Oracle GoldenGate A-Team

Oracle GoldenGate Best Practices: Heartbeat Table for Monitoring Lag times

Document ID: 1299679.1

Version 11.0

Date: March 21, 2013

Steve George
Consulting Member of Technical Staff
Fusion Middleware Architects Team: The A-Team

Table of Contents

1 Document Acceptance

1.1 Contributors:

Name	Title
Mark Geisler	Solutions Consultant
Joe DeBuzna	Solutions Consultant
Sjaak Vossepoel	Team leader, Solutions Consultants EMEA
Herman Verheul	Technical Specialist Expert Services Europe, Middle East & Africa GoldenGate, B.V

1.2 Reviewers:

Reviewer	Title
Sjaak Vossepoel	Team Leader, Solutions Consultants EMEA

1.3 Change Control:

Date	Change	Comment
5-28-2008	Updated configurations –V2	Sjaak Vossepoel
10-30-2008	Updated to support Data Guard	Steve George
4-17-2009	Updated to v5	Steve George
4-28-09	Updated to v6 – added more comments on column mapping and added defgen configuration	Steve George
5-6-09	Added changes from Herman to tables to show lag times up to the microseconds	Steve George
3-11-2010	Updated for Oracle	Steve George
3-17-2010	Remove dependency on source def file.	Tracy West
3-19-2010	Updated for sub-second reporting only, removed need for sourcedef file	Steve George
3-26-2010	Updated some column descriptions. Added a second table – heartbeat history.	Steve George
2-11-2011	Updated document with example parameter files	Steve George
9-12-2012	Updated heartbeat to use DBMS Jobs scheduler and simplify configuration and use.	Steve George
11-20-2012	Added DML and DDL Stats to Heartbeat	Steve George
1-2-2013	Cleaned up scripts	Steve George
3/21/2013	Simplified scripts and config.	Steve George

Caution

This sample code is provided for educational purposes only and not supported by Oracle Support Services. It has been tested internally, however, and works as documented. We do not guarantee that it will work for you, so be sure to test it in your environment before relying on it.

Proofread this sample code before using it! Due to the differences in the way text editors, e-mail packages and operating systems handle text formatting (spaces, tabs and carriage returns), this sample code may not be in an executable state when you first receive it. Check over the sample code to ensure that errors of this type are corrected.

2 Overview

This document is intended to help you implement the best practice for creating a heartbeat process that can be used to determine where and when lag is developing between a source and target system.

This document will walk you thru the step-by-step process of creating the necessary tables and added table mapping statements needed to keep track of processing times between a source and target database. Once the information is added into the data flow, the information is then stored into a target tables that can be analyzed to determine when and when the lag is being introduced between the source and target systems.

The heartbeat tables can be used to identify the following information:

- Where lag is developing in each of the processes
- When lag is developing by comparing the commit timestamp with the current time that the record pass thru each process.

These tables also allow you to:

- Create a history of lag to determine what time of day lag develops.
- Create a history to identify if lag is increasing over time.
- Identify if a upstream process is stopped for any reason.
- Monitor DML and DDL statistics

3 Configuration

3.1 Overview

GoldenGate heartbeat implementation requires the following application and database modifications.

1. Add a Heartbeat table to the Source GoldenGate database schema
2. Add a Heartbeat status table and a history table to the target database
3. Add mapping statement to each process
4. Create a DBMS scheduler job on the source database to update the heartbeat table.

The source and target heartbeat tables are all the same structure. In the source system, the heartbeat is a single row update of the timestamp. The extract process will extract the updated information and add a couple of tokens to the record. This information is written in the trail for the following process to read and add additional information to the record as it passes through the data pump and then into the target.

On the target system the first record will be inserted into the heartbeat table and again inserted into a history table. Following records are then updated in the status table and inserted into the history table. When the record is inserted, additional information will be added to the columns in the table via tokens. A trigger is added to the heartbeat tables to automatically calculate the lag for each record. For this example, the target will include information from the source, information on the data pump, if used, and target replicat information.

The following information will help you determine where and when the lag time is developing.

3.2 Source and Target users

Both the source and target will need a schema in order to create the source and target tables. It is recommended that you use the GoldenGate user/schema that you have already set up as part of the base GoldenGate configuration. Note: if this is a bidirectional configuration where the GoldenGate user is excluded, you will need to create the source heartbeat table in a different schema. The target tables can be under the GoldenGate schema.

On Source system

```
SQL> create user source identified by ggs;
SQL> grant connect, resource, dba to source;
```

One Target system

```
SQL> create user target identified by ggs;
SQL> grant connect, resource, dba to target;
```

3.3 Heartbeat tables – Source and Target

In this configuration we are using three heartbeat tables, HEARTBEAT, GGS_HEARTBEAT and GGS_HEARTBEAT_HISTORY. All of the tables have the same column layout but are used for different information. The HEARTBEAT table may have more than one row in it, depending on the number of threads (RAC) and the only columns that are updated are the update timestamp and the source DB name. The on the target system the GGS_HEARTBEAT table contains the current (last update) heartbeat for all of the replicats that are mapping into the heartbeat. The last table is the GGS_HEARTBEAT_HISTORY table.

All of heartbeat records are inserted into the history table. This table can be used to chart lag for a process over a period of time. In a production environment you may want to partition this table to ease maintenance of this table.

