Oracle GoldenGate - Trail Internals

This page describes the internal structure of the GoldenGate trail files. It is based on original research and therefore cannot be guaranteed to be correct. It should, however, give you a good idea of what information is stored in the trails and why.

The research was performed in Oracle 11.2.0.3 on OEL 5.6 running in VirtualBox. The Oracle GoldenGate version was 11.2.1.0.1

This page describes the trails created using the Classic Capture method. This page does not yet include LOB data types for which further research is still required.

Trail files are generated by both the original extract and the Data Pump (extract) process. Trail files are read by the Data Pump (extract) and Replicat (replicat) processes.

The trail files generated by the original extract and the Data Pump vary a little. Some processing appears to be postponed until the Data Pump stage. Some additional file URIs are added to the trail header by the Data Pump. Also the primary keys for some UPDATE statements appear to be modified by the Data Pump process.

The trail files are binary, though significant portions are stored in ASCII. You would need to encrypt the trail files to prevent someone with operating system access examining the contents using the *strings* utility. Presumably to ensure interoperability across operating systems and databases, all numbers are stored in ASCII format, making the trail files even less secure than other formats such as Oracle exports which at least store numbers using the Oracle internal representation.

In order to complete the research, I developed a C utility called *TrailAnalyzer* which dumps the contents of a trail. At present it can handle any trail that I have encountered, but it will probably need to enhancement as I test additional GoldenGate functionality.

The *TrailAnalyzer* outputs each character in the trail file in hexadecimal. It attempts to interpret these characters based on their context. It embeds XML-like start and end tags to represent the current level. For example <47_54_52> and </47_54_52>. It also embeds comments to clarify the content. Comments start with a hash sign (#) and are terminated by the end of line character. Output from theTrailAnalyzer has been included throughout this page to illustrate the examples.

The trail file contains a file header and a file body. The file header contains one header record. The file body can contain zero or more data records.

Each record has a four byte record header. A record can contain zero or more sub records. Each sub record also has a four byte record header. The design allows for infinite levels of sub records, nested recursively. In practice the deepest level of nesting I have found so far is four.

Record Header

Each record header contains four bytes. The following table shows the structure.

Byte#	**0**	**1**	**2**	**3**
Description	Type	Flag	Length	

Type Field

The type field describes the contents of the record. The type is dependent on the level on which it appears i.e. it is context dependent.

Originally I would guess that the type field was originally an uppercase alphabetic character, probably in the range "A" to "Z". This was probably augmented fairly rapidly by the numeric characters "0" to "9". These ranges appear to have become exhausted and new characters have been added including various punctuation symbols. As the latter characters are not compatible with C syntax, I have chosen to use their hexadecimal representations throughout. These are shown in the following table:

0	1	2	3	4	5	6	7	8	9	:	;	<	=		
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D		
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
P	Q	R	S	T	U	V	W	X	Y	Z					
50	51	52	53	54	55	56	57	58	59	5A					

Other values may be in use for as yet unidentified structures.

I have not yet identified records with all of the alphabetic type values. However, it is a reasonable assumption that these values have been reserved, if not actually used. I have found structures with all of the numeric values and the punctuation characters described above.

Flag Field

The flag field appears to contain bit values describing the contents of the record. I do not currently understand the significance of these values. Values that I have encountered so far include:

- 0x00
- 0x01
- 0x80
- 0xFF

Further examples are required to identify any pattern.

Length Field

The length is a two byte value. This provides a potential range of 0..65535.

GoldenGate uses continuation records when the data length exceeds 65535 bytes. Further research is required to understand the exact algorithm

File Structure

The following illustration shows and overview of the structure of a trail file:

```
# File header
# Record
46 00 04 2A           # Type=46 (F) Flag=0 Len=1066
......                # Data
5A 00 04 2A           # Type=5A (Z) Flag=0 Len=1066
# File body
# Record
47 01 00 3A           # Type=47 (G) Flag=1 Len=58
......                # Data
5A 01 00 3A           # Type=5A (Z) Flag=1 Len=58
# Record
47 01 00 90           # Type=47 (G) Flag=1 Len=144
......                # Data
5A 01 00 90           # Type=5A (Z) Flag=1 Len=144
# Record
47 01 00 78           # Type=47 (G) Flag=1 Len=120
......                # Data
5A 01 00 78           # Type=5A (Z) Flag=1 Len=120
# Record
47 01 00 77           # Type=47 (G) Flag=1 Len=119
......                # Data
5A 01 00 77           # Type=5A (Z) Flag=1 Len=119
# Record
47 01 00 78           # Type=47 (G) Flag=1 Len=120
```

```
......                # Data
5A 01 00 78           # Type=5A (Z) Flag=1 Len=120
# Record
47 01 00 83           # Type=47 (G) Flag=1 Len=131
......                # Data
5A 01 00 83           # Type=5A (Z) Flag=1 Len=131
```

File Header

The file header is identified by a record header with type 0x46 (F).

