After a brief discussion concerning the contents and information held within the DB2 log, details of the syntax and usage of DSN1LOGP will be presented. This will be followed with examples of how to interpret DSN1LOGP output for auditing and recovery purposes.

Acknowledgements / Disclaimers

IBM®, DB2®, z/OS® are registered service marks and trademarks of International Business Machines Corporation, in the United States and/or other countries

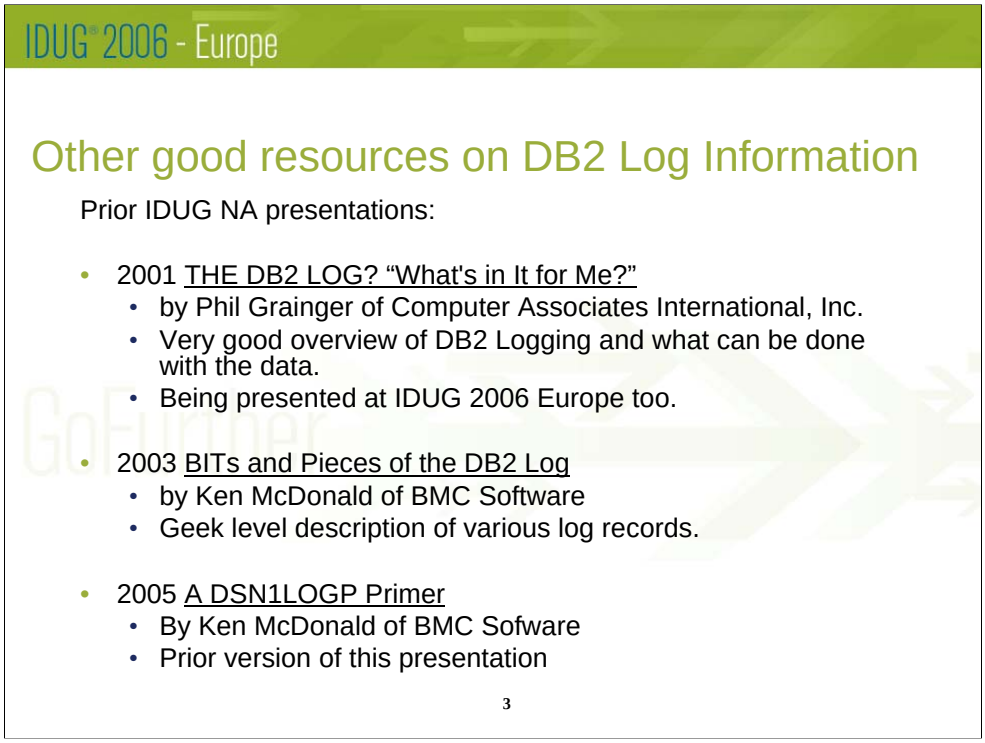
DSN1LOGP / DSNJU004 output and manual / macro references contained are also of IBM.

The information contained in this presentation has not been submitted to any formal review and is distributed on an "As Is" basis. All opinions, mistakes, etc. are my own.