This is the layout for all of the heartbeat tables.

Column	Contents
ID	Sequence number
SRC_DB	Source Database Name
Extract_name	Name of the extract process from token.
Source_commit	Source commit timestamp from header.
Target_commit	When the record was added to the target. This is updated by the trigger.
CAPTIME	Added as a token using DATANOW() function.
CAPLAG	Capture time – commit time on the source. This is updated by the trigger
PMPTIME	When token timestamp was added to trail in the data pump using DATENOW()
PMPGROUP	Data pump group name.
PMPLAG	Capture time minus the time that the record was passed thru the data pump. Value is calculated by the update trigger.
DELTIME	This is added as part of the map statement using DATENOW().
DELGROUP	Name of the replicat group.
DELLAG	The defference between the time the record was passed thry the data pump and when the record was inserted into the target table. This is calculated in the trigger.
TOTALLAG	The difference between the target commit time and the source commit time. This is updated via the trigger.
Thread	Thread number from instance.
Update_timestamp	The system time of the update into the HB table.
EDDLDELTASTATS	DDL operations since the last gathering (Delta) of stats on the extract process.
EDMLDELTASTATS	DML operations since the last gathering (Delta) of stats on the extract process.
RDDLDELTASTATS	DDL operations since the last gathering (Delta) of stats on the replicat process.
RDMLDELTASTATS	DML operations since the last gathering (Delta)

	of stats on the replicat process.

To create the tables you will need to run a sqlplus script to create the heartbeat tables and add a row to the heartbeat table -

```
SQL> @<path>/heartbeat_tables_v11.sql
```

The next step is to add trandata to the source heartbeat table using the GGSCI command interface -

```
GGSCI> ADD TRANDATA SOURCE.HEARTBEAT
```

3.4 Heartbeat tables - Target

As the heartbeat data flows through the different process, each process adds tokens to the record so that you can track the lag time from the source system to the target. The first table, GGS_HEARTBEAT, has only one record for each replicat process that is feeding into it. If you have ten replicats you should have ten rows in this table. The main idea of this table is that if you want to look at the current lag times of your replicats, you only need to look at one small table. The second table is the GGS_HEARTBEAT_HISTORY table. As its name in implies, it is a history of all of the heartbeats. This can be used to determine if and when you have had lag in the past. Because this is an "insert all records" table, it is a good idea to partition the table so that you can manage its size over time.

A trigger on the target heartbeat tables does the calculations when the record is inserted or updated on the target. Lag times expressed in microseconds

NOTE: The system clocks across all system must be synchronized. If not, then the calculated lag times will be inaccurate. If you see negative times, check the clocks.

```
CREATE OR REPLACE TRIGGER GGS_HEARTBEAT_TRIG
BEFORE INSERT OR UPDATE ON GGS_HEARTBEAT
FOR EACH ROW
BEGIN
select seq_ggs_HEARTBEAT_id.nextval
into :NEW.ID
from dual;
select systimestamp
into :NEW.target_COMMIT
from dual;
select trunc(to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT ),1,
instr(:NEW.CAPTIME - :NEW.SOURCE_COMMIT,' '))) * 86400
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
```

```
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.CAPLAG
from dual;
select trunc(to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME),1, instr(:NEW.PMPTIME -
:NEW.CAPTIME,' ')))) * 86400
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+7,2))
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+10,6)) / 1000000
into :NEW.PMPLAG
from dual;
select trunc(to_number(substr((:NEW.DELTIME - :NEW.PMPTIME),1, instr(:NEW.DELTIME -
:NEW.PMPTIME,' ')))) * 86400
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+7,2))
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+10,6)) / 1000000
into :NEW.DELLAG
from dual;
select trunc(to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),1,
instr(:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT,' ')))) * 86400
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.TOTALLAG
from dual;
end ;
/

ALTER TRIGGER "GGS_HEARTBEAT_TRIG" ENABLE;
```

3.5 Updating of the heartbeat table

The heartbeat table is updated via a stored procedure that it executed by a DBMS_JOB using the scheduler.

This is the DBMS_SCHEDULER command to create the job. It may be easier to create the job using a tool like OEM or SQL Developer. The key to using DBMS_SCHEDULER is the ability to repeat the task at a predefined interval. In this example "repeat_interval" is set to every minute.

SQL> @HB_DBMS_SCHEDULER.sql

```
-- connect / as sysdba

accept ogg_user prompt 'GoldenGate User name:'

grant select on v_$instance to &&ogg_user;

grant select on v_$database to &&ogg_user;

BEGIN
    SYS.DBMS_SCHEDULER.DROP_JOB(job_name => '&&ogg_user..OGG_HB',
                                defer => false,
                                force => false);
END;
/

CREATE OR REPLACE PROCEDURE &&ogg_user..gg_update_hb_tab IS
v_thread_num     NUMBER;
v_db_unique_name VARCHAR2 (128);
BEGIN
SELECT db_unique_name
INTO  v_db_unique_name
FROM v$database;

UPDATE &&ogg_user..heartbeat
SET update_timestamp = SYSTIMESTAMP
,src_db = v_db_unique_name;
COMMIT;
END;
/

BEGIN
SYS.DBMS_SCHEDULER.CREATE_JOB (
job_name => '&&ogg_user..OGG_HB',
job_type => 'STORED_PROCEDURE',
job_action => '&&ogg_user..GG_UPDATE_HB_TAB',
number_of_arguments => 0,
start_date => NULL,
repeat_interval => 'FREQ=MINUTELY',
end_date => NULL,
job_class => '"SYS"."DEFAULT_JOB_CLASS"',
enabled => FALSE,
auto_drop => FALSE,
comments => 'GoldenGate',
credential_name => NULL,
destination_name => NULL);
```

```
SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
name => '&&ogg_user..OGG_HB',
attribute => 'restartable', value => TRUE);


SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
name => '&&ogg_user..OGG_HB',
attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);

SYS.DBMS_SCHEDULER.enable(
name => '&&ogg_user..OGG_HB');
END;
/
```

