum number of parallel servers available for parallel processing. LogMiner session will typically request 3 servers. So, the value set for this parameter should be sufficiently large to serve Log Miner session as well as any other parallel processing by Application.

PROCESSES: It specifies the maximum number of user processes that can simultaneously connect to the database. You need to make sure that the value for this parameter is large enough to allow for all the additional Streams processes.

SHARED_POOL_SIZE: A minimum of 100MB of shared pool is recommended when using Streams.

STREAMS_POOL_SIZE: Oracle10g introduced a Streams pool in the SGA to exclusively allocate memory for Streams internal queues. Using this parameter, you can reserve the specified amount of memory for Streams. If this parameter is not used, or set to zero, then Streams uses 10% of Shared pool memory.

If you use SGA_TARGET to let Oracle allocate memory for buffer cache and all other memory pools, then you will not need SHARED_POOL_SIZE and STREAMS_POOL_SIZE. However, you may need to increase SGA_TARGET if you decide to implement Streams.

TIMED_STATISTICS: Make sure that it is set to TRUE to fully populate V\$STREAMS_CAPTURE view which shows statistics for the Capture process.

In this example, both the databases, DBXT and DBXP, are setup with required init.ora parameters.

Configure Archive Logging

Since Oracle Streams uses redo log files to extract database changes, you must enable archive logging in the databases that are running the Capture process. If your database is not operating in archive log mode, you will need to modify related init.ora parameters and then restart the database to operate in archive log mode. DBXT and DBXP databases both operate in archive log mode. However, please note that, the destination database can run in noarchive log mode.

Relocate Logminer Tables

By default, all Log Miner tables are created under SYSTEM schema in the *sysaux* tablespace, and can be left alone.

In Oracle9i those were created in the *system* tablespace, and it was recommended that those be moved to their own tablespace. These tables and their indexes will grow in size. Therefore, it was advisable to have their own tablespace with adequate storage parameters.

In Oracle10g it is not required to move these tables to another tablespace. However, I chose to move them to their own tablespace as I used the same old configuration scripts from my Oracle9i Streams setup.

In the following example, a new tablespace *logminer_ts* is created and Oracle supplied procedure, DBMS_LOGMNR_D.SET_TABLESPACE, is used to re-create all Log Miner tables in that tablespace at the Capture site.

```
REM - ===== Oracle9i Streams Setup =====  
REM - Create a tablespace and move logminer tables from SYSTEM tablespace.  
REM - Change tablespace name, datafile name and size as desired and run
```

```
REM - this script at the Capture site.  
REM - =====
```

```
conn / as sysdba  
  
create tablespace logminer_ts  
datafile '/u19/oradata/DBXT/logminer_ts_01.dbf' size 2000M  
extent management local uniform size 512k  
segment space management auto  
/  
  
alter user system default tablespace logminer_ts;  
  
begin  
  dbms_logmnr_d.set_tablespace('LOGMINER_TS');  
end;  
/  
  
alter user system default tablespace system;
```

Also, please note that in the above example, the default tablespace for SYSTEM user was temporarily changed. I have included those commands here because I found that a couple of indexes were still left behind in the *system* tablespace. Those indexes will grow in size. It is likely that the supplied packaged procedure does not have the 'tablespace' clause for all of the indexes that it recreates. Therefore, this workaround takes care of moving everything to the new log miner tablespace.

Enable Supplemental Logging and Force Logging

You must add supplemental logging at the source database either at the database level or at the table level. Supplemental logging places additional column data in the redo log file whenever an UPDATE operation is performed on these columns. The Capture process puts this additional information in the LCRs to be used by the Apply process. You can read more about supplemental logging in the Oracle10g Streams manual.

Supplemental logging at the database level will generate a lot more redo log entries, and it may affect database performance. So, please choose this option carefully.

Setting supplemental logging at the table level, and only for those tables that are participating in Streams configuration, is strongly suggested.