2

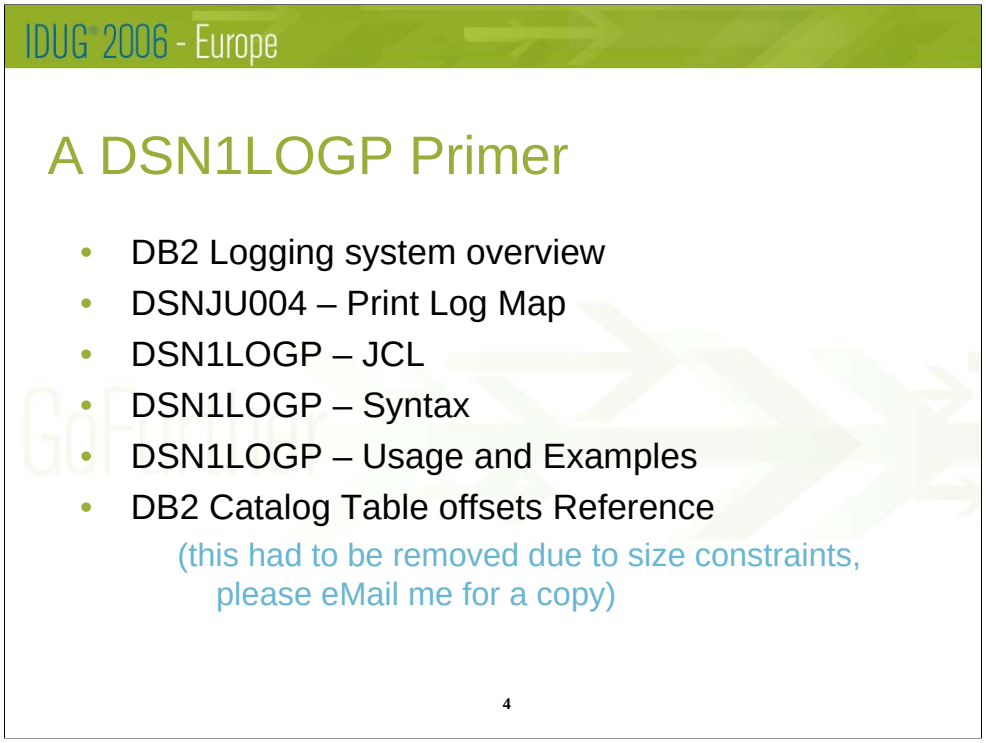
Acknowledgements and Weasel Words.

Most of the contents of this presentation are based upon information from the DB2 UDB for z/OS V8 Utilities Guide and Reference. Examples of DSNJU004 and DSN1LOGP output are included. Some mapping of DSN1LOGP output is based upon the contents of the hlq.DSNMACS(DSNDQJ00) descriptions of DB2 log records and the Diagnosis Guide tables of DB2 Catalog Table column offsets.



I also have a presentation titled "It's RAD – Recovery, Audit, Data migration, plus – Using a DB2 log processing tool". CA has a similar presentation I've seen presented by a couple of people – "Harnessing the Power of Logs".

Oh! And the current title of this presentation is compliments of Phil Grainger… I lifted the "It could save your job one day" from a foil in a DB2 Utilities presentation he did at 2005 NA IDUG.



Bullets of what will be covered in this presentation.

•DB2 Logging system overview
•DSNJU004 – Print Log Map
•DSN1LOGP – JCL
•DSN1LOGP – Syntax
•DSN1LOGP – Usage and Examples
•DB2 Catalog Table offsets Reference

Resources used for information:
DB2 UDB for z/OS V8 Utilities Guide and Reference
DB2 UDB for z/OS V7 and V8 Diagnosis Guide and Reference
Yourhlq.DSNMACS(DSNDQJ00) – Log Record DSECT mapping
Empirical Observation

The Log Records used for this presentation are from DB2 Version 7.1 and 8.1 systems. The references cited are from DB2 Version 8.1.

DB2 Logging System Overview

- BSDS
  - References to logs and the time/RBA ranges they cover
  - References to other members in a data sharing group
  - Archive Log Command, Checkpoint, and Conditional Restart
- ACTIVE Logs
  - Reusable VSAM datasets. Copied to ARCHIVE when full.
  - Limit of 31 in Version 7. Increased to 99 in Version 8.
  - One always current to DB2 logging
- ARCHIVE Logs
  - Sequential files. Can be on DASD or TAPE
  - Represent range (recorded in the BSDS) of log data
  - 1000 Volume limit in DB2 Version 7
  - Increased to 10000 in Version 8

5

All changes to DB2 tables (both user and system) are recorded in the DB2 log for recovery purposes.

The BSDS is the Boot Strap Data Set which is a VSAM data set that contains name and status information for DB2, as well as RBA range specifications, for all active and archive log data sets. It also contains passwords for the DB2 directory and catalog, and lists of conditional restart and checkpoint records.