You can use the following sql to check the status of the job –

```
col REPEAT_INTERVAL format a15
col NEXT_RUN_DATE format a38
col OWNER format a10
col JOB_NAME format a8
set linesize 120
select
   owner,
   job_name,
   job_class,
   enabled,
   next_run_date,
   repeat_interval
from
   dba_scheduler_jobs
where
   owner = decode(upper('&&ogg_user'), 'ALL', owner, upper('&&ogg_user'))
;
```

3.6 Extract Configuration

In the extract parameter file a TABLE statement will need to be added in order to capture the update to the heartbeat table. Along with the update, a couple of tokens also need to be added in order to tell which extract and host the data originated from.

In this example you would add the extract using the following commands –

```
ADD EXTRACT ext_hb, TRANLOG, BEGIN NOW, threads 1
ADD EXTTRAIL ./dirdat/~Primary_trail~, EXTRACT ext_hb, MEGABYTES 100
```

Where you would substitute ~threads~ for the number of threads (RAC) and ~Primary_trail~ for the trail name you want to use.

Here is the include file with the heartbeat map statement for the heartbeat table –

Include file –
./dirprm/HB_Extract.inc

```
-- HB_Extract.inc
-- Heartbeat Table
-- update 9-1-12 SGEORGE - no checkpoint info.
TABLE <source schema>.HEARTBEAT,
      TOKENS (
      CAPGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
      CAPTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
      ("JULIANTIMESTAMP")),
      EDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
      EDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
      );
```

This is example of a complete extract parameter file with the include file for the heartbeat –

```
EXTRACT ext_hb
SETENV (ORACLE_SID=ora11g)
-- Use USERID to specify the type of database authentication for GoldenGate to
use.
USERID source password ggs
EXTTRAIL ./dirdat/db
-- Use DISCARDFILE to generate a discard file to which Extract or Replicat can
log
-- records that it cannot process. GoldenGate creates the specified discard
file in
-- the dirrpt sub-directory of the GoldenGate installation directory. You can
use the
-- discard file for problem-solving.
DISCARDFILE ./dirrpt/ext_hb.dsc, APPEND
-- Use REPORTCOUNT to generate a count of records that have been processed
since
-- the Extract or Replicat process started
-- REPORTCOUNT [EVERY] <count> {RECORDS | SECONDS | MINUTES | HOURS} [, RATE]
REPORTCOUNT EVERY 5 MINUTES, RATE
-- Use FETCHOPTIONS to control certain aspects of the way that GoldenGate
fetches
FETCHOPTIONS, NOUSESNAPSHOT, NOUSELATESTVERSION, MISSINGROW REPORT
-- Use STATOPTIONS to specify information to be included in statistical
displays
-- generated by the STATS EXTRACT or STATS REPLICAT command.
STATOPTIONS REPORTFETCH
-- This is the Heartbeat table
include dirprm/HB_Extract.inc

-- The implementation of this parameter varies depending on the process.
-- TABLE source.*;
```

3.7 Data Pump Configuration

In the Data Pump parameter file you will need to include a TABLE statement so that the pump name and current timestamp are added to the record as the record is passed through the data pump.

Note: It is best to have the heartbeat table in a different schema than your application. If you wildcard the schema and have a TABLE statement for the heartbeat table, you will end up with duplicate records in the output trail.

Here are the commands to add the datapump and remote trail –

```
ADD EXTRACT pmp_hb, BEGIN NOW, EXTTRAILSOURCE ./dirdat/~Primary_trail~
ADD RMTTRAIL ./dirdat/~Target_trail~, EXTRACT pmp_hb, MEGABYTES 100
```

You will need to substitute `~Primary_trail~` and `~Target_trail~` for the trail names you want to use.

The include file for the heartbeat in the data pump is as follows:

./dirprm/HB_pmp.inc

```
-- HB_pmp.inc
-- Heartbeat Table
table <source schema>.heartbeat,
   TOKENS (
             PMPGROUP = @GETENV ("GGENVIRONMENT","GROUPNAME"),
             PMPTIME = @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP"))
          );
```

Example of a complete Data Pump parameter file –

```
-- Data Pump configuration file
-- last update
-- 11-3-08 SGEORGE
-- update 9-1-12 SGEORGE - no checkpoint info.
--
extract PMP_hb
-- Database login info
userid ~ogg_user~, password ~ogg_pass~

-- Just in case we can't process a record we'll dump info here
discardfile ./dirrpt/PMP_hb.dsc, append

-- Remote host and remort manager port to write trail
rmthost ~target_host~, mgrport ~Target_mgr~

-- This is the Trail to where we output
rmttrail ./dirdat/~Target_trail~
-- Heartbeat
include dirprm/HB_pmp.inc

Table  ~source_schema~.*;
```

3.8 Replicat Configuration

The replicat will need to have the heartbeat table added to the map statements along with the token mapping. When the replicat inserts the row into the table a “before insert” trigger will fire and update the values in the GGS_HEARTBEAT table.

