

Troubleshooting - Using LOGDUMP

LOGDUMP is a great utility and a real bonus to the Oracle GoldenGate software bundle. Without LOGDUMP, we could not read a Trail file which would make us blind to troubleshooting data related issues.

LOGDUMP has a command line interface that allows you to open files, format the display, and navigate through a file including filtering data. To invoke the utility, go to the GoldenGate home directory and type "logdump", as shown in the following example.

```
[oracle@dbserver1 ggs]$ ./logdump

Oracle GoldenGate Log File Dump Utility
Version 10.4.0.19 Build 002

Copyright (C) 1995, 2009, Oracle and/or its affiliates. All rights reserved.

Logdump 1 >ENV
Version                : Linux, x86, 32bit (optimized) on Sep 29 2009 08:53:18

Current Directory     : /u01/app/oracle/product/ggs
LogTrail              : *Not Open*
Display RecLen       : 140
Logtrail Filter       : On
Trans History         : 0 Transactions, Records 100, Bytes 100000
LargeBlock I/O       : On, Blocksize 57344
Local System          : LittleEndian
Logtrail Data         : BigEndian/ASCII
Logtrail Headers      : ASCII
Dump                  : ASCII
Timeoffset            : LOCAL
Scan Notify Interval: 10000 records, Scrolling On

Logdump 2 >
```

As with the GGSCI utility, LOGDUMP increments the number at its command prompt for each command entered. Even if you exit LOGDUMP, the number will increment when you return. This is because LOGDUMP maintains a history of commands used.

The preceding example shows the output of the ENV command, which is one of many commands required to be productive with LOGDUMP. Firstly we must tell LOGDUMP to open a file, and then specify how much detail you require before scanning or filtering data. However, should you get stuck there is always the HELP command to get you back on track, which incidentally shows many undocumented commands.

Opening Files

Let's start with the OPEN command. Before opening a file, we must choose one. Execute the following Linux command from the GoldenGate home directory to list the available files.

```
[oracle@dbserver1 ggs]$ ls -l dirdat
-rw-rw-rw- 1 oracle oinstall 3859 Jun 19 17:10 INITLOAD01.DAT
-rw-rw-rw- 1 oracle oinstall 68929 Nov 9 13:28 sa000004
-rw-rw-rw- 1 oracle oinstall 68929 Nov 9 13:32 sa000005
-rw-rw-rw- 1 oracle oinstall 68929 Nov 9 13:35 sa000006
```

Let's open local Trail file sa000024 from LOGDUMP.

```
Logdump 2 >open dirdat/sa000004
Current LogTrail is /u01/app/oracle/product/ggs/dirdat/sa000004
```

Before we can see the contents of the file, we must setup a view in LOGDUMP. The following table of commands will provide the necessary detail depending on your requirements:

Command	Description
FILEHEADER [on off detail]	Controls whether or not the trail file header is displayed and how much detail.
GHDR [on off]	Controls whether or not the record header is displayed with each record.
DETAIL [on off data]	Displays a list of columns that includes the column ID, length, plus values in hex and ASCII. DATA adds hex and ASCII data values to the column list.
USERTOKEN [detail]	Displays the actual token data.
RECLLEN [<# of bytes>]	Controls how much of the record data is displayed in characters

So, working through the list, enable the file header detail, GDHR, user token detail and record length options.

```
Logdump 3 >fileheader detail
Logdump 4 >ghdr on
Logdump 6 >detail on
Logdump 7 >usertoken detail
Logdump 8 >reclen 128
Reclen set to 128
```

Viewing the Header Record

Now it's time to navigate our way through the file starting at position 0, the first record in the file. This is the beginning of the header record.

```
Logdump 9 >pos 0
Reading forward from RBA 0
```

To view the header record we must step to the next Relative Byte Address (RBA). This is easy using LOGDUMP, just type **next** or **n**.

```
2010/11/09 12:56:49.942.356 FileHeader          Len   928 RBA 0
Name: *FileHeader*
 3000 01a2 3000 0008 4747 0d0a 544c 0a0d 3100 0002 | 0...0...GG..TL..1...
 0002 3200 0004 ffff fffd 3300 0008 02f1 bad1 bae9 | ..2.....3.....
```