The ACTIVE logs are VSAM datasets to which DB2 writes to for logging. Only one is in use at a given time. After it fills, DB2 swaps to the next ACTIVE and schedules the newly full log to be archived. After it is archived, it is marked as reusable to be used again by DB2 as an ACTIVE.

The ACTIVE logs are a circular queue and the BSDS records the range of RBAs currently covered by each ACTIVE. The ARCHIVE logs represent a single range of data, as recorded in the BSDS. The oldest ARCHIVE reference is removed from the BSDS when a new ARCHIVE is written and the volume limit has been reached. If an ARCHIVE log spans two or more volumes, there is an entry used in the BSDS for each volume of the archive.

DSNJU004 – Print Log Map

- Use to obtain time and RBA range information and dataset names of both the ACTIVE and ARCHIVE log files
- Also formats other BSDS contained information:
  - Archive Log Command History
  - Checkpoint Records
  - Conditional Restart Records
  - Member Records for other subsystems in the data sharing group
  - Quiesce history
  - BACKUP SYSTEM utility history
  - System CCSID information

6

DSNJU004 is the PRINT LOG MAP utility. It is used to format and print the contents of the BSDS for a single or multiple members of a data sharing group. For the purposes of this presentation, we are concerned about the ACTIVE and ARCHIVE log ranges recorded in the BSDS. However, much more is recorded in the BSDS

DSNJU004 – Example ACTIVE Log Output

```
ACTIVE LOG COPY 1 DATA SETS
  START RBA/LRSN/TIME    END RBA/LRSN/TIME       DATE     LTIME DATA SET INFORMATION
 --------------------  --------------------  --------  ----- --------------------
   00083B440000            00083D5FFFFF       2004.114   2:03 DSN=DSNDHC.DHC1.LOGCOPY1.DS01
     BC8325B9C862            BC83286722DB     PASSWORD=(NULL) STATUS=REUSABLE
 2005.033  14:29:14.7  2005.033  14:41:13.3
   00083D600000            00083F7BFFFF       2004.114   2:03 DSN=DSNDHC.DHC1.LOGCOPY1.DS02
     BC83286722DC            BC832B99FE6C     PASSWORD=(NULL) STATUS=REUSABLE
 2005.033  14:41:13.3  2005.033  14:55:32.0
   00083F7C0000            00084197FFFF       2004.114   2:03 DSN=DSNDHC.DHC1.LOGCOPY1.DS03
     BC832B99FE6D            ............     PASSWORD=(NULL) STATUS=NOTREUSABLE
 2005.033  14:55:32.0  .......  ..........
```

- GMT versus Local Time
- DATE/LTIME of an Active Log is from when the file was created and added to the BSDS (DSNJU003)

7

Most times presented in the PRINT LOG MAP are in GMT. The fields marked LTIME are in local time. The DATE/LTIME on ACTIVE files are related to when the VSAM dataset was defined. The START and END times represent the range of data currently recorded in the corresponding ACTIVE.

The ACTIVE currently being written to will have the periods for the ending LRSN and TIME fields. It will also be marked as NOTREUSABLE.

ACTIVE logs currently being ARCHIVED will also be marked as NOTREUSABLE. If all of your actives are full or truncated by ARCHIVE commands and the archive process is behind such that all ACTIVE logs are marked as NOTREUSABLE, if the last log fills, DB2 processing will stop until at least one archive completes and its ACTIVE is marked as reusable.

Possible Status on ACTIVE files:

•NEW – Added by CHANGE LOG INVENTORY (DSNJU003), but not yet used by DB2

•RESUABLE – Available to be written to by DB2

•NOTREUSABLE – Either the current ACTIVE, or one which is pending archival

•STOPPED – The offload processor encountered an error or an I/O error had occurred against the ACTIVE.

•TRUNCATED – An ACTIVE which when last written to was not full when archived. Truncation can be caused by an ARCHIVE LOG command, a conditional restart, or an I/O error.