There are two heartbeat tables, the first is the heartbeat table that has the current heartbeat information. It will have only one row for each replicat. The second table is the history table that contains all of the heartbeats records. This table can be used to graph the lag time in each replicat end to end.

As with the extract and data pump, we are adding data to the record when we insert the row into the target heartbeat table.

Here is the command to add the replicat –

```
add replicat REP_HB exttrail ./dirdat/~Rep_PR_Trail~ nodbcheckpoint
```

This is the include file for the Map statement:

./dirprm/HB_Rep.inc

```
dirprm/HB_Rep.inc
-- Heartbeat table
MAP <source schema>.HEARTBEAT, TARGET <target schema>.GGS_HEARTBEAT,
KEYCOLS (DELGROUP),
INSERTMISSINGUPDATES,
COLMAP (USEDEFAULTS,
ID = 0,
SOURCE_COMMIT = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
EXTRACT_NAME = @TOKEN ("CAPGROUP"),
CAPTIME = @TOKEN ("CAPTIME"),
PMPGROUP = @TOKEN ("PMPGROUP"),
PMPTIME = @TOKEN ("PMPTIME"),
DELGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
DELTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP")),
EDDLDELTASTATS = @TOKEN ("EDDLDELTASTATS"),
EDMLDELTASTATS = @TOKEN ("EDMLDELTASTATS"),
RDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
RDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
);

MAP <source schema>.HEARTBEAT, TARGET <target schema>.GGS_HEARTBEAT_HISTORY,
KEYCOLS (ID),
INSERTALLRECORDS,
COLMAP (USEDEFAULTS,
ID = 0,
SOURCE_COMMIT = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
EXTRACT_NAME = @TOKEN ("CAPGROUP"),
CAPTIME = @TOKEN ("CAPTIME"),
PMPGROUP = @TOKEN ("PMPGROUP"),
PMPTIME = @TOKEN ("PMPTIME"),
```

```
      DELGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
      DELTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
      ("JULIANTIMESTAMP")),
      EDDLDELTASTATS = @TOKEN ("EDDLDELTASTATS"),
      EDMLDELTASTATS = @TOKEN ("EDMLDELTASTATS"),
      RDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
      RDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
      );
```

This is an example of a complete Replicat parameter file:

```
replicat rep_hb
-- Use ASSUMETARGETDEFS when the source and target tables specified with a MAP
-- statement have the same column structure, such as when synchronizing a hot
-- site DO NOT USE IF YOU USE THE COLMAP Statement. USE Sourcedef file.
assumetargetdefs
--setting oracle_environment variable
--useful in  multi oracle home situations
--setenv (ORACLE_HOME="/u01/app/oracle/product/11.2.0/db112")
setenv (ORACLE_SID="db112r1")
--userid password password encrypted using encrypt ggsci command
userid SOURCE,password GGS

-- Just in case we can't process a record we'll dump info here
discardfile ./dirrpt/REP_hb.dsc, append

-- Use REPORTCOUNT to generate a count of records that have been processed since
-- the Extract or Replicat process started
-- REPORTCOUNT [EVERY] <count> {RECORDS | SECONDS | MINUTES | HOURS} [, RATE]
REPORTCOUNT EVERY 5 MINUTES, RATE
include ./dirprm/HB_Rep.inc
map <SOURCE_SCHEMA>.* ,target <TARGET_SCHEMA>.*;
```

4 Conclusion

In order to calculate the true lag you will need to add the heartbeat table into the extracts (extract and data pump) and replicats. By using the tokens that are added to the trail and the commit time on the target you can tell the true lag between systems even with low data flow. Also using this method you can tell on the target if the data flow from the source has been interrupted because you can check the last update time and compare that to the current time. It is critical that clocks on both the source and target systems are in sync. Note, OGG does correct the commit timestamp for differences between the source and target systems.

5 Appendix A – Parameter and Sql Scripts - Examples

You will need to edit the following scripts and manually change the variables to the correct values for your system. You may need to change to the User that you are installing the scripts under for each system. In the example I use the users "SOURCE" and "TARGET", you will need to update based on your configuration. You will need to run the same scripts for the source and target, but the variables may be different between the two systems.