Following example shows how to add supplemental logging to a table. The ALWAYS option will add the ACCOUNT_NUM column to the redo log entry every time any other column of the table is updated. The ACCOUNT_NUM is defined as NOT NULL and is the primary key column. The table TEST is owned by STRMDEMO schema. It is in the DBXT database.

```
CONN STRMDEMO/&pw  
  
ALTER TABLE TEST  
  ADD SUPPLEMENTAL LOG GROUP SLGPK_TEST  
  (ACCOUNT_NUM) ALWAYS;
```

In addition, when using Oracle Streams you should not perform any 'nologging' operations on the objects that are participating in Streams. Such operations will not generate redo log entries and

will not be captured for replication. To be certain that all required redo gets written to the redo log files, it is recommended that you change the associated tablespaces to always log all changes. This is called forced logging.

The following commands set up tablespaces STRMDEMO_DATA and STRMDEMO_NDX to log all redo even when 'nologging' operations are performed on any of the objects they contain.

```
ALTER TABLESPACE STRMDEMO_DATA FORCE LOGGING;  
ALTER TABLESPACE STRMDEMO_NDX FORCE LOGGING;
```

You can also set the forced logging at the database level using the following command:

```
ALTER DATABASE FORCE LOGGING;
```

The column FORCE_LOGGING in DBA_TABLESPACES and V\$DATABASE view will show if forced logging is in effect or not. To turn it off, simply change 'FORCE LOGGING' to 'NO FORCE LOGGING' in the above commands and rerun those.

Create Streams Administrator Account

In each database participating in Streams, you must create an administrative user account. This user account will be designated as the Streams Administrator. This account must be granted various and powerful privileges. Oracle will create certain tables under this schema to store Streams queue data and other information. It is recommended that this account should have its own tablespace. I will use STRMADMIN for the account name and STREAMS_QUEUE_TS as its default tablespace. The following script creates the Streams Administrator account in DBXT database.

```
REM - Create STRMADMIN user  
REM - =====  
  
CONN / AS SYSDBA  
  
REM - Create tablespace to store messages and objects owned by STRMADMIN user  
REM - =====  
  
CREATE TABLESPACE streams_queue_ts datafile  
  '/u19/oradata/DBXT/streams_queue_ts_01.dbf' SIZE 1000M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K  
SEGMENT SPACE MANAGEMENT AUTO;  
  
CREATE USER STRMADMIN IDENTIFIED BY &strmadmin_pw  
DEFAULT TABLESPACE streams_queue_ts  
QUOTA UNLIMITED ON streams_queue_ts  
TEMPORARY TABLESPACE TEMP;  
  
REM - Grant all the required privileges to Admin account  
REM - =====  
EXEC DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE('STRMADMIN');  
  
REM - Following privileges are not a requirement, but will be helpful when using  
REM - OEM to monitor streams and perform other tasks.  
REM - =====  
GRANT SELECT ANY DICTIONARY TO STRMADMIN;  
GRANT DBA TO STRMADMIN;
```

If you want to know what all privileges are granted to this account by the DBMS_STREAMS_AUTH package, you can execute the following command and check the generated file.

```
CONN / AS SYSDBA

BEGIN
  DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
    GRANTEE      => 'STRMADMIN',
    FILE_NAME    => 'streams_admin_privs.sql',
    DIRECTORY_NAME => 'dumpdir');
END;
/
```

In the above example the file, streams_admin_privs.sql, under the directory object 'dumpdir' will list all the commands that granted various privileges to the STRMADMIN account. You can run this SQL script instead of executing the shown procedure, DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE ('STRMADMIN'), to setup this account.

Create Streams Queue

The Streams information flows from the Capture site to the Apply site through the queues. The package DBMS_STREAMS_ADM provides a procedure called SET_UP_QUEUE to create the required Streams queue.

Following script will create the Streams queue called DBXT_TO_DBXP owned by STRMADMIN. The queue table STREAMS_QUEUE_TABLE will also be created in the specified tablespace.

Streams queue must be created at the Capture and Apply site.

```
REM - Create streams queue for data propagation.
REM - Run as STRMADMIN at both Capture and Apply sites.
REM - =====
```

```
CONN STRMADMIN/&pw

BEGIN
  DBMS_STREAMS_ADM.SET_UP_QUEUE(
    queue_table   => 'STREAMS_QUEUE_TABLE',
    storage_clause => 'TABLESPACE STREAMS_QUEUE_TS',
    queue_name    => 'DBXT_TO_DBXP',
    queue_user    => 'STRMADMIN');
END;
/
```

Create Apply Table Rules at Apply Site (DBXP Database)

Since we are configuring data replication for a table, we must now add table rules for the Apply process. The Apply process will use these rules when applying changes to the table. Oracle supplied package DBMS_STREAMS_ADM has a procedure called ADD_TABLE_RULES to setup these rules.

