Script to Collect Data Guard Primary Site Diagnostic Information for Version 10g and Above (Including RAC). (Doc ID 1577401.1)

APPLIES TO:

Oracle Database - Ent Edition - Version 10.1.0.2 and later
Information in this document applies to any platform.

MAIN CONTENT

```
-- NAME: new_dg_prim_diag.sql
--
-- Copyright 2002, Oracle Corporation
--
-- LAST UPDATED: 02-Sep-2015
--
-- USAGE: @new_dg_prim_diag
--
--  (Run from sqlplus on PRIMARY with a LOGICAL or PHYSICAL STANDBY as SYS)
--
-- PURPOSE:
--
--  This script is to be used to assist in the collection of information to
--  help troubleshoot Data Guard issues with a Primary Database
--
-- DISCLAIMER:
--
--  This script is provided for educational purposes only. It is NOT
--  supported by Oracle World Wide Technical Support.
--  The script has been tested and appears to work as intended.
--  You should always run new scripts on a test instance initially.
--
-- Script output is as follows:

set echo off
set feedback off
column timecol new_v  lue timestamp
column spool_extension new_v  lue suffix
SELECT TO_CHAR(sysdate,'yyyymmdd_hh24mi') timecol, '.html' spool_extension FROM dual;
column output new_v  lue dbname
SELECT value || '_' output FROM v$parameter WHERE name = 'db_unique_name';
spool new_dg_prim_diag_&&dbname&&timestamp&&suffix
set linesize 2000
set pagesize 50000
set numformat 999999999999999
set trim on
set trims on
set markup html on
set markup html entmap off
set feedback on

ALTER SESSION SET nls_date_format = 'DD-MON-YYYY HH24:MI:SS';
SELECT TO_CHAR(sysdate) time FROM dual;

set echo on

-- In the following output the DATABASE_ROLE should be PRIMARY as that is what this script is intended to be run
on.
```

```
-- PLATFORM_ID should match the PLATFORM_ID of the standby(s) or conform to the supported options in
-- Note: 413484.1 Data Guard Support for Heterogeneous Primary and Physical Standbys in Same Data Guard
Configuration
-- Note: 1085687.1 Data Guard Support for Heterogeneous Primary and Logical Standbys in Same Data Guard
Configuration
-- OPEN_MODE should be READ WRITE.
-- L    _MODE should be ARCHIVELOG.
-- FLASHBACK can be YES (recommended) or NO.
-- If PROTECTION_LEVEL is different from PROTECTION_MODE then for some reason the mode listed in
PROTECTION_MODE experienced a need to downgrade.
-- Once the error condition has been corrected the PROTECTION_LEVEL should match the PROTECTION_MODE
after the next log switch;

SELECT database_role role, name, db_unique_name, platform_id, open_mode, log_mode, flashback_on,
protection_mode, protection_level FROM v$database;

-- FORCE_L      ING is not mandatory but is recommended.
-- REMOTE_ARCHIVE should be ENABLE.
-- SUPPLEMENTAL_LOG_DATA_PK and SUPPLEMENTAL_LOG_DATA_UI must be enabled if the standby associated
with this primary is a logical standby.
-- During normal operations it is acceptable for SWITCHOVER_STATUS to be SESSIONS ACTIVE or TO STANDBY.
-- DG_BROKER can be ENABLED (recommended) or DISABLED.;

column force_logging format a13 tru
column remote_archive format a14 tru
column supplemental_log_data_pk format a24 tru
column supplemental_log_data_ui format a24 tru
column dataguard_broker format a16 tru

SELECT force_logging, remote_archive, supplemental_log_data_pk, supplemental_log_data_ui, switchover_status,
dataguard_broker FROM v$database;

-- The following query gives us information about catpatch. From this we can tell if the catalog version doesn''t
match the image version it was started with.

column version format a10 tru

SELECT v                                    _registry WHERE comp_id = 'CATPROC';

-- Check how many threads are enabled and started for this database. If the number of instances below does not
match then not all instances are up.

SELECT thread#, instance, status FROM v$thread;

-- The number of instances returned below is the number currently running.  If it does not match the number
returned in Threads above then not all instances are up.
-- VERSION should match the version from CATPROC above.
-- ARCHIVER can be (STOPPED | STARTED | FAILED). FAILED means that the archiver failed to archive a log last
time, but will try again within 5 minutes.
-- LOG_SWITCH_WAIT the ARCHIVE LOG/CLEAR LOG/CHECKPOINT event log switching is waiting for.
-- Note that if ALTER SYSTEM SWITCH LOGFILE is hung, but there is room in the current online redo log, then the
value is NULL.