5.1 del_lag.sql

```
set pagesize 200
col "Total Lag" format a30
col "Extract Lag" format a30
col "Pump Lag" format a30
select DELGROUP,
(SOURCE_COMMIT - CAPTIME ) "Extract Lag",
(SOURCE_COMMIT - PMPTIME ) "Pump Lag",
(SOURCE_COMMIT - TARGET_COMMIT ) "Total Lag"
from target.ggs_heartbeat_history order by id;
```

5.2 Ext_hb.prm

```
-- Extract example for Heartbeat
-- 4-9-10 SGEORGE
--
EXTRACT ext_hb
SETENV (ORACLE_SID=db112r1)
-- Use USERID to specify the type of database authentication for GoldenGate to
use.
USERID SOURCE password GGS
TRANLOGOPTIONS DBLOGREADER
EXTTRAIL ./dirdat/HB
-- Use DISCARDFILE to generate a discard file to which Extract or Replicat can
log
-- records that it cannot process. GoldenGate creates the specified discard
file in
-- the dirrpt sub-directory of the GoldenGate installation directory. You can
use the
-- discard file for problem-solving.
DISCARDFILE ./dirrpt/ext_hb.dsc, APPEND
-- Use REPORTCOUNT to generate a count of records that have been processed
since
-- the Extract or Replicat process started
```

```
-- REPORTCOUNT [EVERY] <count> {RECORDS | SECONDS | MINUTES | HOURS} [, RATE]
REPORTCOUNT EVERY 5 MINUTES, RATE
-- Use FETCHOPTIONS to control certain aspects of the way that GoldenGate
fetches
FETCHOPTIONS, NOUSESNAPSHOT, NOUSELATESTVERSION, MISSINGROW REPORT
-- Use STATOPTIONS to specify information to be included in statistical
displays
-- generated by the STATS EXTRACT or STATS REPLICAT command.
STATOPTIONS REPORTFETCH
-- This is the Heartbeat table
include dirprm/HB_Extract.inc

-- The implementation of this parameter varies depending on the process.
-- TABLE source.*;
```

5.3 HB_DBMS_SCHEDULER.sql

```
-- connect / as sysdba

accept ogg_user prompt 'GoldenGate User name:'

grant select on v_$instance to &&ogg_user;

grant select on v_$database to &&ogg_user;

BEGIN
    SYS.DBMS_SCHEDULER.DROP_JOB(job_name => '&&ogg_user..OGG_HB',
                                defer => false,
                                force => false);
END;
/

CREATE OR REPLACE PROCEDURE &&ogg_user..gg_update_hb_tab IS
v_thread_num     NUMBER;
v_db_unique_name VARCHAR2 (128);
BEGIN
SELECT db_unique_name
INTO  v_db_unique_name
FROM v$database;

UPDATE &&ogg_user..heartbeat
SET update_timestamp = SYSTIMESTAMP
,src_db = v_db_unique_name;
COMMIT;
END;
/

BEGIN
SYS.DBMS_SCHEDULER.CREATE_JOB (
job_name => '&&ogg_user..OGG_HB',
job_type => 'STORED_PROCEDURE',
job_action => '&&ogg_user..GG_UPDATE_HB_TAB',
number_of_arguments => 0,
```

```
start_date => NULL,
repeat_interval => 'FREQ=MINUTELY',
end_date => NULL,
job_class => '"SYS"."DEFAULT_JOB_CLASS"',
enabled => FALSE,
auto_drop => FALSE,
comments => 'GoldenGate',
credential_name => NULL,
destination_name => NULL);


SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
name => '&&ogg_user..OGG_HB',
attribute => 'restartable', value => TRUE);


SYS.DBMS_SCHEDULER.SET_ATTRIBUTE(
name => '&&ogg_user..OGG_HB',
attribute => 'logging_level', value => DBMS_SCHEDULER.LOGGING_OFF);

SYS.DBMS_SCHEDULER.enable(
name => '&&ogg_user..OGG_HB');
END;
/
```

5.4 HB_Extract.inc

```
-- HB_Extract.inc
-- Heartbeat Table
-- update 9-1-12 SGEORGE - no checkpoint info.
TABLE SOURCE.HEARTBEAT,
TOKENS (
CAPGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
CAPTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP")),
EDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
EDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
);
```

5.5 HB_pmp.inc

```
-- HB_pmp.inc
-- Heartbeat Table
table SOURCE.heartbeat,
TOKENS (
PMPGROUP = @GETENV ("GGENVIRONMENT","GROUPNAME"),
```

```
PMPTIME = @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP"))
);
```

5.6 HB_Rep.inc

```
-- Heartbeat table
MAP target.HEARTBEAT, TARGET SOURCE.GGS_HEARTBEAT,
KEYCOLS (DELGROUP),
INSERTMISSINGUPDATES,
COLMAP (USEDEFAULTS,
ID = 0,
SOURCE_COMMIT = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
EXTRACT_NAME = @TOKEN ("CAPGROUP"),
CAPTIME = @TOKEN ("CAPTIME"),
PMPGROUP = @TOKEN ("PMPGROUP"),
PMPTIME = @TOKEN ("PMPTIME"),
DELGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
DELTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP")),
EDDLDELTASTATS = @TOKEN ("EDDLDELTASTATS"),
EDMLDELTASTATS = @TOKEN ("EDMLDELTASTATS"),
RDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
RDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
);

MAP target.HEARTBEAT, TARGET SOURCE.GGS_HEARTBEAT_HISTORY,
KEYCOLS (ID),
INSERTALLRECORDS,
COLMAP (USEDEFAULTS,
ID = 0,
SOURCE_COMMIT = @GETENV ("GGHEADER", "COMMITTIMESTAMP"),
EXTRACT_NAME = @TOKEN ("CAPGROUP"),
CAPTIME = @TOKEN ("CAPTIME"),
PMPGROUP = @TOKEN ("PMPGROUP"),
PMPTIME = @TOKEN ("PMPTIME"),
DELGROUP = @GETENV ("GGENVIRONMENT", "GROUPNAME"),
DELTIME =  @DATE ("YYYY-MM-DD HH:MI:SS.FFFFFF","JTS",@GETENV
("JULIANTIMESTAMP")),
EDDLDELTASTATS = @TOKEN ("EDDLDELTASTATS"),
EDMLDELTASTATS = @TOKEN ("EDMLDELTASTATS"),
RDDLDELTASTATS = @GETENV ("DELTASTATS", "DDL"),
RDMLDELTASTATS = @GETENV ("DELTASTATS", "DML")
);
```