Included in the header record is a wealth of information, given that we have enabled a detailed view. The information is grouped by type with a list of related tokens, shown in the following example output.

```
GroupID x30 '0' TrailInfo          Info x00 Length 418
TokenID x30 '0' Signature          Info x00 Length 8
TokenID x31 '1' Compatibility      Info x00 Length 2
TokenID x32 '2' Charset            Info x00 Length 4
TokenID x33 '3' CreationTime       Info x00 Length 8
TokenID x34 '4' URI                Info x00 Length 38
TokenID x36 '6' Filename            Info x00 Length 19
TokenID x37 '7' MultiPart           Info x00 Length 1
TokenID x38 '8' Seqno               Info x00 Length 4
TokenID x39 '9' FileSize            Info xff Length 8
TokenID x3a ':' FirstCSN            Info x00 Length 129
TokenID x3b ';' LastCSN            Info xff Length 129
TokenID x3c '<' FirstIOTime         Info x00 Length 8
TokenID x3d '=' LastIOTime          Info xff Length 8

GroupID x31 '1' MachineInfo        Info x00 Length 100
TokenID x30 '0' Sysname             Info x00 Length 7
TokenID x31 '1' Nodename            Info x00 Length 17
TokenID x32 '2' Release             Info x00 Length 14
TokenID x33 '3' Version             Info x00 Length 36
TokenID x34 '4' Hardware            Info x00 Length 6

GroupID x32 '2' DatabaseInfo        Info x00 Length 299
TokenID x30 '0' Vendor              Info x00 Length 2
TokenID x31 '1' Name                Info x00 Length 6
TokenID x32 '2' Instance            Info x00 Length 6
TokenID x33 '3' Charset             Info x00 Length 4
TokenID x34 '4' MajorVersion        Info x00 Length 2
TokenID x35 '5' MinorVersion        Info x00 Length 2
TokenID x36 '6' VerString           Info x00 Length 225
TokenID x37 '7' ClientCharset       Info x00 Length 4
TokenID x38 '8' ClientVerString     Info x00 Length 12

GroupID x33 '3' ProducerInfo        Info x00 Length 83
TokenID x30 '0' Name                Info x00 Length 10
TokenID x31 '1' DataSource           Info x00 Length 2
TokenID x32 '2' MajorVersion        Info x00 Length 2
TokenID x33 '3' MinorVersion        Info x00 Length 2
TokenID x34 '4' MaintLevel          Info x00 Length 2
```

```

TokenID x35 '5' BugFixLevel      Info x00 Length 2
TokenID x36 '6' BuildNumber      Info x00 Length 2
TokenID x37 '7' VerString        Info x00 Length 29

GroupID x34 '4' ContinuityInfo  Info x00 Length 8
TokenID x30 '0' RecoveryMode    Info x00 Length 4

```

Viewing the Transaction Record

Typing **next** or **n** again steps through each record in the file. The following example shows details of an INSERT operation against the SRC.USERS table, including the actual data and record count. You could argue that each record would always have a record count of 1. This is not true for LOBs which are split into 2KB chunks when written to a Trail file.

```
Logdump 19 >n
```

```

-----
Hdr-Ind      :      E (x45)      Partition   :      . (x04)
UndoFlag     :      . (x00)      BeforeAfter:      A (x41)
RecLength    :     29 (x001d)    IO Time     : 2010/11/09 13:25:14.000.000
IOType       :      5 (x05)      OrigNode    :     255 (xff)
TransInd     :      . (x00)      FormatType  :      R (x52)
SyskeyLen    :      0 (x00)      Incomplete  :      . (x00)
AuditRBA     :     138          AuditPos    : 38737936
Continued    :      N (x00)      RecCount    :      1 (x01)

```

```
2010/11/09 13:25:14.000.000 Insert          Len 29 RBA 999
```

```
Name: SRC.USERS
```

```

After Image:
0000 0007 0000 0003 5352 4300 0100 0500 0000 0159 | .....TEST.....Y
0002 0005 0000 0001 4e                               | .....N

```

```

Column      0 (x0000), Len 7 (x0007)
Column      1 (x0001), Len 5 (x0005)
Column      2 (x0002), Len 5 (x0005)

```

The equivalent transaction record in the remote Trail file is identical to that found in the local Trail file, and is identifiable by the same Audit Position number.

Let's query the USERS table in the SRC schema to see the actual record that we are viewing in LOGDUMP.

```
SQL> select * from SRC.USERS
2 where USER_ID = 'TEST';
```

```

USER_ID REGISTERED ASSIGNED
-----
TEST    Y             N

```

Each record in the Trail file contains the following information:

- The operation type, such as an insert, update, or delete
- The transaction indicator (TransInd): 00 beginning, 01 middle, 02 end or 03 whole of transaction
- The before or after indicator (BeforeAfter) for update operations
- The commit timestamp
- The time that the change was written to the GoldenGate file
- The type of database operation
- The length of the record
- The Relative Byte Address (RBA) within the GoldenGate file
- The schema and table name

Miscellaneous Commands

The miscellaneous commands are useful for displaying additional information, and are listed in the following table.

Command	Description
HISTORY	List previous commands
RECORD	Display audit record
SKIP [<count>]	Skip down <count> records
SFH	Scans for the file header record
ENV	Displays GoldenGate environment details
COUNT [detail]	Count the records in the file
EXIT	Exits LOGDUMP

This example highlights the power of the COUNT command:

```

Logdump 34 >count
LogTrail u01/app/oracle/product/ggs/dirdat/sa000004 has 602 records
Total Data Bytes          15703
  Avg Bytes/Record         26
Delete                    280
Insert                    320
RestartOK                  1
Others                     1
Before Images             280
After Images              321

Average of 17 Transactions
  Bytes/Trans .....      2623
  Records/Trans ...       35
  Files/Trans .....       1
  
```

		Partition 0
RestartOK	1	
After Images	1	
FileHeader		
		Partition 0
Total Data Bytes	928	
Avg Bytes/Record	928	
Others	1	
SRC.USERS		
		Partition 4
Total Data Bytes	14775	
Avg Bytes/Record	24	
Delete	280	
Insert	320	
Before Images	280	
After Images	320	

Filtering Records

You can do some pretty fancy stuff with LOGDUMP filtering. A whole suite of commands are set aside for this. We can filter on just about anything that exists in the Trail file, such as process name, RBA, record length, record type, even a string!

The following example shows the required syntax to filter on DELETE operations. Note that LOGDUMP reports how many records have been excluded by the filter.

```
Logdump 52 >filter include iotype delete
Logdump 53 >n

2010/11/09 13:31:40.000.000 Delete          Len   17 RBA 5863
Name: SRC.USERS
Before Image:                               Partition 4  G  b
0000 000d 0000 0009 414e 4f4e 594d 4f55 53  | .....ANONYMOUS

Filtering suppressed      42 records
```