The file header is terminated by a record header with type 0x59 (Z). The length field of the terminator record header is identical to the length field of the original record header.

File Body

The file body consists of zero or more records. Each record is identified by a record header with type 0x47 (G).

Each record is terminated by a record header with type 0x59 (Z). The length field of the terminator record header is identical to the length field of the original record header.

File Header

The file header is identified by a record header with type 0x46 (F).

The file header contains a number of sub records

The file header generated by the extract process is generally around 1050 bytes in size. The file header generated by the data pump process is larger as it includes additional records. The size of the file header will vary according to the configuration is it includes several variable length fields such as process names and directory names.

The file header has five sub records. Each sub record contains further sub records nested within it.

The top level records are:

Type	Description
0x30	Extract details
0x31	Operating System details
0x32	Database details
0x33	GoldenGate details
0x34	Unknown

Extract Details (0x30)

The type field for an extract details record is 0x30 (0).

The following is an example of the extract details:

```
30 00 01 C2   # Type=30 (0) Flag=0 Len=450
<46_30>
  30 00 00 08   # Type=30 (0) Flag=0 Len=8
  <46_30_30>
    47 47 0D 0A 54 4C 0A 0D
    </46_30_30>
      31 00 00 02   # Type=31 (1) Flag=0 Len=2
    <46_30_31>
      00 03
    </46_30_31>
    32 00 00 04   # Type=32 (2) Flag=0 Len=4
```

```
   <46_30_32>
     20 00 00 00
   </46_30_32>
   33 00 00 08   # Type=33 (3) Flag=0 Len=8
   <46_30_33>
     # Timestamp
     02 F1 FD 87 75 A4 17 24
   </46_30_33>
   34 00 00 30   # Type=34 (4) Flag=0 Len=48
   <46_30_34>
     # URI extract
     # String Len = 46 (0x2E)
     uri:vm4:juliandyke:com::home:oracle:goldengate
   </46_30_34>
   36 00 00 29   # Type=36 (6) Flag=0 Len=41
   <46_30_36>
     # filename
     # String Len = 39 (0x27)
     /home/oracle/goldengate/dirdat/ex000016
   </46_30_36>
   37 00 00 01   # Type=37 (7) Flag=0 Len=1
   <46_30_37>
     01
   </46_30_37>
   38 00 00 04   # Type=38 (8) Flag=0 Len=4
   <46_30_38>
     00 00 00 10
   </46_30_38>
   39 FF 00 08   # Type=39 (9) Flag=255 Len=8
   <46_30_39>
     00 00 00 00 00 00 00 00
   </46_30_39>
  3A 00 00 81   # Type=3A (:) Flag=0 Len=129
  <46_30_3A>
    # Commit SCN
    07 31 31 34 35 30 37 34 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00
  </46_30_3A>
  3B FF 00 81   # Type=3B (;) Flag=255 Len=129
  <46_30_3B>
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
    00
  </46_30_3B>
  3C 00 00 08   # Type=3C (<) Flag=0 Len=8
  <46_30_3C>
    # Timestamp
    02 F1 FD 87 76 B3 53 40
  </46_30_3C>
  3D FF 00 08   # Type=3D (=) Flag=255 Len=8
  <46_30_3D>
    00 00 00 00 00 00 00 00
  </46_30_3D>
</46_30>
```

The following table describes the sub records within this record:

Type	Description
0x30	Purpose is unknown. Currently only observed as a 8 byte record with the following contents: 47 47 0D 0A 54 4C 0A 0D. This is equivalent to GG<cr><lf>TL<lf><cr>
0x31	Purpose unknown. Currently only observed as a 2 byte record with the following contents: 00 03
0x32	Purpose unknown. Currently only observed as a 4 byte record with the following contents: 20 00 00 00
0x33	Timestamp – 8 bytes fixed size
0x34	String - URI of GoldenGate home directory. For example:uri:vm4:juliandyke:com::home:oracle:goldengate
0x35	Contains one of more sub-records of type 35. Each sub-record is a string containing a URI representing the GoldenGate home directory. There is one sub-record for each data pump extract trail file
0x36	String - Filename of extract trail file
0x37	Purpose unknown. Currently only observed as a 1 byte record with the following contents: 01
0x38	Purpose unknown. Currently only observed as a 4 byte record with the following contents: 00 00 00 10 or 00 00 00 18
0x39	Purpose unknown. Currently only observed as a 8 byte record with the following contents: 00 00 00 00 00 00 00 00
0x3A	129 byte record. First 8 bytes contain Commit SCN for example 07 31 31 34 35 30 37 34 which represents 1145074. Size of SCN will presumably increase when it reaches 10,000,000. Remaining bytes are 0x00
0x3B	129 byte record. All bytes are 0x00. Purpose unknown
0x3C	Timestamp – 8 bytes fixed size
0x3D	Purpose unknown – 8 bytes fixed size. All bytes are 0x00

Operating System Details (0x31)

The type field for an operating system details record is 0x31 (1).