5.7 HB_Stop.sql

```
set verify off
set linesize 200
set pagesize 80
col OWNER format a15
col JOB_NAME format a20
col JOB_CLASS format a20
col NEXT_RUN_DATE format a40
col REPEAT_INTERVAL format a50

accept ogg_user prompt 'GoldenGate User name:'

select
owner,
job_name,
job_class,
enabled,
next_run_date,
repeat_interval
from
dba_scheduler_jobs
where
owner = decode(upper('&&ogg_user'), 'ALL', owner, upper('&&ogg_user'))
;

COL ID               format 999,999
COL SRC_DB           format a10
COL EXTRACT_NAME     format a8
COL SOURCE_COMMIT    format a28
COL TARGET_COMMIT    format a28
COL CAPTIME          format a28
COL CAPLAG           format 999.000
COL PMPTIME          format a28
COL PMPGROUP         format a8
COL PMPLAG           format 999.000
COL DELTIME          format a28
COL DELGROUP         format a8
COL DELLAG           format 999.000
COL TOTALLAG         format 999.000
COL THREAD           format 99
COL UPDATE_TIMESTAMP format a28

select * from &&ogg_user.heartbeat;

BEGIN
SYS.DBMS_SCHEDULER.DROP_JOB(job_name => '"&&ogg_user"."OGG_HB"',
defer => false,
force => true);
END;
/
```

```
select
owner,
job_name,
job_class,
enabled,
next_run_date,
repeat_interval
from
dba_scheduler_jobs
where
owner = decode(upper('&&ogg_user'), 'ALL', owner, upper('&&ogg_user'))
;
```

5.8 HB_table.sql

```
set pagesize 200
col Lag format a30
col SOURCE_COMMIT  format a30
col TARGET_COMMIT  format a30
col CAPTIME  format a30
col PMPTIME  format a30
col DELTIME  format a30
col START_TIME format a30
col RECOVERY_TIMESTAMP format a30
col UPDATE_TIMESTAMP   format a30

accept ogg_user prompt 'GoldenGate User name:'

select * from &&ogg_user.ggs_heartbeat;
```

5.9 heartbeat_check.sql

```
set echo off
set concat ,
set verify off

COL ID               format 999,999
COL SRC_DB           format a10
COL EXTRACT_NAME     format a8
COL SOURCE_COMMIT    format a28
COL TARGET_COMMIT    format a28
COL CAPTIME          format a28
COL CAPLAG           format 999.000
COL PMPTIME          format a28
COL PMPGROUP         format a8
COL PMPLAG           format 999.000
COL DELTIME          format a28
COL DELGROUP         format a8
```

```
COL DELLAG           format 999.000
COL TOTALLAG         format 999.000
COL THREAD           format 99
COL UPDATE_TIMESTAMP format a28

accept ogg_user prompt 'GoldenGate User name:'

select * from &&ogg_user.GGS_HEARTBEAT;


select src_db, extract_name, Caplag, pmplag, dellag, totallag, UPDATE_TIMESTAMP
from &&ogg_user.GGS_HEARTBEAT;
```

