ORADEBUG - UNDOCUMENTED ORACLE UTILITY

By

Miladin Modrakovic

rankoni@hotmail.com

Introduction

This document describes Oracle utility called *oradebug*. This tool is primarily used by Oracle worldwide customer support .With *oradebug* utility you can literally *see* the database engine. The *oradebug* is especially useful when things go very bad - e.g. the database just stops, hangs, or the database keeps crashing with the ORA-0600 error! To run this tool you must have administrator privileges.

Among the many useful things that can be done with oradebug are:

- enabling/disabling the SQL tracing for another user's session.
- suspending intensive processes
- finding information about shared memory and semaphores
- closing the trace file so that new one can be generated
- manipulating and dumping internal structures
- wake up processes etc.

Oradebug Commands

The *oradebug* utility consists of the following commands:

HELP	[command]	Describe one or all commands
SETMYPID		Debug current process
SETOSPID	<ospid>	Set OS pid of process to debug
SETORAPID	<orapid> ['force']	Set Oracle pid of process to debug
DUMP	<dump_name> <lvl> [addr]	Invoke named dump
DUMPSGA	[bytes]	Dump fixed SGA
DUMPLIST		Print a list of available dumps
EVENT	<text>	Set trace event in process
SESSION_EVENT	<text>	Set trace event in session
DUMPVAR	<p|s|uga> <name> [level]	Print/dump a fixed PGA/SGA/UGA variable
SETVAR	<p|s|uga> <name> <value>	Modify a fixed PGA/SGA/UGA variable
PEEK	<addr> <len> [level]	Print/Dump memory
POKE	<addr> <len> <value>	Modify memory
WAKEUP	<orapid>	Wake up Oracle process
SUSPEND		Suspend execution
RESUME		Resume execution
FLUSH		Flush pending writes to trace file
CLOSE_TRACE		Close trace file
TRACEFILE_NAME		Get name of trace file

```
LKDEBUG                                   Invoke global enqueue service debugger
NSDBX                                          Invoke CGS name-service debugger
-G              <Inst-List | def | all>        Parallel oradebug command prefix
-R              <Inst-List | def | all>        Parallel oradebug prefix (return output
SETINST          <instance# .. | all>          Set instance list in double quotes
SGATOFILE       <SGA dump dir>   Dump SGA to file; dirname in double quotes
DMPCOWSGA <SGA dump dir>   Dump & map SGA as COW; dirname in double quotes
MAPCOWSGA       <SGA dump dir>   Map SGA as COW; dirname in double quotes
HANGANALYZE     [level]                        Analyze system hang
FFBEGIN                                        Flash Freeze the Instance
FFDEREGISTER                                   FF deregister instance from cluster
FFTERMINST                                     Call exit and terminate instance
FFRESUMEINST                                   Resume the flash frozen instance
FFSTATUS                                       Flash freeze status of instance
SKDSTTPCS       <ifname>  <ofname>             Helps translate PCs to names
WATCH           <address> <len> <self|exist|all|target>  Watch a region of memory
DELETE          <local|global|target> watchpoint <id>    Delete a watchpoint
SHOW            <local|global|target> watchpoints     Show  watchpoints
CORE                                           Dump core without crashing process
IPC                                            Dump ipc information
UNLIMIT                                        Unlimit the size of the trace file
PROCSTAT                                       Dump process statistics
CALL            <func> [arg1] ... [argn]       Invoke function with arguments
```

oradebug help

Describe one or all debug commands available for use.

Syntax	Parameter
oradebug help < command name >	<command name > name of the debug command

If left alone *oradebug help* will list all debug commands.

Example

```
SQL> oradebug help show
SHOW          <local|global|target> watchpoints       Show  watchpoints
```

oradebug setmypid

Debug current process.

Syntax	Parameter
oradebug setmypid	None

oradebug setospid

Set OS process id of process to debug.

Syntax	Parameter
oradebug setospid <ospid>	<ospid> OS PID to attach

Example

```
select spid, pid
from v$process
where addr = (select paddr from v$session where sid = <your SID >);

SQL> oradebug setospid 19592
Oracle pid: 18, Unix process pid: 19592, image: oracle@apollo(TNS V1-V3)
```

oradebug setorapid

Set Oracle pid of process to debug.