DSNJU004 – Example ARCHIVE Log Output

```
ARCHIVE LOG COPY 1 DATA SETS
  START RBA/LRSN/TIME   END RBA/LRSN/TIME      DATE    LTIME DATA SET INFORMATION
 --------------------  --------------------  --------  ----- --------------------
   000834F00000           0008370BFFFF       2005.031  13:59 DSN=DSNDHC.DHC1.ARCLOG1.A0000997
     BC80EA7B2D8C           BC80EBFDE6E3                    PASSWORD=(NULL) VOL=ARC007
    UNIT=SYSALLDA
 2005.031  19:53:32.4  2005.031  20:00:17.9
                                                           CATALOGUED
   0008370C0000           00083927FFFF       2005.033   7:19 DSN=DSNDHC.DHC1.ARCLOG1.A0000998
     BC80EBFDE6E4           BC83162B6929                    PASSWORD=(NULL) VOL=ARC022
    UNIT=SYSALLDA
 2005.031  20:00:17.9  2005.033  13:19:38.9
                                                           CATALOGUED
   000839280000           00083B43FFFF       2005.033   8:28 DSN=DSNDHC.DHC1.ARCLOG1.A0000999
     BC83162B692A           BC8325B9C861                    PASSWORD=(NULL) VOL=ARC025
    UNIT=SYSALLDA
 2005.033  13:19:38.9  2005.033  14:29:14.7
                                                           CATALOGUED
```

- GMT versus Local Times
- DATE/LTIME for Archive Logs are from when the file was created

8

Again, most times presented in the PRINT LOG MAP are in GMT. And the fields marked as LTIME are in local time. The DATE/LTIME on ARCHIVE files are related to when the archive dataset was defined and should be fairly close to the END TIME of the range recorded on the corresponding archive.

This example comes off of a z/OS system with a 6 hours West of GMT offset. Note the LTIMEs are about 6 hours less than the corresponding GMT values.

Keep the GMT versus LTIME in mind when searching for logs to include in a DSN1LOGP attempt.

DSNJU004 – Example multi volume ARCHIVE

```
ARCHIVE LOG COPY 1 DATA SETS
  START RBA/LRSN/TIME   END RBA/LRSN/TIME      DATE    LTIME DATA SET INFORMATION
 --------------------  --------------------  --------  ----- --------------------
   07A1B25C1000           07A1ED74AFFF       2005.051  20:18 DSN=LOG.LOG06922.D3E1.D05051.T2015084.A0007794
   BC99B97BD6DC           BC99FE82EB0F                     PASSWORD=(NULL) VOL=L35136 UNIT=VCART
  2005.051  13:27:26.7  2005.051  18:36:16.1
                                                            CATALOGUED
   07A1ED74B000           07A1FB388FFF       2005.051  20:18 DSN=LOG.LOG06922.D3E1.D05051.T2015084.A0007794
   BC99FE82EB10           BC9A0732FDE9                     PASSWORD=(NULL) VOL=L38167 UNIT=VCART
  2005.051  18:36:16.1  2005.051  19:15:08.2
                                                            CATALOGUED
```

- GMT versus Local Times – Note LTIME is near basically the Active Full time plus offload overhead time
- Multi volume considerations:
  - How many blocks on the second volume?
  - Are you reducing the amount of log referenced in your BSDS by having small second volume entries?