5.10 heartbeat_tables_v11.sql

This script is used to create the heartbeat tables. In environments that are only running OGG in one direction, the GGS_Heartbeat and the GGS_HEARTBEAT_HISTORY tables only need to be created on the target system.
In bi-directional configurations you will need all tables on both sides.

```
-- Heartbeat table V11
-- This is created on the SOURCE system
-- Last update
-- 10-30-08 SGEORGE
-- 11-25-08 Table updated to match target.
--        PK is different on source.
-- 3-19-10 SGEORGE - changed format, updated for timestamp.
-- 9-1-12  SGEORGE - Updated HB table for Bi-directional
-- 3-4-13 SGEORGE - Updated script to prompt for schema name.

accept ogg_user prompt 'GoldenGate User name:'

drop table &&ogg_user..heartbeat;
-- Create table statement
CREATE TABLE &&ogg_user..HEARTBEAT
(     ID NUMBER ,
SRC_DB           VARCHAR2(30),
EXTRACT_NAME     varchar2(8),
SOURCE_COMMIT    TIMESTAMP,
TARGET_COMMIT    TIMESTAMP,
CAPTIME     TIMESTAMP,
CAPLAG           NUMBER,
PMPTIME     TIMESTAMP,
PMPGROUP    VARCHAR2(8 BYTE),
PMPLAG           NUMBER,
DELTIME     TIMESTAMP,
DELGROUP    VARCHAR2(8 BYTE),
DELLAG           NUMBER,
TOTALLAG    NUMBER,
thread           number,
```

```
update_timestamp timestamp,
EDDLDELTASTATS   number,
EDMLDELTASTATS   number,
RDDLDELTASTATS   number,
RDMLDELTASTATS   number,
CONSTRAINT HEARTBEAT_PK PRIMARY KEY (SRC_DB) ENABLE
)
/

-- this assumes that the table is empty
INSERT INTO &&ogg_user..HEARTBEAT (SRC_DB) select db_unique_name from
V$database;
commit;


DROP SEQUENCE &&ogg_user..SEQ_GGS_HEARTBEAT_ID ;
CREATE SEQUENCE &&ogg_user..SEQ_GGS_HEARTBEAT_ID INCREMENT BY 1 START WITH 1
ORDER ;
DROP TABLE &&ogg_user..GGS_HEARTBEAT;
CREATE TABLE &&ogg_user..GGS_HEARTBEAT
(     ID NUMBER ,
SRC_DB           VARCHAR2(30),
EXTRACT_NAME     varchar2(8),
SOURCE_COMMIT    TIMESTAMP,
TARGET_COMMIT    TIMESTAMP,
CAPTIME    TIMESTAMP,
CAPLAG           NUMBER,
PMPTIME    TIMESTAMP,
PMPGROUP   VARCHAR2(8 BYTE),
PMPLAG           NUMBER,
DELTIME    TIMESTAMP,
DELGROUP   VARCHAR2(8 BYTE),
DELLAG           NUMBER,
TOTALLAG   NUMBER,
thread           number,
update_timestamp timestamp,
EDDLDELTASTATS   number,
EDMLDELTASTATS   number,
RDDLDELTASTATS   number,
RDMLDELTASTATS   number,
CONSTRAINT GGS_HEARTBEAT_PK PRIMARY KEY (DELGROUP) ENABLE
);

CREATE OR REPLACE TRIGGER &&ogg_user..GGS_HEARTBEAT_TRIG
BEFORE INSERT OR UPDATE ON &&ogg_user..GGS_HEARTBEAT
FOR EACH ROW
BEGIN
select seq_ggs_HEARTBEAT_id.nextval
into :NEW.ID
from dual;
select systimestamp
into :NEW.target_COMMIT
from dual;
```

```
select trunc(to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT ),1,
instr(:NEW.CAPTIME - :NEW.SOURCE_COMMIT,' '))) * 86400
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.CAPLAG
from dual;
select trunc(to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME),1,
instr(:NEW.PMPTIME - :NEW.CAPTIME,' '))) * 86400
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+7,2))
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+10,6)) / 1000000
into :NEW.PMPLAG
from dual;
select trunc(to_number(substr((:NEW.DELTIME - :NEW.PMPTIME),1,
instr(:NEW.DELTIME - :NEW.PMPTIME,' '))) * 86400
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+7,2))
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+10,6)) / 1000000
into :NEW.DELLAG
from dual;
select trunc(to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),1,
instr(:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT,' '))) * 86400
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.TOTALLAG
from dual;
end ;
/

ALTER TRIGGER &&ogg_user..GGS_HEARTBEAT_TRIG ENABLE;

--
```

```
-- This is for the History heartbeat table
--
DROP SEQUENCE &&ogg_user..SEQ_GGS_HEARTBEAT_HIST ;
CREATE SEQUENCE &&ogg_user..SEQ_GGS_HEARTBEAT_HIST INCREMENT BY 1 START WITH 1
ORDER ;
DROP TABLE &&ogg_user..GGS_HEARTBEAT_HISTORY;
CREATE TABLE &&ogg_user..GGS_HEARTBEAT_HISTORY
(     ID NUMBER ,
SRC_DB           VARCHAR2(30),
EXTRACT_NAME     varchar2(8),
SOURCE_COMMIT    TIMESTAMP,
TARGET_COMMIT    TIMESTAMP,
CAPTIME     TIMESTAMP,
CAPLAG           NUMBER,
PMPTIME     TIMESTAMP,
PMPGROUP    VARCHAR2(8 BYTE),
PMPLAG           NUMBER,
DELTIME     TIMESTAMP,
DELGROUP    VARCHAR2(8 BYTE),
DELLAG           NUMBER,
TOTALLAG    NUMBER,
thread           number,
update_timestamp timestamp,
EDDLDELTASTATS   number,
EDMLDELTASTATS   number,
RDDLDELTASTATS   number,
RDMLDELTASTATS   number
);

CREATE OR REPLACE TRIGGER &&ogg_user..GGS_HEARTBEAT_TRIG_HIST
BEFORE INSERT OR UPDATE ON &&ogg_user..GGS_HEARTBEAT_HISTORY
FOR EACH ROW
BEGIN
select seq_ggs_HEARTBEAT_HIST.nextval
into :NEW.ID
from dual;
select systimestamp
into :NEW.target_COMMIT
from dual;
select trunc(to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT ),1,
instr(:NEW.CAPTIME - :NEW.SOURCE_COMMIT,' '))) * 86400
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.CAPTIME - :NEW.SOURCE_COMMIT), instr((:NEW.CAPTIME -
:NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.CAPLAG
from dual;
select trunc(to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME),1,
instr(:NEW.PMPTIME - :NEW.CAPTIME,' '))) * 86400
```

```
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+7,2))
+ to_number(substr((:NEW.PMPTIME - :NEW.CAPTIME), instr((:NEW.PMPTIME -
:NEW.CAPTIME),' ')+10,6)) / 1000000
into :NEW.PMPLAG
from dual;
select trunc(to_number(substr((:NEW.DELTIME - :NEW.PMPTIME),1,
instr(:NEW.DELTIME - :NEW.PMPTIME,' ')))) * 86400
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+1,2)) * 3600
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+4,2) ) * 60
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+7,2))
+ to_number(substr((:NEW.DELTIME - :NEW.PMPTIME), instr((:NEW.DELTIME -
:NEW.PMPTIME),' ')+10,6)) / 1000000
into :NEW.DELLAG
from dual;
select trunc(to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),1,
instr(:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT,' ')))) * 86400
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+1,2)) * 3600
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+4,2) ) * 60
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+7,2))
+ to_number(substr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),
instr((:NEW.TARGET_COMMIT - :NEW.SOURCE_COMMIT),' ')+10,6)) / 1000000
into :NEW.TOTALLAG
from dual;
end ;
/

ALTER TRIGGER &&ogg_user..GGS_HEARTBEAT_TRIG_HIST ENABLE;
```