Syntax	Parameter(s)
oradebug setorapid <orapid> ['force']	<orapid> Oracle PID ['force'] Force process

Example

```
SQL> oradebug setorapid 18  (use pid from the query above)
Unix process pid: 19592, image: oracle@apollo (TNS V1-V3) or  using force option
SQL> oradebug setorapid 18 force
Statement processed.
```

oradebug dump commands are explained in **ORADEBUG DUMPS** a special chapter of this paper dedicated to a various oracle dumps.

oradebug event

Set trace event in process.

Syntax	Parameter
oradebug event <text>	<text> Event name

Event numbers can be found in $ORACLE_HOME/rdbms/mesg/oraus.msg

or on the address below:

http://www.kevinloney.com/free/events.htm

Example

In this example I have used event event 10046 which is probably the most used event. To enable tracing for another session, the Oracle (PID) or the OS PID (SPID) must be identified from v$process view.

```
SQLPLUS> oradebug setospid 10929
Oracle pid: 91, Unix process pid: 10929, image: oracleorcl
SQLPLUS> oradebug EVENT 10046 trace name context forever, level 12
Statement processed.
```

The level can affect the behaviour of the event.Event 10046 can have level with following values:

Trace Name	**Level**	**Description**
TRACE_ACALL	1	Trace all calls
TRACE_ECALL	2	Trace "enabled" calls
TRACE_AEXCP	4	Trace all exceptions
TRACE_EEXCP	8	Trace "enabled" exceptions
TRACE_CIRCULAR	16	Trace w/ circular buffer
TRACE_BIND_VARS	32	Trace bind variables

Other combinations :

Level	**Description**
17	Trace all calls, using the buffer
22	Trace enabled calls and all exceptions using the buffer
32	Trace bind variables, without using the buffer
53	Yields the maximum level of tracing, using the buffer
37	Yields the maximum level of tracing, without using the buffer

oradebug session_event

Set trace event in session.

Syntax	Parameter
oradebug session_event <text>>	<text> Event name

Example

```
SQL>oradebug session_event 10046 trace name context forever,level 12
Statement proceed.
SQL > oradebug session_event 10046 trace name context off
Statement proceed.
```

The information goes to user dump destination.

oradebug dumpvar

Print/dump a fixed PGA/SGA/UGA variable.

Syntax	Parameter
oradebug dumpvar <p|s|uga> <name> [level]	<p|s|uga> PGA,SGA or UGA <name> Variable name [level] Level

Example

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug dumpvar sga kslwlst
ksllt kslwlst_ [200040AC, 20004174) = 00000000 00000009 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000000
00 00000000 00000000 00000000 ...
```

Let’s change variable value

```
SQL> oradebug poke 536887468 4 1
BEFORE: [200040AC, 200040B0) = 00000000
AFTER:    [200040AC, 200040B0) = 00000001

SQL> oradebug dumpvar sga kslwlst
ksllt kslwlst_ [200040AC, 20004174) = 00000001 00000009 00000000 00000000
00000000 00000000 00000000 00000000 00000000 000000
00 00000000 00000000 00000000 ...
```

oradebug setvar

Modify a fixed PGA/SGA/UGA variable.

Syntax	Parameter
oradebug setvar <p|s|uga> <name> <value>	<p|s|uga> PGA,SGA or UGA <name> Variable name <value> Variable’s new value

Example:

```
SQL> oradebug setmypid
Statement processed.

SQL> oradebug dumpvar sga kcfdfk
kfil kcfdfk_ [2000F6B0, 2000F6B4) = 00000190
SQL> show parameter db_files;
```

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------
db_files                             integer      200
SQL> oradebug setvar sga kcfdfk 200
BEFORE: [2000F6B0, 2000F6B4) = 00000190
AFTER:  [2000F6B0, 2000F6B4)   = 000000C8
SQL> oradebug dumpvar sga kcfdfk
kfil kcfdfk_ [2000F6B0, 2000F6B4) = 000000C8
```

oradebug peek

This command will dump memory address to a trace file which can be found in user dump destination.

