

1. Prepare

In this Streams test environment, there are two databases called “strm1” and “strm2”, whose global name are “strm1.com” and “strm2.com” respectively. The source side is “strm1”, and target side is “strm2”. The synchronization level is SCHEMA level and double-directions replication.

1.1 Init parameter prerequisite (Both db)

a) global_names should be true. global_name =db_name+”.”+domain_name

```
alter system set global_names=true;
select * from global_name;
GLOBAL_NAME
-----
STRM1.COM
GLOBAL_NAME
-----
STRM2.COM
```

b) job_queue_processes >= 2

```
alter system set job_queue_processes=10;
```

c) compatible >= 10.2 and target side >= source side

```
show parameter compatible
```

d) streams_pool_size should set up with suitable value (here is 50MB)

```
alter system set streams_pool_size=50M scope=spfile;
```

e) archive mode is enabled in source side

```
archive log list
Database log mode Archive Mode
.....
```

f) open_links >= 4

```
alter system set open_links=4 scope=spfile;
```

g) parallel_max_servers (each capture process and apply process can use multiple parallel execution servers).

```
alter system set parallel_max_servers=20 scope=both;
```

h) change logminer data dictionary default tablespace from SYSAUX to TBS_LOGMN tablespace

```
-- strm1
create tablespace tbs_logmn datafile '/oracle/oradata/strm1/tbs_logmn01.dbf' size 100m autoextend on;
execute dbms_logmnr_d.set_tablespace('tbs_logmn');
-- strm2
create tablespace tbs_logmn datafile '/oracle/oradata/strm2/tbs_logmn01.dbf' size 100m autoextend on;
execute dbms_logmnr_d.set_tablespace('tbs_logmn');
```

i) supplemental logging

In 10gR2, supplemental logging is automatically configured for tables on which primary, unique, or foreign keys are defined when database object is prepared for Streams capture.

Related concepts, please refer < Streams Replication Administrator's Guide (10.2)> Chapter 12

Part Number B14228-04

Supplemental log can be explicitly enabled by command below:

```
-- minimal supplemental log mode
alter database add supplemental log data;
COLUMN log_min HEADING 'Minimum|Supplemental|Logging?' FORMAT A12
COLUMN log_pk HEADING 'Primary Key|Supplemental|Logging?' FORMAT A12
COLUMN log_fk HEADING 'Foreign Key|Supplemental|Logging?' FORMAT A12
COLUMN log_ui HEADING 'Unique|Supplemental|Logging?' FORMAT A12
COLUMN log_all HEADING 'All Columns|Supplemental|Logging?' FORMAT A12
SELECT SUPPLEMENTAL_LOG_DATA_MIN log_min,
SUPPLEMENTAL_LOG_DATA_PK log_pk,
SUPPLEMENTAL_LOG_DATA_FK log_fk,
SUPPLEMENTAL_LOG_DATA_UI log_ui,
SUPPLEMENTAL_LOG_DATA_ALL log_all
FROM V$DATABASE;
```

Minimum	Primary Key	Foreign Key	Unique	All Columns
Supplemental Logging?	Supplemental Logging?	Supplemental Logging?	Supplemental Logging?	Supplemental Logging?
YES	NO	NO	NO	NO

Disable supplemental log is using “alter database drop supplemental log data”.

Note: If you forget to enable it, it is OK. Because when configuring Streams capture process, it will be automatically enabled.

1.2 Edit tnsnames.ora in both sides

```
STRM1 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.10.101)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = strm1.com)
    )
  )
STRM2 =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.10.101)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = strm2.com)
    )
  )
```

2. Streams administrator account creation

2.1 Create streams admin in strm1 db

```
-- Create streams tablespace
sqlplus sys@strm1 as sysdba
create tablespace tbs_stream datafile '/oracle/oradata/strm1/tbs_stream01.dbf' size 100m autoextend on;
-- Create streams admin
create user strmadmin identified by nhy67ujm default tablespace tbs_stream temporary tablespace temp;
-- streams admin MUST have dba privilege
grant connect,resource,dba to strmadmin;
begin
dbms_streams_auth.grant_admin_privilege(
grantee => 'strmadmin',
grant_privileges => true);
end;
/
-- Confirm
select * from dba_streams_administrator;