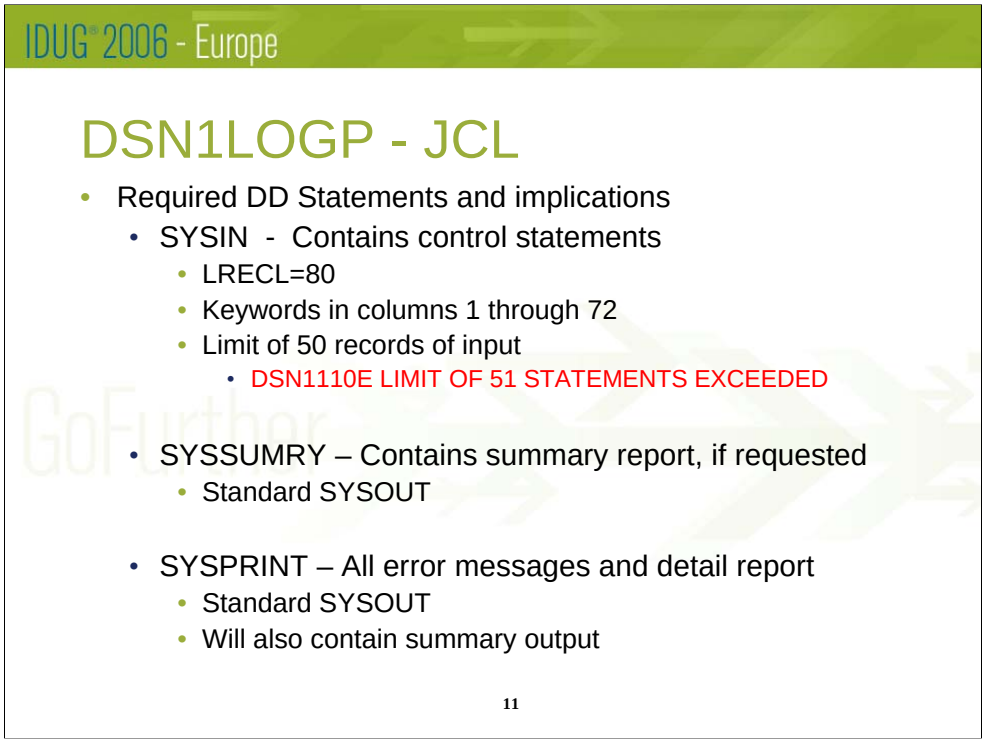
9

Remember that in Version 7 and prior there was a limit of 1,000 archive volume entries in the BSDS. Taking the scenario where your ACTIVE log files are defined to hold 10,000 pages of log. But, the archive media has a limit of 9,000 pages per volume. Each ARCHIVE will take two volume entries with the first covering 9,000 pages and the second 1,000 pages. Given that only 5,000,000 pages of log will be covered by the ARCHIVE chain. If you reduce the ACTIVE log file size to be 9,000 pages to match the archive media limitation to use one volume entry per archive… the amount of log recorded in the BSDS will be increased to 9,000,000 pages.

DSN1LOGP is the IBM utility to format the contents of the recovery log. Two types of reports are available… both a detail and summary.

The detail report allows you to look at individual log records.

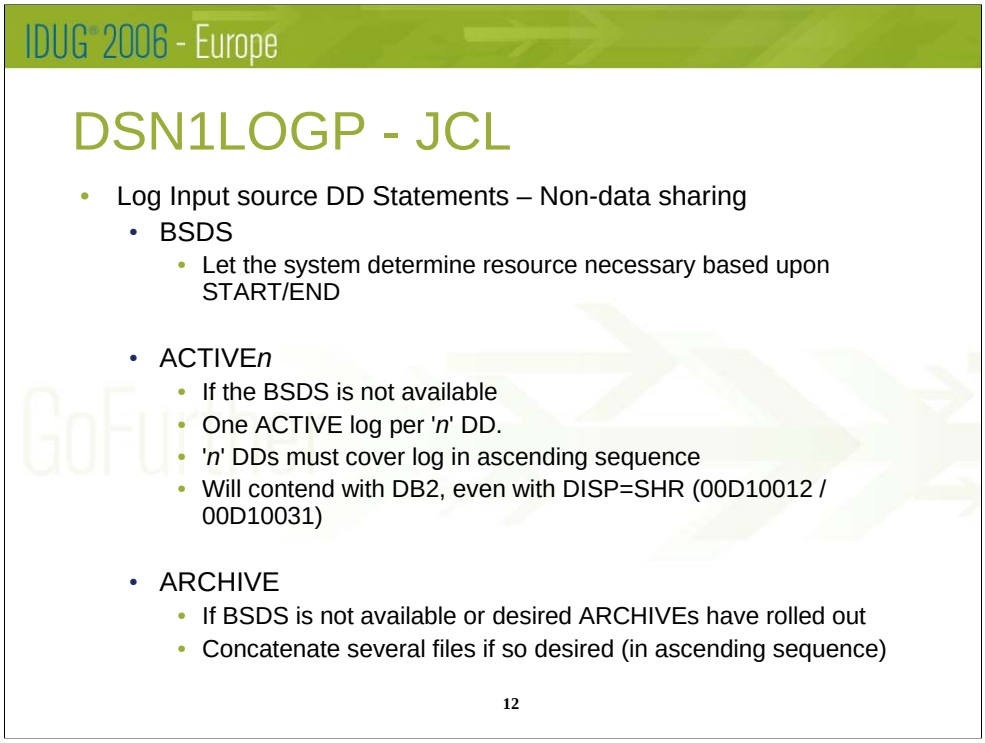
The summary report allows you to see which DB2 objects are touched by any given unit of recovery.