Syntax	Parameters
oradebug peek <addr> <len> [level]	<addr> memory address <len> length [level level

Example

```
 select fsv.KSMFSNAM,sga.*
 from x$ksmfsv fsv, x$ksmmem sga
 where sga.addr=fsv.KSMFSADR
 and fsv.ksmfsnam like 'kgl%'
SQL> /
KSMFSNAM                                  ADDR      INDX    INST_ID KSMMMVAL
------------------------------------------------------------ -------- ---------- ---------- --------
kglt1_                                  200150F8     21566          1        0000011C
kgllat_                                  20015F64     22489          1        00
kglpnl_                                 2001602C     22539          1        00
kglpal_                                 200160F4     22589          1        00
kglllt_                                 200161BC     22639          1        00
SQL> oradebug peek 536957176 4
[200150F8, 200150FC) = 0000011C
```

oradebug poke

Modify memory.

Syntax	Parameters
oradebug poke <addr> <len> [level]	<addr> memory address <len> length <value> value

Example

```
SQL> oradebug help poke
POKE          <addr> <len> <value>      Modify memory
SQL> oradebug poke 536957176 4 668
BEFORE: [200150F8, 200150FC) = 0000011C
AFTER:  [200150F8, 200150FC) = 0000029C

 select fsv.KSMFSNAM,sga.*
 from x$ksmfsv fsv, x$ksmmem sga
 where sga.addr=fsv.KSMFSADR
 and fsv.ksmfsnam like 'kgl%'
SQL> /

KSMFSNAM                      ADDR        INDX    INST_ID    KSMMMVAL
---------------------------------------------------------------- -------- ---------- ---------- --------
kglt1_                      200150F8     21566         1       0000029C
kgllat_                     20015F64     22489         1       00
kglpnl_                     2001602C     22539         1       00
kglpal_                     200160F4      22589        1       00
kgllt_                      200161BC     22639         1       00
```

NOTE: The *oradebug poke* command is a very danger command. Please do not run this command in production database.

oradebug wakeup

Wake up process .

Syntax	Parameter
oradebug wakeup <orapid>	<orapid> Oracle PID

Example

You could post smon to cleanup the temporary segments using this command. This will wakeup smon to clean up the temporary segments

```
 SQL> select pid
         from v$process p, v$bgprocess b
         where b.paddr = p.addr
         and   name='SMON';
      PID
----------
        6

SQL> oradebug wakeup 6
```

oradebug suspend

oradebug also allows you to suspend a process. Firstly you need to identify the shadow process that you want to suspend. Then set your debug session to point to that process.

Syntax	Parameter
oradebug suspend	None.

Example

```
SQL> oradebug setospid 19272
Oracle pid: 26, Unix process pid: 19272, image: oracle@apollo (TNS V1-V3)
```

and then suspend its execution:

```
Sqlplus > oradebug suspend
Statement processed.
```

This stops a process .Examining v$session_wait shows that it is waiting on debugger.

oradebug resume

Resume suspended process.

Syntax	Parameter
oradebug resume	None.

Example

Resume the process which was already suspended:

```
SQL> oradebug resume
```

oradebug flush

Flush pending writes to trace file.

Syntax	Parameter
oradebug flush	None.

oradebug close_trace

Close trace file.

Syntax	Parameter
oradebug close_trace	None.

Example

This option is very useful if you have deleted a trace file with the session still live and now you want the session to resume tracing, but a new file doesn't appear.The reason why you have a problem under Unix is that the trace file is not closed - even if you set ***sql_trace*** to false.

```
select spid, pid
from v$process
where addr = (select paddr from v$session where sid = <your SID >);
```

```
oradebug setorapid {the pid above}    or  oradebug setospid {the spid above} then
oradebug flush
oradebug close_trace
```

The ***close_trace*** option is that you can do it even after you have deleted the trace file from the operating system level.

oradebug tracefile_name

Get name of the trace file.

Syntax	Parameter(s)
oradebug tracefile_name	None.