The following is an example of the operating system details:

```
31 00 00 71   # Type=31 (1) Flag=0 Len=113
<46_31>
  30 00 00 07   # Type=30 (0) Flag=0 Len=7
  <46_31_30>
    # Operating System
    # String Len = 5 (0x5)
    Linux
  </46_31_30>
  31 00 00 14   # Type=31 (1) Flag=0 Len=20
  <46_31_31>
    # Hostname
    # String Len = 18 (0x12)
    vm4.juliandyke.com
  </46_31_31>
  32 00 00 15   # Type=32 (2) Flag=0 Len=21
  <46_31_32>
    # Kernel Release
    # String Len = 19 (0x13)
    2.6.32-100.26.2.el5
  </46_31_32>
  33 00 00 25   # Type=33 (3) Flag=0 Len=37
  <46_31_33>
    # Kernel Version
    # String Len = 35 (0x23)
    #1 SMP Tue Jan 18 20:11:49 EST 2011
  </46_31_33>
```

```
  34 00 00 08   # Type=34 (4) Flag=0 Len=8
  <46_31_34>
    # Architecture
    # String Len = 6 (0x6)
    x86_64
  </46_31_34>
</46_31>
```

The following table describes the sub records within this record:

Type	Description
0x30	String - Operating system e.g. Linux
0x31	String - Hostname e.g. vm4.juliandyke.com
0x32	String - Kernel Releasee.g. 2.6.32-100.26.2.el5. Equivalent to uname -r
0x33	String - Kernel Version e.g. #1 SMP Tue Jan 18 20:11:49 EST 2011. Equivalent to uname -v
0x34	String - Architecturee.g. x86_64. Equivalent to uname -m, -i or -p

Note - it is not clear which of the three uname parameters is used to determine the architecture.

Database Details (0x32)

The type field for a database details record is 0x32 (2).

The following is an example of the database details:

```
32 00 01 61   # Type=32 (2) Flag=0 Len=353
<46_32>
  30 00 00 02   # Type=30 (0) Flag=0 Len=2
  <46_32_30>
    00 07
  </46_32_30>
  31 00 00 07   # Type=31 (1) Flag=0 Len=7
  <46_32_31>
    # Database Name
    # String Len = 5 (0x5)
    NORTH
  </46_32_31>
  32 00 00 07   # Type=32 (2) Flag=0 Len=7
  <46_32_32>
    # Database Name
    # String Len = 5 (0x5)
    NORTH
  </46_32_32>
  33 00 00 04   # Type=33 (3) Flag=0 Len=4
  <46_32_33>
    00 00 03 EB
  </46_32_33>
  34 00 00 02   # Type=34 (4) Flag=0 Len=2
  <46_32_34>
    00 0B
  </46_32_34>
  35 00 00 02   # Type=35 (5) Flag=0 Len=2
  <46_32_35>
    00 02
  </46_32_35>
  36 00 00 E7   # Type=36 (6) Flag=0 Len=231
  <46_32_36>
    # Oracle Banner
    # String Len = 229 (0xE5)
    Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production
    PL/SQL Release 11.2.0.3.0 - Production
    CORE      11.2.0.3.0      Production
    TNS for Linux: Version 11.2.0.3.0 - Production
```

```
    NLSRTL Version 11.2.0.3.0 - Production
  </46_32_36>
  37 00 00 04   # Type=37 (7) Flag=0 Len=4
  <46_32_37>
    40 00 00 00
  </46_32_37>
  38 00 00 0C   # Type=38 (8) Flag=0 Len=12
  <46_32_38>
    # Oracle Version
    # String Len = 10 (0xA)
    11.2.0.3.0
  </46_32_38>
  39 00 00 04   # Type=39 (9) Flag=0 Len=4
  <46_32_39>
    00 00 00 01
  </46_32_39>
  3A 00 00 02   # Type=3A (:) Flag=0 Len=2
  <46_32_3A>
    00 00
  </46_32_3A>
  3B 00 00 04   # Type=3B (;) Flag=0 Len=4
  <46_32_3B>
    00 00 00 01
  </46_32_3B>
  3C 00 00 14   # Type=3C (<) Flag=0 Len=20
  <46_32_3C>
    # Case Sensitivity
    00 00 00 10 14 14 14 14 14 14 14 14 14 14 14 14
    11 14 14 14
  </46_32_3C>
</46_32>
```

The following table describes the sub records within this record:

Type	Description
0x30	Possibly Database Type e.g. ORACLE. Currently only observed as a 2 byte record with the following contents: 00 07. (ORACLE?)
0x31	String - Database Name e.g. NORTH (may be Database Unique Name or Instance Name)
0x32	String - Database Name e.g. NORTH (may be Database Unique Name or Instance Name)
0x33	Purpose is unknown. Currently only observed as a 4 byte record with the following contents: 00 00 03 EB (1003 decimal)
0x34	Purpose is unknown. Currently only observed as a 2 byte record with the following contents: 00 0B (11 decimal)Possibly major version number (11)
0x35	Purpose is unknown. Currently only observed as a 2 byte record with the following contents: 00 02 (2 decimal)Possibly minor version number (2)
0x36	String - Oracle Banner from "SELECT banner FROM v$version" For example: Oracle Database 11g Enterprise Edition Release 11.2.0.3.0 - 64bit Production PL/SQL Release 11.2.0.3.0 - Production CORE 11.2.0.3.0 Production TNS for Linux: Version 11.2.0.3.0 - Production NLSRTL Version 11.2.0.3.0 - Production
0x37	Purpose is unknown. Currently only observed as a 4 byte record with the following contents: 40 00 00 00
0x38	String - Oracle Version e.g. 11.2.0.3.0
0x39	Purpose is unknown. Currently only observed as a 4 byte record with the following contents: 00 00 00 01
0x3A	Purpose is unknown. Currently only observed as a 2 byte record with the following contents: 00 00
0x3B	Purpose is unknown. Currently only observed as a 4 byte record with the following contents: 00 00 00 01
0x3C	Case sensitivity. Currently only observed as a 20 byte record with the following contents: 00 00 00 10 14 14 14 14 14 14 14 14 14 14 14 14 11 14 14 14.First 4 bytes are length 16 - remaining 16 bytes are data

The significance of the Case Sensitivity field is unknown. The same data appears in the DEFGEN report.

The following table is a possible list of database types:

Value	Database
1	DB2 UDB
2	DB2 ZOS
3	CTREE
4	INGRES
5	MSSQL
6	MYSQL
7	ORACLE
8	SQLMX
9	SYBASE
10	TERADATA
11	TIMESTEN
12	INFORMIX
13	NONSTOP
14	ENSCRIBE
15	MSACCESS
16	ODBC
17	VSAM
18	DB2400
19	GENERIC
20	DB2400 REMOTE
21	POSTGRESQL

GoldenGate Details (0x30)

The type field for a GoldenGate details record is 0x33 (3).

The following is an example of the GoldenGate details:

```
33 00 00 72   # Type=33 (3) Flag=0 Len=114
<46_33>
  30 00 00 05   # Type=30 (0) Flag=0 Len=5
  <46_33_30>
    # Extract Process
    # String Len = 3 (0x3)
    EX1
  </46_33_30>
  31 00 00 02   # Type=31 (1) Flag=0 Len=2
  <46_33_31>
    00 03
  </46_33_31>
  32 00 00 02   # Type=32 (2) Flag=0 Len=2
  <46_33_32>
    00 0B
  </46_33_32>
  33 00 00 02   # Type=33 (3) Flag=0 Len=2
  <46_33_33>
    00 02
  </46_33_33>
  34 00 00 02   # Type=34 (4) Flag=0 Len=2
  <46_33_34>
    00 01
  </46_33_34>
  35 00 00 02   # Type=35 (5) Flag=0 Len=2
  <46_33_35>
    00 00
```

```
  </46_33_35>
  36 00 00 02   # Type=36 (6) Flag=0 Len=2
  <46_33_36>
    00 01
  </46_33_36>
  37 00 00 41   # Type=37 (7) Flag=0 Len=65
  <46_33_37>
    # GoldenGate Version
    # String Len = 63 (0x3F)
    Version 11.2.1.0.1 OGGCORE_11.2.1.0.1_PLATFORMS_120423.0230_FBO
  </46_33_37>
</46_33>
```

The following table describes the sub records within this record:

Type	Description
0x30	Extract Process name e.g. EX1
0x31	Purpose is unknown. Currently only observed as a 2 byte record with the following contents: 00 03
0x32	2 byte number – GoldenGate version number part 1 of 5 e.g. 0x00 0x0B = 11 decimal
0x33	2 byte number – GoldenGate version number part 2 of 5 e.g. 0x00 0x02 = 2 decimal
0x34	2 byte number – GoldenGate version number part 3 of 5 e.g. 0x00 0x01 = 1 decimal
0x35	2 byte number – GoldenGate version number part 4 of 5 e.g. 0x00 0x00 = 0 decimal
0x36	2 byte number – GoldenGate version number part 5 of 5 e.g. 0x00 0x01 = 1 decimal
0x37	String – GoldenGate versionFor example:Version 11.2.1.0.1 OGGCORE_11.2.1.0.1_PLATFORMS_120423.0230_FBO

Unknown Details (0x34)

The type field for an unknown details record is 0x34 (4).