Following script adds the rule for STRMDEMO.TEST table to the Apply process. The rule will specify that the DDL and DML changes will be applied to this table and that the changes will be dequeued from streams queue named STRMADMIN.DBXT_TO_DBXP that was already setup at the Apply site.

The procedure assigns the name STREAMS_APPLY to the Apply process. It also specifies the source database name.

```

REM - Add table rules for Apply process
REM - =====
CONN STRMADMIN/&pw
BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name    => 'STRMDEMO.TEST',
    streams_type  => 'APPLY',
    streams_name  => 'STREAMS_APPLY',
    queue_name    => 'STRMADMIN.DBXT_TO_DBXP',
    include_dml   => TRUE,
    include_ddl   => TRUE,
    source_database => 'DBXT.world');
END;
/

```

Create Apply User At Apply Site (DBXP Database)

The Apply user is the user account in the destination database that will apply all the DML and DDL changes to the destination table. The package DBMS_APPLY_ADM has a procedure called ALTER_APPLY to specify the user name. The user specified in the APPLY_USER parameter must have the necessary privileges to perform all DDL and DML operations on the table.

In this example I use the STRMADMIN schema user as the apply user.

```

REM - Define the Apply user at the Destination database.
REM - =====

CONN / AS SYSDBA

BEGIN
  DBMS_APPLY_ADM.ALTER_APPLY(
    apply_name => 'STREAMS_APPLY',
    apply_user => 'STRMADMIN');
END;
/

REM - Grant all privileges on destination table to Streams Apply user
REM - =====

CONN STRMDEMO/&pw

grant all on test to strmadmin;

```

Change Apply process Parameter (DBXP Database)

By default, the Apply process will terminate for any error it encounters and will stay disabled. Relevant messages will be written to the alert log file. I think it is a good idea to let the Apply process continue processing other events after reporting the ones that caused the error. The error will still be visible in the DBA_APPLY_ERROR view, and the associated event can be replayed if the error could be resolved. Using the SET_PARAMETER procedure in the DBMS_APPLY_ADM package we can instruct Oracle Streams to not abort the Apply process on error. Following script shows how to do this.

```
REM - Change Apply process parameter 'DISABLE_ON_ERROR' from default 'Y' to 'N'
REM - So that the Apply process is not aborted when errors are detected.
REM - =====
```

```
CONN STRMADMIN/&pw
```

```
BEGIN
  DBMS_APPLY_ADM.SET_PARAMETER(
    apply_name => 'STREAMS_APPLY',
    parameter  => 'DISABLE_ON_ERROR',
    value      => 'N' );
END;
/
```

Start Apply process (DBXP Database)

At this point, we can start the Apply process at the destination database. All the downstream (Apply) components must be in place before we start the Capture process on the source database. The Apply process is started using the START_APPLY procedure in the DBMS_APPLY_ADM package. Following script has the required command. Review alert log file to make sure the Apply process was started successfully without any errors. You will notice that the background process a001 is now active.

```
REM - Start the Apply process at the destination database
REM - =====
```

```
CONN STRMADMIN/&pw
```

```
BEGIN
  DBMS_APPLY_ADM.START_APPLY(apply_name => 'STREAMS_APPLY');
END;
/
```

Please note that we have not confirmed if the destination table is available in the destination database. We only completed the Streams setup at the Apply site.

Create Database Link At Source Site (DBXT Database)

Oracle Streams uses Oracle Net (SQL*Net) to move data from the source queue to the destination queue. Therefore, a database link from the source to the destination is required.

Following script creates the database link in the DBXT database. Please notice that this is a private database link under STRMADMIN schema in the source database.

```
REM - Create DB Link to the destination database
REM - =====
```

```
CONN STRMADMIN/&pw
```

```
create database link DBXP
connect to strmadmin identified by &strmadmin_pw_applysite
using 'DBXP.world'
/
```

Create Capture Table Rules At Capture Site (DBXT Database)

We need to add table rules for the Capture process, just like we did for the Apply process. The Capture process when capturing changes to the table at the Capture site will use these rules.