The ***lkdebug*** and ***nsdbx*** are utilities within the *oradebug* utility. They are intended for OPS/RAC.

oradebug lkdebug

```
SQL> oradebug lkdebug help
Usage:lkdebug [options]
 -l [rlp] <enqueue pointer>    Enqueue Object
 -r <resource pointer>         Resource Object
 -b <gcs shadow pointer>       GCS shadow Object
 -p <process id>               client pid
 -P <process pointer>                     Process Object
 -O <i1> <i2> <types>                     Oracle Format resname
 -a <res/lock/proc/pres>                  all <res/lock/proc/pres> pointers
 -a <res> [<type>]                        all <res> pointers by an optional type
 -a convlock                              all converting enqueue (pointers)
 -a convres                               all res ptr with converting enqueues
 -a name                                  list all resource names
 -a hashcount                             list all resource hash bucket counts
 -t                                       Traffic controller info
```

-s	summary of all enqueue types
-k	GES SGA summary info

oradebug nsdbx

Invoke Cluster Group Services (CGS) name-service debugger. The *nsdbx* is utility within the *oradebug* utility.

```
SQL> oradebug nsdbx help
Usage:nsdbx [options]
 -h                                              Help
 -p  <owner> <namespace> <key> <value> <nowait>  Publish a name-entry
 -d  <owner> <namespace> <key> <nowait>          Delete a name-entry
 -q  <namespace> <key>                           Query a namespace
 -an <namespace>                                 Print all entries in namespace
 -ae                                             Print all entries
 -as                                             Print all namespaces
```

oradebug -G

Parallel oradebug command prefix.

Syntax	Parameter
oradebug -G <Inst-List | def | all>	<Inst-List | Instance list def | Default. all> All instances.

Example

ORADEBUG [-g [DEF | INSTLIST]] subcommand subcommand-parameters

oradebug -g def event 10706 trace name context forever, level 10

oradebug -R

Parallel oradebug prefix.

Syntax	Parameter
oradebug -R <Inst-List | def | all>	<Inst-List | Instance list def | Default. all> All instances.

oradebug setinst

Set instance(s).

Syntax	Parameter
oradebug setinst <instance# .. | all>	<instance# .. | instance list in double quotes all> set all instances

Example

```
SQL> select * from v$active_instances;

INST_NUMBER      INST_NAME
----------- --------------------------------
        1                alpha:test1
        2                beta:test2
        3                omega:test3

SQL> oradebug setinst "1","2","3"
Statement processed.                                        or

SQL> oradebug setinst all
Statement processed.
```

oradebug sgatofile

Dump SGA to file.

Syntax	Parameter
oradebug sgatofile <SGA dump dir>	<SGA dump dir> SGA Directory name in double quotes.

oradebug dmpcowsga

Dump & map SGA as COW(Copy On Write).

Syntax	Parameter(s)
oradebug dmpcowsga <SGA dump dir>	<SGA dump dir> SGA Directory name in double quotes.

oradebug mappcowsga

Dump & map SGA as COW(Copy On Write). Shared(cow) where cow stands for 'Copy On Write' is memory that will write through the cache into real memory.

Syntax	Parameter(s)
oradebug mappcowsga <SGA dump dir>	<SGA dump dir> SGA Directory name in double quotes.

Example

```
SQL> oradebug ffbegin
Statement processed.

SQL> oradebug dmpcowsga "/ora-main/app/oracle/admin/test/udump"
Statement processed.
 [apollo]/ora-main/app/oracle/admin/test/udump/Aug__1_19:39:31_2003>

SQL> oradebug sgatofile "/ora-main/app/oracle/admin/test/udump"
Statement processed.

SQL>  oradebug mapcowsga "/ora-
main/app/oracle/admin/test/udump/Aug__1_19:39:31_2003"
Statement processed.
```

oradebug hanganalyze

Analyze system hang.

Syntax	Parameter
oradebug hanganalyze <level>	<level>

The levels are defined as follows:

Dump Level	Dump Contains
1-2	Only HANGANALYZE output, no process dump at all
3	Level 2 + Dump only processes thought to be in a hang (IN_HANG state)
4	Level 3 + Dump leaf nodes (blockers) in wait chains (LEAF,LEAF_NW,IGN_DMP state)
5	Level 4 + Dump all processes involved in wait chains (NLEAF state)
10	Dump all processes (IGN state)

Example

To perform cluster wide HANGANALYZE use the following syntax:

```
ORADEBUG setmypid
ORADEBUG setinst all
ORADEBUG -g def hanganalyze  <level>
```

oradebug ffbegin

Flash Freeze the Instance.