5.11 mgr.prm

```
COMMENT ********************************************************************
COMMENT * Primary System MANAGER CONFIG FILE
*
COMMENT * Filename: mgr.prm                                          *
COMMENT * Purpose: The MGR process specifies the port for which all golden    *
COMMENT *          gate processes will communicate.                           *
COMMENT ********************************************************************
--
-- 4-21-09 - Updated for GGS 10 SGEORGE
```

```
PORT 8999

-- DYNAMICPORTLIST was 7810-7899

COMMENT ********************************************************************
COMMENT * Golden gate trails matching es* will be purged after there has been *
COMMENT * no activity for 5 days.                                            *
COMMENT ********************************************************************

-- PURGEOLDEXTRACTS ./dirdat/*, USECHECKPOINTS, MINKEEPDAYS 5

COMMENT ********************************************************************
COMMENT * Automatically start any extract and replicat processes at startup   *
COMMENT * and will attempt to restart any extract process that abends after   *
COMMENT * waiting 2 minutes, but only up to 5 attempts.                       *
COMMENT ********************************************************************

-- These are commented out for testing
-- AUTOSTART EXTRACT  EXT_*
-- AUTOSTART EXTRACT  PMP_*

-- These are commented out for testing
-- AUTORESTART EXTRACT  EXT_*, WAITMINUTES 2, RETRIES 5
-- AUTORESTART EXTRACT  PMP_*, WAITMINUTES 2, RETRIES 5

LAGREPORTHOURS 1
LAGINFOMINUTES 3
LAGCRITICALMINUTES 5
```

5.12 pmp_hb.prm

```
-- Data Pump configuration file
-- last update
-- 11-3-08 SGEORGE
-- update 9-1-12 SGEORGE - no checkpoint info.
--
extract PMP_hb
-- Database login info
userid SOURCE, password GGS

-- Just in case we can't process a record we'll dump info here
discardfile ./dirrpt/PMP_hb.dsc, append

-- Remote host and remort manager port to write trail
rmthost coe-02, mgrport 9000
```

```
-- This is the Trail to where we output
rmttrail ./dirdat/hb
-- Heartbeat
include dirprm/HB_pmp.inc

-- Table  source.*;
```

5.13 rep_hb.prm

```
-- Updates
-- 3-17-10 - SGEORGE - Added reporting and heartbeat table and some comments.
-- 3-18-10 - SGEORGE - Updated the heartbeat table
-- 9-1-12  - SGEORGE - Udated HB table removed checkpoint info not needed.
--
replicat rep_hb
-- Use ASSUMETARGETDEFS when the source and target tables specified with a MAP
-- statement have the same column structure, such as when synchronizing a hot
-- site DO NOT USE IF YOU USE THE COLMAP Statement. USE Sourcedef file.
assumetargetdefs
--setting oracle_environment variable
--useful in  multi oracle home situations
--setenv (ORACLE_HOME="/u01/app/oracle/product/11.2.0/db112")
setenv (ORACLE_SID="db112r1")
--userid password password encrypted using encrypt ggsci command
userid SOURCE,password GGS

-- Just in case we can't process a record we'll dump info here
discardfile ./dirrpt/REP_hb.dsc, append

-- Use REPORTCOUNT to generate a count of records that have been processed
since
-- the Extract or Replicat process started
-- REPORTCOUNT [EVERY] <count> {RECORDS | SECONDS | MINUTES | HOURS} [, RATE]
REPORTCOUNT EVERY 5 MINUTES, RATE
include ./dirprm/HB_Rep.inc
--map <SOURCE_SCHEMA>.* ,target <TARGET_SCHEMA>.*;
```

Appendix C - Troubleshooting

Some common issues with the heartbeat tables:

Issue	Solution
Duplicate records in the heartbeat table	Check mapping in the datapump. Make sure you are not capturing the heartbeat twice, once with the MAP statement and a second with a wildcard

	map statement.
Heartbeat not showing up at target	Make sure the heartbeat table is not in a "excluded user" schema. Common issue in bi-directional configurations.
No heartbeat in trail	Check to make sure the source heartbeat table has one row for each thread and the thread numbers are listed. The heartbeat table is updated by which thread the heartbeat is running on.