column host_name format a32 wrap

SELECT thread#, instance_name, host_name, version, archiver, log_switch_wait FROM gv$instance ORDER BY
thread#;

-- Check how often logs are switching. Log switches should not regularly be occuring in < 20 mins.
-- Excessive log switching is a performance overhead. Whilst rapid log switching is not in itself a Data Guard issue it
can affect Data guard.
```

```
-- It may also indicate a problem with log shipping. Use redo log size = 4GB or redo log size >= peak redo rate x
20 minutes.

SELECT fs.log_switches_under_20_mins, ss.log_switches_over_20_mins FROM (SELECT  SUM(COUNT
(ROUND((b.first_time - a.first_time) * 1440) )) "LOG_SWITCHES_UNDER_20_MINS"  FROM v$archived_log a,
v$archived_log b WHERE a.sequence# + 1 = b.sequence# AND a.dest_id = 1 AND a.thread# = b.thread#  AND
a.dest_id = b.dest_id AND a.dest_id = (SELECT MIN(dest_id) FROM gv$archive_dest WHERE t
                    IS NOT NULL) AND ROUND((b.first_time - a.first_time) * 1440)  < 20 GROUP BY
ROUND((b.first_time - a.first_time) * 1440))  fs, (SELECT  SUM(COUNT (ROUND((b.first_time - a.first_time) *
1440) )) "LOG_SWITCHES_OVER_20_MINS"  FROM v$archived_log a, v$archived_log b WHERE a.sequence# + 1 =
b.sequence# AND a.dest_id = 1 AND a.thread# = b.thread#  AND a.dest_id = b.dest_id AND a.dest_id = (SELECT
MIN(dest_id) FROM gv$archive_dest WHERE target='PRIMARY' AND destination IS NOT NULL) AND
ROUND((b.first_time - a.first_time) * 1440)  > 19 GROUP BY ROUND((b.first_time - a.first_time) * 1440)) ss;

column  minutes  format a12

SELECT (CASE WHEN bucket = 1 THEN '<= ' || TO_CHAR(bucket * 5) WHEN (bucket >1 AND bucket < 9) THEN
TO_CHAR(bucket * 5 - 4) || ' TO ' || TO_CHAR(bucket * 5) WHEN bucket > 8 THEN '>= ' || TO_CHAR(bucket * 5 -
4) END) "MINUTES", switches "LOG_SWITCHES" FROM (SELECT bucket , COUNT(b.bucket) SWITCHES FROM
(SELECT WIDTH_BUCKET(ROUND((b.first_time - a.first_time) * 1440), 0, 40, 8) bucket FROM v$archived_log a,
v$archived_log b WHERE a.sequence# + 1 = b.sequence# AND a.dest_id = b.dest_id  AND a.thread# =
b.thread#  AND a.dest_id = (SELECT MIN(dest_id) FROM gv$archive_dest WHERE t                IMARY' AND
destination IS NOT NULL)) b GROUP BY bucket ORDER BY bucket);

-- Check the number and size of online redo logs on each thread.

SELECT thread#, group#, sequence#, bytes, archived ,status FROM v$log ORDER BY thread#, group#;

-- The following query is run to see if standby redo logs have been created in preparation for switchover.
-- The standby redo logs should be the same size as the online redo logs.<br>There should be (( # of online logs
per thread + 1) * # of threads) standby redo logs.
-- A value of 0 for the thread# means the log has never been allocated.

SELECT thread#, group#, sequence#, bytes, archived, status FROM v$standby_log order by thread#, group#;

-- This query produces a list of defined archive destinations. It shows if they are enabled, what process is servicing
that destination,
-- if the destination is local or remote.

column destination format a35 wrap
column process format a7
column ID format 99
column mid format 99

SELECT thread#, dest_id, destination, target, schedule, process FROM gv$archive_dest gvad, gv$instance gvi
WHERE gvad.inst_id = gvi.inst_id AND destination is NOT NULL ORDER BY thread#, dest_id;

-- This select will give further detail on the destinations as to what options have been set.
-- Register indicates whether or not the archived redo log is registered in the remote destination control fileOptions.

set numwidth 8
column archiver format a8
column affirm format a6
column error format a55 wrap
column register format a8

SELECT thread#, dest_id, gvad.archiver, transmit_mode, affirm, async_blocks, net_timeout, max_failure,
delay_mins, reopen_secs reopen, register, binding FROM gv$archive_dest gvad, gv$instance gvi WHERE
gvad.inst_id = gvi.inst_id AND destination is NOT NULL ORDER BY thread#, dest_id;

-- The following select will show any errors that occured the last time an attempt to archive to the destination was
```

```
attempted.
-- If ERROR is blank and status is VALID then the archive completed correctly.