The following is an example of the unknown details:

```
<46_34>
  30 00 00 04   # Type=30 (0) Flag=0 Len=4
  <46_34_30>
    00 00 00 01
  </46_34_30>
</46_34>
```

The following table describes the sub record within this record:

Type	Description
0x30	Purpose is unknown. Currently only observed as a 4 byte record with the following contents: 00 00 00 01

File Body

The file body consists of one or more records with type field 0x47 (G).

Each record is followed by a terminator record with type field 0x5A (Z)

Each record contains up to three sub record types.

The top level record types are:

Type	Description
0x44	Data
0x48	Header
0x54	Transaction

Header Record

The header record has a type field value of 0x48 (H)

The header record consists of a fixed part and a variable part.

The fixed part is around 34 bytes in length. It contains a structure of containing information about the row including:

- flags
- timestamp
- redo byte address (in redo log file)
- length of data record

The variable part contains the schema name and object name of the table affected by the change.

The following is an example of a header record:

```
48 00 00 2A   # Type=48 (H) Flag=0 Len=42
<47_48>
  45 04 00 41 00 15 73 FF 02 F1 FD 87 76 B3 53 40
  00 00 00 00 00 AF 72 3C 00 00 00 00 06 52 00 00
  00 01 55 53 30 31 2E 54 38 00
</47_48>
```

Byte(s)	Length	Comments
0	1	Always 0x45 - purpose unknown - may be a type field
1	1	Always 0x04 - purpose unknown
2	1	Always 0x00 - purpose unknown
3	1	Always 0x41 - purpose unknown
4-5	2	Length of data record (type 44)
6	1	Flag byte - see below
7	1	Always 0xFF - purpose unknown
8-15	8	Timestamp
16-19	4	Always 0x00 - possibly part of RBA
20-23	4	RBA block and offset. See below
24-27	4	Always 0x00 - possibly part of RBA
28	1	RBA sequence number
29	1	Always 0x52 - purpose unknown - may be a type field
30-32	3	Always 0x00 - purpose unknown
33	1	Always 0x01 - purpose unknown
34+	-	<Schema name>.<object name> for table being changed

The following flag values have been observed:

Hex	Binary	Description
0x03	00000011	Delete Row
0x05	00000101	Insert Row
0x0F	00001111	Update Row
0x73	01110011	Update Row including primary key
0x74	01110100	Insert Row including LONG column
0x96	10010110	Start Extract

0x97	10010111	Start Extract
0xA1	10100001	Insert Row - continued row

The exact significance of each bit value has not yet been identified

Note that the Start Extract record is GoldenGate specific. It does not appear in the Oracle redo log.

The Redo Byte Address (RBA) specifies the location at which the change is located in the Oracle online redo log. A single-instance RBA consists of three elements;

```
<sequence number>.<block number>.<offset>
```

The sequence number is incremented every time a log file switch occurs. The sequence number allows archived redo logs to be applied in sequential order during a recovery. Each redo thread has a separate set of sequence numbers.

The block number refers to the block within the redo log in which the change first appears. The block number reflects a number of redo log blocks. The redo log block size varies between platforms. It is typically 512 or 1024 bytes. The redo log block size can be determined from various views within the database, for example:

```
SQL> SELECT blocksize FROM v$log;
```

Note that it is possible to override the default redo log block size in the ALTER DATABASE ADD LOGFILE GROUP, so it is theoretically possible for different redo log groups to have different block sizes. In practice, I would suspect this would only be a transitional state for most sites.

The block number and the offset are combined in the GoldenGate trail file. The default redo log block size in Linux is 512 bytes. It requires nine bits to represent the range of values between 0 and 511. The block number is therefore shifted nine bits to the left and combined with the offset using a bitwise OR.

For example consider the following RBA in the trail file:

```
00 00 00 00 00 AF 72 3C 00 00 00 00 06
```

The block number/offset component is 00 00 00 00 00 AF 72 3C. We will call this number N. It converts as follows:

```
Offset = N & 0x1FF = 0xAF723C & 0x1FF = 0x3C

Block Number = N >> 9 = 0xAF723C >> 9 = 0x57B9
```

The sequence number component is 00 00 00 06. This value does not require conversion.

The equivalent RBA in the redo log file symbolic dump is therefore 0x000006.000057b9.003c

Data Record

The data record has a type field value of 0x44 (D)

Data records are the most complex trail structure and are described in more detail in the following section.

Transaction Record

The transaction record always contains a ROWID. For the first change in any transaction, the transaction record also contains the Commit SCN and the Oracle transaction ID.