```

USERNAME	LOC	ACC
STRMADMIN	YES	YES

2.2 Create streams admin in strm2 db

```
-- Create streams tablespace
sqlplus sys@strm2 as sysdba
create tablespace tbs_stream datafile '/oracle/oradata/strm2/tbs_stream01.dbf' size 100m autoextend on;
-- Create streams admin
create user strmadmin identified by nhy67ujm default tablespace tbs_stream temporary tablespace temp;
-- Grant dba to streams admin
grant connect,resource,dba to strmadmin;
begin
dbms_streams_auth.grant_admin_privilege(
grantee => 'strmadmin',
grant_privileges => true);
end;
/
-- Confirm
select * from dba_streams_administrator;

```

USERNAME	LOC	ACC
STRMADMIN	YES	YES

```
-----
STRMADMIN                                YES YES
```

3. Create Database link

```
-- Note: database link name must same as global name of target db
-- command below can change global name
alter database rename global to strm1.com;
-- strm1: connect as stream admin
conn strmadmin@strm1
strmadmin@STRM1> create database link strm2.com connect to strmadmin identified by nhy67ujm using 'strm2';
-- login as sys
sys@STRM1> select NAME,CTIME,USERID from sys.link$;
NAME          CTIME          USERID
-----
STRM2.COM     2009/05/03 23:55:16 STRMADMIN
-- strm2: connect as stream admin
conn strmadmin@strm2
strmadmin@STRM2> create database link strm1.com connect to strmadmin identified by nhy67ujm using 'strm1';
-- login as sys
sys@STRM2> select NAME,CTIME,USERID from sys.link$;
NAME          CTIME          USERID
-----
STRM1.COM     2009/05/11 20:50:47 STRMADMIN
```

4.1 Configure streams queue in both sides. (Afterwards, if not specially mentioned, the user is always “strmadmin”)

```
-- strm1
begin
dbms_streams_adm.set_up_queue(
queue_table => 'strm1_cap_queue_table',
queue_name => 'strm1_cap_queue');
end;
/
begin
dbms_streams_adm.set_up_queue(
queue_table => 'strm1_apl_queue_table',
queue_name => 'strm1_apl_queue');
end;
/
-- confirm
strmadmin@STRM1> select name,queue_table from user_queues;
NAME          QUEUE_TABLE
-----
AQ$_STRM1_CAP_QUEUE_TABLE_E STRM1_CAP_QUEUE_TABLE
STRM1_CAP_QUEUE STRM1_CAP_QUEUE_TABLE
AQ$_STRM1_APL_QUEUE_TABLE_E STRM1_APL_QUEUE_TABLE
STRM1_APL_QUEUE STRM1_APL_QUEUE_TABLE
-- strm2
begin
dbms_streams_adm.set_up_queue(
queue_table => 'strm2_apl_queue_table',
queue_name => 'strm2_apl_queue');
end;
/
begin
dbms_streams_adm.set_up_queue(
queue_table => 'strm2_cap_queue_table',
queue_name => 'strm2_cap_queue');
end;
/
```

```
-- confirm
strmadmin@STRM2> select name,queue_table from user_queues;
NAME                                QUEUE_TABLE
-----                                -
AQ$_STRM2_CAP_QUEUE_TABLE_E        STRM2_CAP_QUEUE_TABLE
STRM2_CAP_QUEUE                     STRM2_CAP_QUEUE_TABLE
AQ$_STRM2_APL_QUEUE_TABLE_E        STRM2_APL_QUEUE_TABLE
STRM2_APL_QUEUE                     STRM2_APL_QUEUE_TABLE
```

Note: Program of removing queue is as below:

```
begin
dbms_streams_adm.remove_queue(
queue_name => 'strm1_cap_queue',
cascade => true);
end;
/
```

4.2 Create capture process in strm1 side

```
-- strm1: set up capture process
-- before setup, minimal supplemental log is disabled
SELECT SUPPLEMENTAL_LOG_DATA_MIN log_min,SUPPLEMENTAL_LOG_DATA_PK log_pk,
SUPPLEMENTAL_LOG_DATA_FK log_fk,SUPPLEMENTAL_LOG_DATA_UI log_ui,
SUPPLEMENTAL_LOG_DATA_ALL log_all FROM V$DATABASE;
Minimum      Primary Key  Foreign Key  Unique      All Columns
Supplemental Supplemental Supplemental Supplemental Supplemental
Logging?     Logging?     Logging?     Logging?     Logging?
-----
NO           NO           NO           NO           NO

begin
dbms_streams_adm.add_schema_rules(
schema_name => 'dbausr',
streams_type => 'capture',
streams_name => 'strm1_capture',
queue_name => 'strmadmin.strm1_cap_queue',
include_dml => true,
include_ddl => true,
include_tagged_lcr => false,
source_database => null,
inclusion_rule => true);
end;
/

-- after set up, supplemental log is enabled automatically
Minimum      Primary Key  Foreign Key  Unique      All Columns
Supplemental Supplemental Supplemental Supplemental Supplemental
Logging?     Logging?     Logging?     Logging?     Logging?
-----
YES          NO           NO           NO           NO
```