Syntax	Parameter
oradebug ffbegin	None

oradebug ffderegister

Flash Freeze deregister instance from cluster

Syntax	Parameter
oradebug ffderegister	None

oradebug ffterminst

Call exit and terminate instance.

Syntax	Parameter
oradebug ffterminst	None

oradebug ffresumeinst

Resume the flash frozen instance.

Syntax	Parameter
oradebug ffresumeinst	None

oradebug ffstatus

Flash freeze status of instance.

Syntax	Parameter
oradebug ffstatus	None

oradebug skdsttpcs

Helps translate PCs to names.

Syntax	Parameter
oradebug skdsttpcs <ifname> <ofname>	<ifname> File name <ofname>

oradebug watch

Watch a region of memory.

Syntax	Parameter
oradebug watch<address> <len> <self|exist|all|target>	<address> <len> <self|exist|all|target>

oradebug delete

Delete a watchpoint.

Syntax	Parameter
oradebug delete <local|global|target> watchpoint <id>	<local|global|target> watchpoint <id>

oradebug show

Show watchpoints.

Syntax	Parameter
oradebug show <local|global|target> watchpoints	<local|global|target> watchpoints

Example

Before you run this command you need to set init parameter **_ use_ism=false** (intimate shared memory)**.**Using ISM greatly reduces the overhead in the virtual-to-physical address translation layers for large Oracle instances, and also provides the feature for locking shared pages, eliminating any page outs.

```
SQL> oradebug setmypid
Statement processed.

SQL> oradebug dump errorstack 3
Statement processed.

SQL> oradebug help watch
WATCH        <address> <len> <self|exist|all|target>  Watch a region of memory
SQL> oradebug watch 4290682200 1 self
Local watchpoint 0 created on region [0xFFBE9D58, 0xFFBE9D59).
SQL> oradebug help show
SHOW         <local|global|target> watchpoints       Show  watchpoints
SQL> oradebug show local watchpoints
ID   Address                          Nbytes    Mode
==============================================================
0    0xFFBE9D58                           1        SELF

SQL> oradebug help delete
```

```
DELETE        <local|global|target> watchpoint <id>    Delete a watchpoint
SQL> oradebug delete local watchpoint 0
Local watchpoint 0 deleted on region [0xFFBE9D58, 0xFFBE9D59).
SQL> oradebug show local watchpoints
ID   Address                          Nbytes    Mode
```

oradebug core

Dump core without crashing process.

Syntax	Parameter
oradebug core	None.

oradebug ipc

command list semaphores and shared memory segments in use.Set process id before using it.
Also this option shows which network is Oracle using for RAC traffic.

Syntax	Parameter(s)
oradebug ipc	None.

Example

List semaphores and shared memory segments in use:

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug ipc
Information written to trace file.

---------             part of the trace file  ----------------------------
Number of semaphores per set:    = 77
Semaphores key overhead per set: = 4
User Semaphores per set:         = 73
Number of semaphore sets:        = 3
Semaphore identifiers:           = 3
Semaphore List=
2293760
-------------- system semaphore information -------------
IPC status from <running system> as of Tue Jul 22 13:49:16 EDT 2003
T      ID     KEY       MODE       OWNER    GROUP  CREATOR   CGROUP NSEMS
OTIME    CTIME
Semaphores:
s   2293760  0xddce3cac --ra-r-----   oracle    dba   oracle    dba    77 10:32:36   11:06:49
s    524289  0xddce3cad --ra-r-----   oracle    dba   oracle    dba    77 no-entry   11:06:49
s    524290  0xddce3cae --ra-r-----   oracle    dba   oracle    dba    77 11:06:50   11:06:49
```