Pretty much all on the slide…

I just want to stress that the SUMMARY report output goes to both the SYSPRINT and SYSSUMRY DD if SUMMARY(YES) specified.

The SUMMARY report is formatted only in SYSSUMMARY if SUMMARY(YES) is specified.



You can use these DDs on data sharing systems as well to limit output to a single member of the data sharing group.

On non-data sharing systems the ACTIVEs are not accessible to DSN1LOGP even with DISP=SHR and appropriate share options. DB2 opens ACTIVE files on non-data sharing with an exclusive lock.

With BSDS input, the following message will be produced, but DSN1LOGP will continue if an ARCHIVE for the range in question is available:

```
DSN1222E DSNJSLR ERROR RETCODE=0004 REASON CODE= 00D10031
         DYNAMIC ALLOCATION INFORMATION CODE=0000 ERROR CODE=000C
```

```
For the current active with no corresponding ARCHIVE
for both non-data and data sharing:
```

```
DSN1210E DSNJSLR ERROR RETCODE=00000008 REASON CODE = 00D10012
```

DSN1LOGP - JCL

- Log Input source DD Statements – Data sharing
  - GROUP
    - Synonymous with BSDS DD for a non-data sharing system
    - Will print log records from all members of the group
  - M*xx*BSDS
    - Will print log records from only members with BSDS DDs
    - Will merge appropriately
  - M*xx*ACT*n*
    - Synonymous with ACTIVEn
    - Will merge appropriately
  - M*xx*ARCHV
    - Synonymous with ARCHIVE
    - Will merge appropriately

13

Specifying a single Mxx type DD is the equivalent of specifying the non-data sharing DDs.

By “merge appropriately”, I mean that log information from different members will be interspersed in the SYSPRINT and SYSSUMRY output in appropriate LRSN sequence.

DSN1LOGP – JCL, all together now

```
//RDAKMMLR JOB (acct),'DSN1LOGP',
//         CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//*************************************************
//JS010    EXEC PGM=DSN1LOGP,REGION=0M
//STEPLIB  DD DISP=SHR,DSN=SYS3.DECI.DSNEXIT
//         DD DISP=SHR,DSN=CSGI.DB2V81M.DSNLOAD
//BSDS     DD DISP=SHR,DSN=DECICAT.BSDS01
//*GROUP   DD DISP=SHR,DSN=DSNDHA.DHA1.BSDS01
//*ARCHIVE DD DISP=SHR,DSN=DECICAT.ARCLOG1.A0014636
//SYSIN    DD *
   STARTRBA(0066A3CCFB2F) ENDRBA(0066B81792C9)
   DBID(02AD) OBID(00C7) RID(000000148F)
/*
//SYSPRINT DD SYSOUT=*
//SYSSUMRY DD SYSOUT=*
//
```

14

Here's my standard JCL… I modify and uncomment or comment GROUP or BSDS depending upon whether or not I'm working on a non-data sharing system or want only a single member's output for a data sharing group. I also have samples of all the DSN1LOGP keywords following the final JCL line to copy into the SYSIN area and modify to fit my needs for that execution.