Following script adds the rules for STRMDEMO.TEST table to the CAPTURE process specifying that the DDL and DML change to this table will be captured and sent to STRMADMIN.DBXT_TO_DBXP.

The procedure assigns the name STREAMS_CAPTURE to the Capture process. It also specifies the source database name.

```
REM - Add table rules for Capture process
REM - =====

CONN STRMADMIN/&pw

BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_RULES(
    table_name      => 'STRMDEMO.TEST',
    streams_type    => 'CAPTURE',
    streams_name    => 'STREAMS_CAPTURE',
    queue_name      => 'STRMADMIN.DBXT_TO_DBXP',
    include_dml     => TRUE,
    include_ddl     => TRUE,
    source_database => 'DBXT.world');
END;
/
```

Create Propagation Table Rules At Capture Site (DBXT Database)

All captured and enqueued events must be propagated to the destination queue. To do this, Oracle Streams uses a propagation process that maps the source queue to the destination queue. The procedure ADD_TABLE_PROPAGATION_RULES in the DBMS_STREAM_ADM package creates the propagation process.

In the following script, a propagation process STRMADMIN_PROPAGATE will be created and changes captured for STRMDEMO.TEST table in the STRMADMIN.DBXT_TO_DBXP will be sent to the STRMADMIN.DBXT_TO_DBXP at the destination database DBXP.

All DDL and DML changes will be propagated to the destination database. The database link will be used to connect to the destination database.

```
REM - Add propagation rules at Capture site
REM - =====

CONN STRMADMIN/&pw

BEGIN
  DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
    table_name          => 'STRMDEMO.TEST',
    streams_name        => 'STREAMS_PROPAGATE',
    source_queue_name   => 'STRMADMIN.DBXT_TO_DBXP',
    destination_queue_name => 'STRMADMIN.DBXT_TO_DBXP@DBXP.world',
    include_dml         => TRUE,
    include_ddl         => TRUE,
    source_database     => 'DBXT.world');
END;
```

/

Set Instantiation SCN for Destination Table (DBXP Database)

At this point all the Oracle Streams setup at the Capture and Apply site is ready. It is just not completely active yet. We need to create the destination table in the destination database, if it did not already exist. In addition, we must set the instantiation SCN (system change number) for the table from the source database. This ensures that all the changes made to the source table after this SCN will be applied to the destination table.

There are following ways to set the instantiation SCN for the destination table:

1. If table is not present in the destination database, perform an export/import of the table
 - a. Export from source database using OBJECT_CONSISTENT=Y
 - b. Import in destination database using STREAMS_INSTANTIATION=Y
2. You can also use Transportable Tablespace feature to copy tables from source to the destination database. You will still need to manually set the instantiation SCN for the destination table.
3. If the table is already present in the destination database and the data in these tables is identical, you can set the instantiation SCN by exporting/importing the metadata only
 - a. Export with rows=n, OBJECT_CONSISTENT=Y
 - b. Import with ignore=y, STREAMS_INSTANTIATION=Y
4. Manual instantiation:

Find current SCN at the source database from V\$DATABASE.CURRENT_SCN column.

```
conn strmadmin/&pw
```

```
SET NUMWIDTH 18  
select current_scn from v$database;
```

Use the SCN from above to set the instantiation SCN for the table in the destination database. Following script shows how to do this for the STRMDEMO.TEST table.

```
conn strmadmin/&pw@DBXP
```

```
begin  
  dbms_apply_adm.set_table_instantiation_scn(  
    source_object_name => 'STRMDEMO.TEST',  
    source_database_name => 'DBXT.WORLD',  
    instantiation_scn => &SCN);  
end;  
/
```

Start Capture process (DBXT Database)

Now, we have completed all the required setup for Oracle Streams to replicate changes made to STRMDEMO.TEST table in DBXT (Source) database to the same table in DBXP (Destination) database. All we need to do now is to start the Capture process that was already created.

The Capture process is started using the START_CAPTURE procedure in the DBMS_CAPTURE_ADM package as shown in the following script. Review alert log file to make sure the Capture process

was started successfully, and did not abort. You will notice that the background process c0001 is now active.

```
REM - Start the Capture process
REM - =====

CONN STRMADMIN/&pw

BEGIN
  DBMS_CAPTURE_ADM.START_CAPTURE(
    capture_name => 'STREAMS_CAPTURE');
END;
/
```

Oracle10g Streams setup is now complete. All changes (DDL and DML) made to STRMDEMO.TEST table in DBXT database will be replicated to STRMDEMO.TEST table in DBXP database.