The following is an example of a transaction record for the first change in a transaction:

```
54 00 00 2F   # Type=54 (T) Flag=0 Len=47
```

```
<47_54>
  52 00 00 14   # Type=52 (R) Flag=0 Len=20
  <47_54_52>
    # ROWID
    41 41 41 53 66 39 41 41 45 41 41 41 41 49 4E 41
    41 41 00 01
  </47_54_52>
  4C 00 00 07   # Type=4C (L) Flag=0 Len=7
  <47_54_4C>
    # Commit SCN
    1145074
  </47_54_4C>
  36 00 00 08   # Type=36 (6) Flag=0 Len=8
  <47_54_36>
    # Transaction ID
    3.11.848
  </47_54_36>
</47_54>
```

The following is an example of a transaction record for a subsequent change in a transaction:

```
54 00 00 18   # Type=54 (T) Flag=0 Len=24
<47_54>
  52 00 00 14   # Type=52 (R) Flag=0 Len=20
  <47_54_52>
    # ROWID
    41 41 41 53 66 39 41 41 45 41 41 41 41 49 4E 41
    41 42 00 01
  </47_54_52>
</47_54>
```

The transaction record contains the following fields:

Type	Description
0x36	ASCII value - Oracle transaction ID e.g 3.11.848. Only included for first change in transaction
0x4C	ASCII value - Commit SCN e.g. 1145074. Only included for first change in transaction
0x52	ROWID e.g. 41 41 41 53 66 39 41 41 45 41 41 41 41 49 4E 41 41 41 00 01 which is equivalent to AAASf9AAEAAAAINAAA. The significance of the two final bytes (always 00 01) is not currently known

Data

The data record has a type field value of 0x44 (D)

The data record contains all columns changed in a single row. If multiple rows are changed there will be one data record for each change.

A single row can span multiple data records. In this case the header record flag field will differ between the first and subsequent rows

The data record also contains any primary key values required to identify the row being updated. Although the ROWID is recorded in the transaction record, it cannot be used as there is ROWIDs cannot be guaranteed to be identical between the source and target databases.

In the extract trail, the primary key can be null for some updates (at least). The primary key appears to be updated by the data pump extract. Further investigation is required to understand the conditions under which this occurs.

If a primary key is included in the data record, it appears at the start of the record. The remainder of the record consists of column values. The primary key is preceded by a two byte length value which specifies the total size of all columns in the key value.

Each column value has a header and a body. The header consists of four bytes. The first two bytes are the column number; the remaining two bytes are the length of the value. The body contains the value. Column numbers start at zero.

The values stored in the data record differ from the Oracle internal representations for some data types.

There are at least four different internal control structures which appear within the data section:

- For multi-column primary keys there is a two-byte length field, followed by each primary key value. Each primary key value consists of a:
    - two byte column number
    - two byte length value
    - variable length column value

- For regular columns there is a two byte column number followed by a two byte length. For each column there will be:
    - two byte column number
    - two byte length value
    - variable length column value

- For LONG columns spanning multiple records there is a 20 byte identifier. This consists of:
    - four byte number e.g. 00 00 00 03. Purpose unknown
    - four byte number e.g. 00 00 00 06. Purpose unknown. May be maximum number of pieces
    - four byte number e.g. 00 00 00 02. Sequence number for row piece. Sequence numbers start at 1
    - four byte number e.g. 00 00 00 D7. Number of bytes for column in previous data records. In other words start offset.
    - four byte number e.g. 00 00 07 D0. Number of bytes for this column in current record
    - variable length column value

Data records containing LONG column pieces appear to have flag field values of 0x74.

- For regular columns spanning multiple records e.g. VARCHAR2, the variable length column value is continued in the new record without any additional control data. For an insert, the first data record has flag field 0x05 and continuation records have flag field 0xA1

The following section discusses the representation of each data type:

NUMBER

NUMBER data types are represented a four byte length value followed by one or more ASCII values. The number of ASCII values is equivalent to the number of characters in the column value.

For example 2013 is represented as follows:

```
00 00 00 04 32 30 31 33
```

In the above the length is 4 bytes. The digits have been converted into their ASCII values e.g. 0x32 is "2" 0x30 is "0" etc.

Null values are represented by the following 4 bytes:

```
FF FF 00 00
```

Note that the GoldenGate representation is less efficient than the equivalent Oracle internal NUMBER representation.

VARCHAR2

VARCHAR2 data types are represented by a four byte length value followed by one or more ASCII values. The number of ASCII values is equivalent to the number of characters in the column value.

For example "GoldenGate" is represented in a VARCHAR (20) column as follows:

```
00 00 00 0A 47 6F 6C 64 65 6E 47 61 74 65
```

NULL values are represented by the following 4 bytes:

```
FF FF 00 00
```

CHAR

CHAR data types are represented by a value that is the size of the CHAR column plus two bytes. For a non-NULL value, the first two bytes are 00 00. These are followed by one or more ASCII values. The ASCII value is padded with spaces up to the maximum length of the CHAR column. The number of ASCII values is equivalent to the number of characters in the column value.