4.3 Create capture process in strm2 side

```
-- strm1: set up capture process
-- before setup, minimal supplemental log is disabled
SELECT SUPPLEMENTAL_LOG_DATA_MIN log_min,SUPPLEMENTAL_LOG_DATA_PK log_pk,
SUPPLEMENTAL_LOG_DATA_FK log_fk,SUPPLEMENTAL_LOG_DATA_UI log_ui,
SUPPLEMENTAL_LOG_DATA_ALL log_all FROM V$DATABASE;
Minimum      Primary Key  Foreign Key  Unique      All Columns
Supplemental Supplemental Supplemental Supplemental Supplemental
Logging?     Logging?     Logging?     Logging?     Logging?
-----
NO           NO           NO           NO           NO
```

作者: 叶熙昌

blog: <http://www.orafan.net>

```

begin
dbms_streams_adm.add_schema_rules(
schema_name => 'dbausr',
streams_type => 'capture',
streams_name => 'strm2_capture',
queue_name => 'strmadmin.strm2_cap_queue',
include_dml => true,
include_ddl => true,
include_tagged_lcr => false,
source_database => null,
inclusion_rule => true);
end;
/
-- after set up, supplemental log is enabled automatically

```

Minimum Supplemental Logging?	Primary Key Supplemental Logging?	Foreign Key Supplemental Logging?	Unique Supplemental Logging?	All Columns Supplemental Logging?
YES	NO	NO	NO	NO

5. Duplicate DBAUSR schema from source to target

5.1 Duplicate by using exp/imp

```

-- make sure DBAUSR user exists in strm2 side, or else create it
strmadmin@STRM2> create user dbausr identified by nhy67ujm default tablespace users temporary tablespace temp;
strmadmin@STRM2> grant connect,resource to dbausr;
-- export DBAUSR schema from source side
exp userid=dbausr/nhy67ujm@strm1 file='/tmp/dbausr.dmp' object_consistent=y rows=y
-- import to strm2 side
imp userid=system/nhy67ujm@strm2 file='/tmp/dbausr.dmp' ignore=y commit=y log='/tmp/dbausr.log'
streams_instantiation=y fromuser=dbausr touser=dbausr

```

5.2 Another way: Duplicate by using data pump tool: DBMS_DATAPUMP

Refer Streams Replication Administrator's Guide (10.2) chp20 Multiple-Source Replication Example

6 Create propagation/apply rules

6.1 Create propagation rules

```

-- strm1: set propagation rule
begin
dbms_streams_adm.add_schema_propagation_rules(
schema_name => 'dbausr',
streams_name => 'strm1_to_strm2',
source_queue_name => 'strmadmin.strm1_cap_queue',
destination_queue_name => 'strmadmin.strm2_apl_queue@strm2.com',
include_dml => true,
include_ddl => true,
include_tagged_lcr => false,
source_database => 'strm1.com',
inclusion_rule => true);
end;
/

-- strm2: set propagation rule
begin
dbms_streams_adm.add_schema_propagation_rules(
schema_name => 'dbausr',
streams_name => 'strm2_to_strm1',
source_queue_name => 'strmadmin.strm2_cap_queue',
destination_queue_name => 'strmadmin.strm1_apl_queue@strm1.com',
include_dml => true,
include_ddl => true,

```

```

include_tagged_lcr => false,
source_database => 'strm2.com',
inclusion_rule => true);
end;
/

-- change the latency to 0, which means real time propagation
begin
dbms_aqadm.alter_propagation_schedule(
queue_name => 'strm1_cap_queue',
destination => 'strm2',
latency => 0);
end;
/