SELECT thread#, dest_id, gvad.status, error, fail_sequence FROM gv$archive_dest gvad, gv$instance gvi WHERE
gvad.inst_id = gvi.inst_id AND destination is NOT NULL ORDER BY thread#, dest_id;

-- The query below will determine if any error conditions have been reached by querying the v$dataguard_status
view (view only available in 9.2.0 and above).

column message format a80

SELECT gvi.thread#, timestamp, message FROM gv$dataguard_status gvds, gv$instance gvi WHERE gvds.inst_id =
gvi.inst_id AND severity in ('Error','Fatal') ORDER BY timestamp, thread#;

-- Query v$managed_standby to see the status of processes involved in the shipping redo on this system.
-- Does not include processes needed to apply redo.

SELECT inst_id, thread#, process, pid, status, client_process, client_pid, sequence#, block#, active_agents,
known_agents FROM gv$managed_standby ORDER BY thread#, pid;

-- The following query will determine the current sequence number and the last sequence archived.
-- If y                ly archiving using the LGWR process then the archived sequence should be one higher than the
current sequence.
-- If remotely archiving using the ARCH process then the archived sequence should be equal t
          .
-- The applied sequence information is updated at log switch time.
-- The "Last Applied" value should be checked with the actual last log applied at the standby, only the standby is
guaranteed to be correct.

SELECT cu.thread#, cu.dest_id, la.lastarchived "Last Archived", cu.currentsequence "Current Sequence",
appl.lastapplied "Last Applied" FROM (select gvi.thread#, gvd.dest_id, MAX(gvd.log_sequence) currentsequence
FROM gv$archive_dest gvd, gv$instance gvi WHERE gvd.status = 'VALID' AND gvi.inst_id = gvd.inst_id GROUP BY
thread#, dest_id) cu, (SELECT thread#, dest_id, MAX(sequence#) lastarchived FROM gv$archived_log WHERE
resetlogs_change# = (SELECT resetlogs_change# FROM v$database) AND archived = 'YES' GROUP BY thread#,
dest_id) la, (SELECT thread#, dest_id, MAX(sequence#) lastapplied FROM gv$archived_log WHERE
resetlogs_change# = (SELECT resetlogs_change# FROM v$database) AND applied = 'YES' GROUP BY thread#,
dest_id) appl WHERE cu.thread# = la.thread# AND cu.thread# = appl.thread# AND cu.dest_id = la.dest_id AND
cu.dest_id = appl.dest_id ORDER BY 1, 2;

-- The following select will attempt to gather as much information as possible from the standby.
-- Standby redo logs are not supported with Logical Standby until Version 10.1.
-- The ARCHIVED_SEQUENCE# from a logical standby is the sequence# created by the apply
                          .

set numwidth 8
column dest_id format 99
column Active format 99

SELECT dest_id, database_mode, recovery_mode, protection_mode, standby_logfile_count, standby_logfile_active
FROM v$archive_dest_status WHERE destination IS NOT NULL;

-- Non-default init parameters. For a RAC DB Thread# = * means the value is the same for all threads (SID=*)
-- Threads with different values are shown with their individual thread# and values.

column num noprint

SELECT num, '*' "THREAD#", name, value FROM v$PARAMETER WHERE NUM IN (SELECT num FROM v$parameter
WHERE (isdefault = 'FALSE' OR ismodified <> 'FALSE') AND name NOT LIKE 'nls%'
MINUS
SELECT num FROM gv$parameter gvp, gv$instance gvi WHERE num IN (SELECT DIS  INCT gvpa.num FROM
gv$parameter gvpa, gv$parameter gvpb WHERE gvpa.num = gvpb.num AND  gvpa.value <> gvpb.value AND
```

```
(gvpa.isdefault = 'FALSE' OR gvpa.ismodified <> 'FALSE') AND gvpa.name NOT LIKE 'nls%') AND gvi.inst_id =
gvp.inst_id  AND (gvp.isdefault = 'FALSE' OR gvp.ismodified <> 'FALSE') AND gvp.name NOT LIKE 'nls%')
UNION
SELECT num, TO_CHAR(thread#) "THREAD#", name, value FROM gv$parameter gvp, gv$instance gvi WHERE num
IN (SELECT DISTINCT gvpa.num FROM gv$parameter gvpa, gv$parameter gvpb WHERE gvpa.num = gvpb.num
AND gvpa.v  lue <> gvpb.v  lue AND (gvpa.isdefault = 'FALSE' OR gvpa.ismodified <> 'FALSE') AND gvp.name NOT
LIKE 'nls%') AND gvi.inst_id = gvp.inst_id  AND (gvp.isdefault = 'FALSE' OR gvp.ismodified <> 'FALSE') AND
gvp.name NOT LIKE 'nls%' ORDER BY 1, 2;

spool off
set markup html off entmap on
set echo on
```

Didn't find what you are looking for?