For example "GoldenGate" is represented in a CHAR (20) column as follows:

```
00 00 47 6F 6C 64 65 6E 47 61 74 65 20 20 20 20 20 20 20 20 20 20
```

Note the two initial zero bytes and the padding of the value with space characters (0x20). This means that the 10 byte value requires 22 bytes to store it in the trail file.

If the CHAR (20) column is NULL then the following bytes are stored:

```
FF FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The first two bytes are NULL. The remaining 20 bytes are set to 0. This means that NULL values for the CHAR value are very inefficient in GoldenGate.

DATE

DATE data types are represented by a 21-byte value. For a non-NULL value, the first two bytes are 00 00. These are followed by ASCII values for the date in the following format:

```
YYYY-MM-DD:HH24:MI:SS
```

For example 2013-03-14:22:39:56 is represented as follows:

```
00 00 32 30 31 33 2D 30 33 2D 31 34 3A 32 32 3A 33 39 3A 35 36
```

If the DATE column is NULL then the following bytes are stored

```
FF FF 31 39 30 30 2D 30 31 2D 30 31 3A 30 30 3A 30 30 3A 30 30
```

For NULL values the first two bytes are FF FF. The remaining bytes represent the date 1900-01-01:00:00:00.

The above shows that the GoldenGate representation of a NULL value for a DATE column is also very inefficient.

TIMESTAMP

TIMESTAMP data types are represented by a 31-byte value. For a non-NULL value, the first two bytes are 00 00. These are followed by ASCII values for the date in the following format

```
YYYY-MM-DD:HH24:MI:SS.NNNNNNNNN
```

GoldenGate is capable of storing timestamps internally down to nanoseconds.

For example 2013-03-14:22:39:56.236526 is represented as follows:

```
00 00 32 30 31 33 2D 30 33 2D 31 34 3A 32 32 3A 33 39 3A 35 36 2E
32 33 36 35 32 36 30 30 30
```

If the TIMESTAMP column is NULL then the following bytes are stored:

```
FF FF 31 39 30 30 2D 30 31 2D 30 31 3A 30 30 3A 30 30 3A 30 30 2E
30 30 30 30 30 30 30 30 30
```

For NULL values the first two bytes are FF FF. The remaining bytes represent the timestamp 1900-01-01:00:00:00.000000000

RAW

RAW values have a four byte length value followed by the binary values of the bytes in the data value.

For example, in Oracle 11.2.0.3 on Linux x86-64, the default base address is 0x60000000. As this is a 64-bit platform, addresses are stored as 8-byte values. The base address can be determined from X$KSMMEM as follows:

```
SQL> SELECT addr FROM x$ksmmem WHERE indx = 0;

ADDR
----------------
0000000060000000
```

Note that the above query must be executed by a user with SYSDBA privileges.

The following table was created for this example:

```
CREATE TABLE t20
(
  c1 NUMBER PRIMARY KEY,
  c2 RAW(8)
);
```

One row was inserted as follows:

```
INSERT INTO us01.t20
SELECT 1,addr FROM x$ksmmem WHERE indx = 0;

COMMIT;
```

Note that the above statement must be executed by a user with SYSDBA privileges.

The following value is stored in the GoldenGate trail for the RAW column:

```
00 00 00 08 00 00 00 00 60 00 00 00
```

NULL values are represented by the following 4 bytes:

```
FF FF 00 00
```

LONG

The following table and primary key index were created to test LONG values

```
CREATE TABLE t13
(
  owner VARCHAR2(30),
  view_name VARCHAR2(30),
  text_length NUMBER,
  text LONG
);

ALTER TABLE t13 ADD CONSTRAINT t13_pk PRIMARY KEY (owner,view_name);
```

Oracle does not support LONG columns in INSERT..SELECT statements. However the SQL*Plus COPY command can copy LONG columns. In this example, the source view is DBA_VIEWS.

For example:

```
COPY TO us01/us01@NORTH APPEND t13 USING -
SELECT owner,view_name,text_length,text -
FROM dba_views -
WHERE owner = 'SYS' AND view_name = 'DBA_TABLES';
```

The length of the text column for the DBA_TABLES view in DBA_VIEWS is reported by the following query:

```
SQL> SELECT text_length FROM dba_views WHERE view_name = 'DBA_TABLES';

TEXT_LENGTH
-----------
7070
```

The first record (flag field 0x05) contains the non-LONG columns:

```
00 00 00 07                                       # Column 0, Length 7
   00 00 00 03 53 59 53                           # owner_name=SYS
00 01 00 0E                                       # Column 1, Length 14
   00 00 00 0A 44 42 41 5F 54 41 42 4C 45 53      # view_name=DBA_TABLES
00 02 00 08                                       # Column 2, Length 8
   00 00 00 04 37 30 37 30                        # text_length=7070
00 03 00 04                                       # Column 3, Length 4
   FF FF 00 00                                    # text=NULL
```

Initially the LONG column is inserted as a NULL value. It is updated by subsequent records. Note that in the Oracle online redo log, this operation is performed as a single change. GoldenGate is responsible for transforming the change into multiple operations.

The second record (flag field 0x74) contains the following control data:

```
00 00 00 03            # Unknown
00 00 00 04            # Number of pieces = 4
00 00 00 01            # Record number 1
00 00 00 00            # Start offset 0 bytes
00 00 04 2E            # 1070 bytes
73 65 6C 65 63 74 20 75 2E 6E 61 6D 65 2C 20 6F
2E 6E 61 6D 65 2C 0A 20 20 20 20 20 20 20 64 65
63 6F 64 65 28 62 69 74 61 6E 64 28 74 2E 70 72
........
```

The third record (flag field 0x74) contains the following control data:

```
00 00 00 03            # Unknown
00 00 00 04            # Number of pieces = 4
00 00 00 02            # Record number 2
00 00 04 2E            # Start offset 1070 bytes
```

```
00 00 07 D0            # 2000 bytes
20 20 20 20 20 20 64 73 2E 6D 69 6E 65 78 74 5F
73 74 67 2C 20 73 2E 6D 69 6E 65 78 74 73 29 2C
0A 20 20 20 20 20 20 64 65 63 6F 64 65 28 62
........
```

The fourth record (flag field 0x74) contains the following control data:

```
00 00 00 03            # Unknown
00 00 00 04            # Number of pieces = 4
00 00 00 03            # Record number 3
00 00 0B FE            # Start offset 3070 bytes
00 00 07 D0            # 2000 bytes
70 72 6F 70 65 72 74 79 2C 20 38 31 39 32 29 2C
20 38 31 39 32 2C 20 27 59 45 53 27 2C 0A 20 20
20 20 20 20 20 20 20 20 20 20 20 64 65 63 6F
........
```

The fifth record (flag field 0x74) contains the following control data:

```
00 00 00 03            # Unknown
00 00 00 04            # Number of pieces = 4
00 00 00 04            # Record number 4
00 00 13 CE            # Start offset 5070 bytes
00 00 07 D0            # 2000 bytes
73 2E 66 6C 61 67 73 5F 73 74 67 2C 20 34 29 2C
20 34 2C 0A 20 20 20 20 20 20 20 20 20 20 63 61
73 65 20 77 68 65 6E 20 62 69 74 61 6E 64 28 64
........
```

In the second example, the view is DBA_TAB_COLUMNS:

```
COPY TO us01/us01@NORTH APPEND t13 USING -
SELECT owner,view_name,text_length,text -
FROM dba_views -
WHERE owner = 'SYS' AND view_name = 'DBA_TAB_COLUMNS';
```

The size of this view is:

```
SQL> SELECT text_length FROM dba_views WHERE view_name = 'DBA_TAB_COLUMNS';

TEXT_LENGTH
-----------
513
```

Therefore this LONG value should fit into a single GoldenGate record.

The first and only record (flag field 0x05) contains the following:

```
00 00 00 07                                       # Column 0 Length 7
   00 00 00 03 53 59 53                           # SYS
00 01 00 13                                       # Column 1 Length 19
   00 00 00 0F 44 42 41 5F 54 41 42 5F 43 4F 4C 55 # DBA_TAB_COLUMNS
   4D 4E 53
00 02 00 07                                       # Column 2 Length 7
   00 00 00 03 35 31 33                           # 513
00 03 02 05                                       # Column 3 Length 517
   00 00 02 01 73 65 6C 65 63 74 20 4F 57 4E 45 52 2C 20
   54 41 42 4C 45 5F 4E 41 4D 45 2C 0A 20 20 20 20 20 20
   20 43 4F 4C 55 4D 4E 5F 4E 41 4D 45 2C 20 44 41 54 41
   ........
```

The first four bytes of the LONG value are the length of the column 00 00 02 01 (0x201) which is 513 decimal.

NULL values are represented by the following 4 bytes:

```
FF FF 00 00
```

ROWID

ROWID columns are unlikely to be useful in a replication environment as there is no guarantee that ROWID values will be equivalent in both the source and target databases. However, it is possible that ROWID columns will be replicated by default or to reduce configuration.

For example the ROWID AAASiCAAEAAAAMVAAD is stored as a 22-byte value in the GoldenGate trail file:

```
00 00 00 12 41 41 41 53 69 43 41 41 45 41 41 41 41 4D 56 41 41 44
```

The first four bytes represent the length of the ROWID (0x12 = 18 bytes). The remaining bytes are the ROWID represented in ASCII.

NULL values are represented by the following 4 bytes:

```
FF FF 00 00
```