```

6.2 Create apply process in both side

```

-- strm2 : set init SCN
DECLARE
iscn NUMBER; -- Variable to hold instantiation SCN value
BEGIN
iscn := dbms_flashback.get_system_change_number();
dbms_apply_adm.set_schema_instantiation_scn@strm1.com(
source_schema_name => 'DBAUSR',
source_database_name => 'strm2.com',
instantiation_scn => iscn);
END;
/

-- strm2 : set apply rule
begin
dbms_streams_adm.add_schema_rules(
schema_name => 'dbausr',
streams_type => 'apply',
streams_name => 'strm2_apply',
queue_name => 'strmadmin.strm2_apl_queue',
include_dml => true,
include_ddl => true,
include_tagged_lcr => false,
source_database => 'strm1',
inclusion_rule => true);
end;
/

```

```

-- strm1 : set init SCN
DECLARE
iscn NUMBER; -- Variable to hold instantiation SCN value
BEGIN
iscn := dbms_flashback.get_system_change_number();
dbms_apply_adm.set_schema_instantiation_scn @strm2.com(
source_schema_name => 'DBAUSR',
source_database_name => 'strm1.com',
instantiation_scn => iscn);
END;
/

-- strm1 : set apply rule
begin
dbms_streams_adm.add_schema_rules(
schema_name => 'dbausr',
streams_type => 'apply',
streams_name => 'strm1_apply',
queue_name => 'strmadmin.strm1_apl_queue',
include_dml => true,

```

```

include_ddl => true,
include_tagged_lcr => false,
source_database => 'strm2',
inclusion_rule => true);
end;
/

```

7. Start/stop streams and verificating its functionality

7.1 Start streams

```

-- strm1 : start capture
begin
dbms_capture_adm.start_capture(
capture_name => 'strm1_capture');
end;
/
-- start apply
begin
dbms_apply_adm.start_apply(
apply_name => 'strm1_apply');
end;
/
select capture_name,status from dba_capture;
select propagation_name,status from dba_propagation;
select apply_name,status from dba_apply;

```

```

-- strm2: start apply
begin
dbms_apply_adm.start_apply(
apply_name => 'strm2_apply');
end;
/

```

```

-- strm2: start capture
begin
dbms_capture_adm.start_capture(
capture_name => 'strm2_capture');
end;
/
select capture_name,status from dba_capture;
select propagation_name,status from dba_propagation;
select apply_name,status from dba_apply;

```

7.2 Verification

```

-- [DDL Test]
-- strm1
strmadmin@STRM1> create table dbausr.aaa (id number(3));

```

```

-- strm2 :
strmadmin@STRM2> desc dbausr.aaa
Name                               Null?  Type
-----
ID                                   NUMBER(3)
strmadmin@STRM2> alter table dbausr.aaa modify (id
number(4));

```

```

-- strm1
strmadmin@STRM1> desc dbausr.aaa
Name                               Null?  Type
-----
ID                                   NUMBER(4)

```

```

-- [DML Test]
strmadmin@STRM1> insert into dbausr.aaa values (1111);
strmadmin@STRM1> commit;

```

```

-- strm2 :

```

```
strmadmin@STRM2> select * from dbausr.aaa;
      ID
-----
      1111
strmadmin@STRM2> update dbausr.aaa set id=id*2;
strmadmin@STRM2> commit;
```

```
-- strm1
strmadmin@STRM1> select * from dbausr.aaa;
      ID
-----
      2222
```

7.3 Stop streams

```
-- stop capture
begin
dbms_capture_adm.stop_capture(
capture_name => 'strm1_capture');
end;
/
select capture_name,status from dba_capture;

-- stop propagation
begin
dbms_propagation_adm.stop_propagation(
propagation_name => 'strm1_to_strm2',
force => false);
end;
/
select propagation_name,status from dba_propagation;

-- stop apply
begin
dbms_apply_adm.stop_apply(
apply_name => 'strm2_apply');
end;
/
select apply_name,status from dba_apply;
```

8. Remove and clear streams environment

```
-- strm1
exec DBMS_STREAMS_ADM.remove_streams_configuration();
-- strm2
exec DBMS_STREAMS_ADM.remove_streams_configuration();
```

Note: If you want remove particular capture/propagation/apply process,
use dbms_capture(/propagation/apply) _adm.drop_capture(/propagation/apply)

```
begin
dbms_capture_adm.drop_capture(
capture_name => '<name>',
drop_unused_rule_sets => true);
end;
/

begin
dbms_propagation_adm.drop_propagation(
propagation_name => '<name>',
drop_unused_rule_sets => true);
end;
/

begin
```

```
dbms_apply_adm.drop_apply(  
  apply_name => '<name>',  
  drop_unused_rule_sets => true);  
end;  
/
```