```
s     983043   0x3f1b2fd4 --ra-r-----   oracle     dba   oracle     dba   129 13:44:25  11:03:38
s     196612   0xa5a509ac --ra-r-----   oracle     dba   oracle     dba   129 11:51:22 11:04:38
s    1507333  0x9bf960c8 --ra-r-----   oracle     dba   oracle     dba   129 10:33:43 10:14:12
```

Alternative is to use oracle ***sysresv*** command which is located at $ORACLE_HOME/bin directory.

Example

Which network is Oracle using for RAC traffic:

SSKGXPT 0x1a2932c flags SSKGXPT_READPENDING info for network 0
socket no 10 **IP 172.16.193.1 UDP 43739**
sflags SSKGXPT_WRITESSKGXPT_UP info for network 1
socket no 0 IP 0.0.0.0 UDP 0...

So you can see that we are using IP 172.16.193.1 with a UDP protocol.

oradebug unlimit

Remove the file size limit. Useful when need trace file larger then size specified by max_dump_size parameter.

Syntax	Parameter
oradebug unlimit	None

Example

oradebug procstat

Dump process Statistics.

Syntax	Parameter
oradebug procstat <ospid>	<ospid> OS PID

Example

Dump statistics for the DBWR background process:

```
  select pid,name
  from v$process p, v$bgprocess b
  where b.paddr = p.addr
SQL> /

       PID NAME
---------- -----
         2 PMON
         3 DBW0
```

```
          4 LGWR
          5 CKPT
          6 SMON
          7 RECO

6 rows selected.

SQL> oradebug setorapid 3
Unix process pid: 15668, image: oracle@apollo (DBW0)
SQL> oradebug procstat
Statement processed.
```

To find out where is the trace file located run:

```
SQL> oradebug tracefile_name
/ora-main/app/oracle/admin/test/bdump/test_dbw0_15668.trc
SQL>
```

oradebug call

Invoke function with arguments.

Syntax	Parameters
oradebug call <func> [arg1] ... [argn]	<func> function name [arg1] ... [argn] function's argument(s)

ORADEBUG DUMPS

Oradebug can be used to obtain information about internal database structures.

oradebug dump

Invoke named dump.

Syntax	Parameter
oradebug dump <dump_name> <level> [addr]	<dump_name> <level> [<addr>]

To see all the available list of dumps, use the "oradebug dumplist" command. It is very useful to know the dump list, because it is not documented with "alter session set events." You can also use the "alter session set events" command to take a dump.

Excerpt from the oradebug dumplist command:

```
SQL> oradebug dumplist
```

EVENTS
TRACE_BUFFER_ON
TRACE_BUFFER_OFF
HANGANALYZE
LATCHES
PROCESSSTATE
SYSTEMSTATE
INSTANTIATIONSTATE
REFRESH_OS_STATS
CROSSIC
CONTEXTAREA
HEAPDUMP
HEAPDUMP_ADDR
POKE_ADDRESS
POKE_LENGTH ……….

Dumps of State Objects

oradebug dump systemstate

Dump of all state objects for all the processes on the system

Syntax	Parameter
oradebug dump systemstate <level>	<level>

Example

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug unlimit
Statement processed.
SQL> oradebug setinst all
Statement processed.
SQL> oradebug -g def dump systemstate 10
Statement processed.
```

oradebug dump processstate

Dump of all state objects for process

Syntax	Parameter
oradebug dump processstate <level>	<level>

Example

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug -g all dump processtate 10
Statement processed.
```

oradebug dump errorstack

Dump of the process call stack and other information.

Syntax	Parameter
oradebug dump errorstack <level>	<level>

The levels for the *errorstack* dump are as follows:

Dump Level	Dump Contains
0	dump error buffer
1	level 0 + call stack
2	level 1 + process state objects
3	level 2 + context area

Example

```
SQL> oradebug setospid 13446
Oracle pid: 12, Unix process pid: 13446, image: oracle@apollo (TNS V1-V3)
SQL> oradebug unlimit
Statement processed.
SQL> oradebug dump errorstack 3
Statement processed.
```

File Dumps

oradebug dump controlf

The contents of the current controlfile can be dumped in text form to a process trace file in the *user_dump_dest* directory using the CONTROLF dump.

Syntax	Parameter
oradebug dump controlf <level>	<level>

The levels for this dump are as follows.

Dump Level	Dump Contains
1	only the file header
2	just the file header, the database info record, and checkpoint progress records
3	all record types, but just the earliest and latest records for circular reuse record types
4	as above, but includes the 4 most recent records for circular reuse record types
5+	as above, but the number of circular reuse records included doubles with each level

Example