DSN1LOGP - Syntax

- **RBASTART**(hex-constant) / **LRSNSTART**(hex-constant)
  - Defaults to x'000000000000'
- **RBAEND** (hex-constant) / **LRSNEND** (hex-constant)
  - Defaults to x'FFFFFFFFFFFF'
- Comments
  - RBA and LRSN are mutually exclusive
  - Specify a value… defaulting to everything can produce lots of output
  - Only consider not specifying if printing a specific log file
    - (`//ARCHIVE DD`)
  - LRSN must be used for BSDS multiple Member processing

15

Leading zeroes do not need to be specified for RBASTART and RBAEND. Leaving out any of the 12 digits for the LRSN values leads to right justification with zeroes in the upper digits of the LRSN field and subsequent errors.

DSN1LOGP - Syntax

- **DATA ONLY**(NO/YES)
  - Include only records reflecting changes to data
  - Defaults to NO
  - YES can reduce the size of your output by suppressing marginally interesting records.
- **SUMMARY**(NO/YES/ONLY)
  - Include URID SUMMARY information in output
  - Defaults to NO (YES can greatly increase the size of your output)
  - SUMMARY(YES) data is in both SYSPRINT and SYSSUMRY
- **FILTER** (sub-keyword of SUMMARY)
  - Restrict to only specified URIDs and LUWIDs
  - Must appear after SUMMARY keyword

16

DATAONLY(YES) will cause PAGESET type log records to be suppressed (e.g. OPEN, WRITE)

SUMMARY(NO) default is good. Don't specify SUMMARY(YES) unless you *really* want to produce a SUMMARY output.

SUMMARY(ONLY) will go to SYSSUMRY only and SYSPRINT will contain the SYSIN control cards and processing messages.

DSN1LOGP – Sample output SUMMARY(YES)

```
DSN1151I DSN1LPRT UR  CONNID=DB2CALL   CORRID=RDACAL2Z  AUTHID=RDACAL2 PLAN=ASU620DS
  START DATE=05.033 TIME=16:45:44  DISP=COMMITTED          INFO=COMPLETE
  STARTRBA=024317787C86  ENDRBA=0243177884EC  STARTLRSN=BC8394C81557
    ENDLRSN=BC8394C81B1B
  NID=*  LUWID=USBMCN01.DEBALU.BC8394C68CE2.0001
  COORDINATOR=*  PARTICIPANTS=*
  DATA MODIFIED:
       DATABASE=0A75=BMCASU62  PAGE SET=0010=BMCRSIX
       DATABASE=0A75=BMCASU62  PAGE SET=0050=DXRSIX
       DATABASE=0A75=BMCASU62  PAGE SET=005E=DXRSIX2
       DATABASE=0A75=BMCASU62  PAGE SET=0068=DXRSIX3
       DATABASE=0A75=BMCASU62  PAGE SET=0024=BMCRSCD
       DATABASE=0A75=BMCASU62  PAGE SET=006A=DXRSDIX
       DATABASE=0A75=BMCASU62  PAGE SET=000E=BMCRSIP
       DATABASE=0A75=BMCASU62  PAGE SET=004E=DXRSIP
```


The fields listed in the first 4 lines are directly from the BEGIN UR record. They identify who what how…

- CONNID – Connection ID – BATCH, DB2CALL, UTILITY, etc.
- CORRID – Job name, TSO UserID, CICS or IMS Thread, etc.
- AUTHID – The PRIMARY AuthID of the user making the call
- PLAN – The PLAN executed for these changes. The PACKAGE is not recorded in the BEGIN UR record
- DISP – COMMITTED, ABORTED, INFLIGHT, INDOUBT, etc.
- STARTRBA – Is the RBA of the BEGIN UR record. Use this value in the URID(xxxxxxxxxxxx) syntax
- ENDRBA – Is the RBA of the END COMMIT record associated with the Unit of Recovery
- STARTLRSN/ENDLRSN - LRSN associated with the two prior RBAs
- LUWID – Logical Unit of Work ID… three part ID usable in the LUWID(value) syntax