Turbulences in Oracle Streams

During my implementation of Oracle Streams I encountered a few 'turbulences' listed below. Oracle had workarounds to address those.

- The Capture process may need to read an older archived log when it is restarted. This problem occurs due to how log miner checkpoints are captured. You may need to change a couple of Capture process parameters as shown in the following script. In addition, you may need to periodically force the log miner check point to keep in synch the CAPTURED_SCN and APPLIED_SCN values in DBA_CAPTURE view.

```
CONN STRMADMIN/&pw

begin
  dbms_capture_adm.set_parameter (
    capture_name => 'streams_capture',
    parameter    => '_CHECKPOINT_FORCE',
    value        => 'Y');
end;
/

REM - Force a logminer checkpoint after 1MB of redo is written.
REM - =====
begin
  dbms_capture_adm.set_parameter (
    capture_name => 'streams_capture',
    parameter    => '_CHECKPOINT_FREQUENCY',
    value        => '1');
end;
/
```

- OEM 9.2.0.1 (and OMS) needs patch # 3023858 to correct a few issues with the Streams Wizard.
- By default Oracle10g Streams uses up to 10MB from shared pool for Log Miner process. At times, this memory will not be sufficient and Oracle will report ORA-1341: LogMiner out-of-memory error in the alert log file. The Capture process will get terminated. To avoid this error, you may need to allocate more memory to Log Miner process. This is done by setting

_SGA_SIZE parameter of the Capture process as shown below. Please note that this is not an initialization parameter. If you do not have enough free memory in shared pool, it is advisable to increase it to accommodate the increase in the Log Miner memory allocation before making this change.

```
REM - Change Log Miner memory allocation to 50MB
REM - =====

begin
  dbms_capture_adm.set_parameter (
    capture_name => 'streams_capture',
    parameter    => '_SGA_SIZE',
    value        => '50');
end;
/
```

Streams Monitoring

Following are a few of the views that I found useful to monitor Oracle Streams. I use these just to see if the capture and Apply processes are running and if there were any errors reported by the Apply process.

DBA_APPLY_ERROR - Displays information about error transactions encountered by Apply process.

DBA_APPLY - Displays information about all Apply processes, and their status.

DBA_CAPTURE - Displays information about Capture processes and their status

Following views show how the Apply process is working.

V\$STREAMS_APPLY_READER

V\$STREAMS_APPLY_SERVER

V\$STREAMS_APPLY_COORDINATOR

The easiest way to monitor Oracle Streams processes is to use OEM screens. Those screens provide useful information and statistics on all of the streams processes.

Oracle Corporation also provides a utility called 'strmmon'. I have not used it yet, and I believe one has to ask Oracle Support to get it.

Oracle Support also has another script that you should ask for. It is called 'streams_hc_10GR1.sql' (for Oracle9i Streams it is 'streams_health_check.sql'). This is a comprehensive script that reports all of your Streams setup information. If you run into any Streams problem, Oracle Support would ask for a report from this script.

Oracle Streams Resources

Here are the resources that I used most for Oracle Streams:

Oracle Metalink: Streams Library Index Page (Note# 248875.1)

Oracle Metalink: 10g Streams Recommended Configuration (Note # 298877.1)

Oracle10g Streams Concepts and Administration Guide

Oracle10g Streams Replication Administrator's Guide

Oracle Technology Network – Oracle Streams Forum:
<http://forums.oracle.com/forums/forum.jsp?forum=70>

http://www.oracle.com/technology/sample_code/tech/streams/index.html

There is a lot more to Oracle10g Streams than what I discussed in this paper. I hope you will find this new feature interesting to learn more about it.

About The Author

Kirtikumar Deshpande works for Verizon Information Services (www.Superpages.com) as a Senior Oracle DBA. He has over 25 years of IT experience including over 11 years as an Oracle DBA. He co-authored Oracle Press books titled 'Oracle Wait Interface: A Guide to Performance Diagnostics & Tuning' and 'Oracle Performance Tuning 101'. He has presented a number of papers at various Oracle User Group Conferences in the USA and abroad.