```
SQL> oradebug setospid 19272
Oracle pid: 26, Unix process pid: 19272, image: oracle@tomcat (TNS V1-V3)
SQL> oradebug dump controlf  4
Statement processed.
```

oradebug dump file_hdrs   <level>

Dump datafile headers.

Syntax	Parameter
oradebug dump file_hdrs <level>	<level>

The levels for this dump are as follows.

Dump Level	Dump Contains
1	Record of datafiles in controlfile (for practice compare with controlfile dump)
2	Level 1 + generic information
3	Level 2 + additional datafile header information.

Increasing level more than 3 do not provide additional information (size of trace files are identical for levels equivalent or greater than 3.

```
-rw-r-----   1 oracle   dba       119345 Feb  2 20:58 demo1_ora_3788.trc
-rw-r-----   1 oracle   dba       119345 Feb  2 21:03 demo1_ora_8043.trc
-rw-r-----   1 oracle   dba       119347 Feb  2 21:07 demo1_ora_10607.trc
```

Also you can use command *alter system dump datafile <file_number> block <block_id>;* to dump one block .

To dump one or more blocks:

alter system dump datafile <file_number> block min <first block > block max <last block >;

Example

```
select segment_name, header_file, header_block from dba_segments
where segment_type='ROLLBACK';


SEGMENT_NAME    HEADER_FILE HEADER_BLOCK
--------------- ----------- ------------
SYSTEM                    1          2
RBS0                      2          2
RBS1                      2       3202
RBS2                      2       6402
```

oradebug dump redohdr

Dump redo headers.

Syntax	Parameter
oradebug dump redohdr <level>	<level>

The levels for this dump are as follows.

Dump Level	Dump Contains
1	Record of log file records in controlfile
2	Level 1 + generic information
3	Level 2 + additional log file header information.

Memory Dumps

oradebug dump buffers

Dump of buffer cache.

Syntax	Parameter
oradebug dump buffers <level>	<level>

The level number to specify in the event syntax is the decimal tablespace relative data block address.

The levels for the BUFFERS dump are as follows:

Dump Level	**Dump Contains**
1	dump the buffer headers only
2	include the cache and transaction headers from each block
3	include a full dump of each block
4	dump the working set lists and the buffer headers and the cache header for each block
5	include the transaction header from each block
6	include a full dump of each block

Most levels high than 6 are equivalent to 6, except that levels 8 and 9 are the same as 4 and 5 respectively.
For level 1 to 3 the information is dumped in buffer header order.

For levels higher than 3, the buffers and blocks are dumped in hash chain order.

oradebug dump library_cache

Dump library cache statistics.

Syntax	Parameter
oradebug dump library_cache <level>	<level>

The levels for the *library cache* dump are as follows :

Dump Level	**Dump Contains**
1	dump libracy cache statistics
2	also include a hash table
3	level 2 + dump of the library object handles
4	Level 3 + dump of the heap

Example

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug dump library_cache 2
Statement processed.
```

oradebug dump heapdump

Dump structure of a memory heap.

Syntax	Parameter
oradebug dump heapdump <level>	<level>

The levels for the *heapdump* are as follows :

Dump Level	Dump Contains
1	include PGA heap
2	include Shared Pool
4	include UGA heap
8	include CGA heap
16	include Top CGA
32	include Large Pool

oradebug dump heapdump_addr

Syntax	Parameter
oradebug dump heapdump_addr<level><address>	<level> <address> descriptor address

The levels for the *heapdump*_addr are as follows :

Dump Level	Dump Contains
1	dump structure
2	also include contents

Example

```
SQL> oradebug setmypid
Statement processed.
SQL> oradebug unlimit
Statement processed.
SQL> oradebug dump heapdump 2
Statement processed.
```

```
------------------      part of the trace file       --------------------
*** SESSION ID:(7.14111) 2003-09-03 13:01:21.412
****************************************************
HEAP DUMP heap name="sga heap"  desc=0x80000030
extent sz=0xfc4 alt=48 het=32767 rec=9 flg=2 opc=0
parent=0 owner=0 nex=0 xsz=0x3d2bdf4
EXTENT 0
 Chunk b5815ff8 sz= 41699524    perm      "perm           "  alo=19395672
 Chunk b7fda8bc sz=   887380    free      "               "
 Chunk b80b3310 sz=      560    freeable  "library cache  "  ds=b80b5a0c
 Chunk b80b3540 sz=     2588    freeable  "sql area       "  ds=b80b5898
 Chunk b80b3f5c sz=      732    freeable  "sql area       "  ds=b80ba69c
```

Open the trace file and look for the ds (descriptor address).Convert hex to decimal and do the dump.

```
SQL> oradebug dump heapdump_addr 1 3087751692
```

-------------------------- part of the trace file generated using command above -------------

```
Statement processed.
*** 2003-09-03 13:12:35.330
*** SESSION ID:(8.20679) 2003-09-03 13:12:35.322
HEAP DUMP heap name="library cache"  desc=0xb80b5a0c
extent sz=0x224 alt=32767 het=8 rec=9 flg=2 opc=0
parent=80000030 owner=b80b5804 nex=0 xsz=0x224
EXTENT 0
 Chunk b80b3324 sz=      464    perm      "perm           "  alo=176
 Chunk b80b34f4 sz=       76    freeable  "kgltbtab       "
EXTENT 1
 Chunk b80b55b4 sz=      500    perm      "perm           "  alo=500
 Chunk b80b57a8 sz=       40    free      "               "
EXTENT 2
 Chunk b80b57f4 sz=      244    perm      "perm           "  alo=244
 Chunk b80b58e8 sz=       52    free      "               "
 Chunk b80b591c sz=       76    freeable  "kgltbtab       "
 Chunk b80b5968 sz=       76    freeable  "kgltbtab       "
```

INDEX DUMP

oradebug dump treedump

Dumps the structure of an index tree. The dump file contains one line for each block in the tree, indented to show its level, together with a count of the number of index entries in the block.

Syntax	Parameter
oradebug dump treedump <object_id>	<level> object_id

Example

```
select  object_id  from sys.dba_objects
where owner = upper('&Owner') and
object_name = upper('&IndexName');

SQL> oradebug setmypid
Statement processed.
SQL> oradebug dump treedump 40
Statement processed.
```

Instead of *oradebug dump treedump* command you can also use

```
alter session set events 'immediate trace name treedump level n';
```

That will give you a trace file with the one line for each block in the B*-tree and a row count for each block.If there are large numbers of empty or nearly empty blocks, then the index is a good candidate for being rebuilt.

Conclusion

To conclude, oradebug utility gives the dba another piece of important information that helps in identifying and resolving different kind of database issues.

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a first version, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author does not take any responsibility for that.

References

Metalink – Oracle Corporation

Doc ID: 39817.1 Interpreting Raw SQL_TRACE and dbms_support.start_trace output
Doc ID: 28863.1 ORADBX - Quick Reference.
Doc ID: 175006.1 Steps to generate hanganalyze trace files.

Doc ID: 105395.1 How to find PID for setospid in oradebug.
Doc ID: 205809.1 Script to Collect OPS Diagnostic Information.
Doc ID: 123322.1 SYSRESV Utility.
Doc ID: 68281.1 Determinig which instance owns which shared memory & semaphore segments.
Doc ID: 215858.1 Interpreting hanganalyze trace files to diagnose hanging and performance problems.
Doc ID: 2112587.6 How to take OPS-aware Oradebug dump in 8i multiple instances database.

Books

Oracle8i Internal Services for Waits, Latches, Locks and Memory - Steve Adams
Solaris Internals - published by Jim Mauro and Richard McDougall.

White Papers

Oracle X$ Tables – Steve Adams
09'97 SunWorld article Shared memory uncovered by Jim Mauro
Oracle9i Memory Management: Easier Than Ever , *Sushil Kumar, Oracle Corporation*

Class

Oracle Internals and Advanced Performance Tuning- Master Class Copenhagen, Denmark 2003

Web Resources :

http://www.ixora.com.au/ site authored by Steve Adams.
http://www.jlcomp.demon.co.uk/ site authored by Jonathan Lewis
http://freespace.virgin.net/bill.doyle/ork_trc.htm
http://www.oracleadvice.com/Tips/9ibreak.htm

.