DSN1LOGP – Line count comparison

20 Million bytes of log included in scan range

- SUMMARY(YES) DATAONLY(NO) - 1,215,710 lines
- SUMMARY(NO) DATAONLY(NO) - 1,158,151 lines
- SUMMARY(NO) DATAONLY(YES) - 782,644 lines

I had actually updated 19 rows in my table in this range

- DBID(0210) SUMMARY(YES) - 65,465 lines
- SUMMARY(NO) DATAONLY(NO) - 7,905 lines
- SUMMARY(NO) DATAONLY(YES) - 4,050 lines
- Added OBID(0002) (space records only) - 2,123 lines

The purpose of this foil is to stress a couple of points…

ONLY SPECIFY SUMMARY(YES) if you absolutely want SUMMARY information. The default of NO is good.

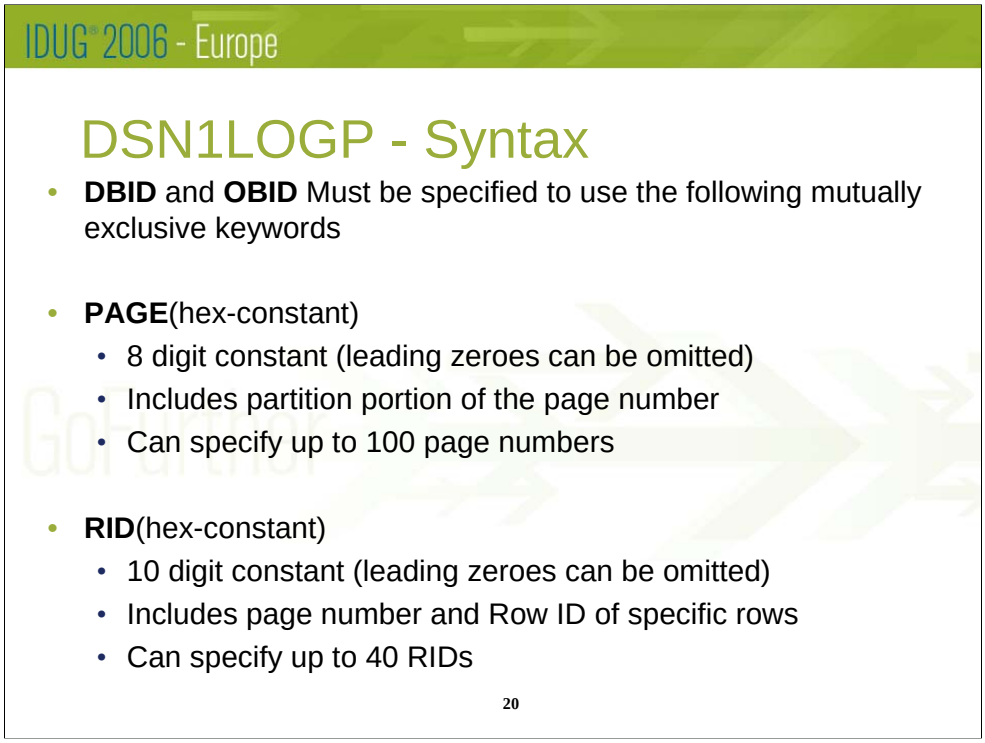
SPECIFY DATAONLY(YES) if you are not interested in page set open/close/write type records to reduce your output.

DSN1LOGP - Syntax

- **DBID**(hex-constant)
  - 1 to 4 characters… leading zeros not necessary
  - Only one DBID can be specified per execution
- **OBID**(hex-constant)
  - 1 to 4 characters… leading zeros not necessary
  - Table space PSID or Index space ISOBID
  - Only one OBID can be specified per execution
  - DBID is required to use OBID in the syntax

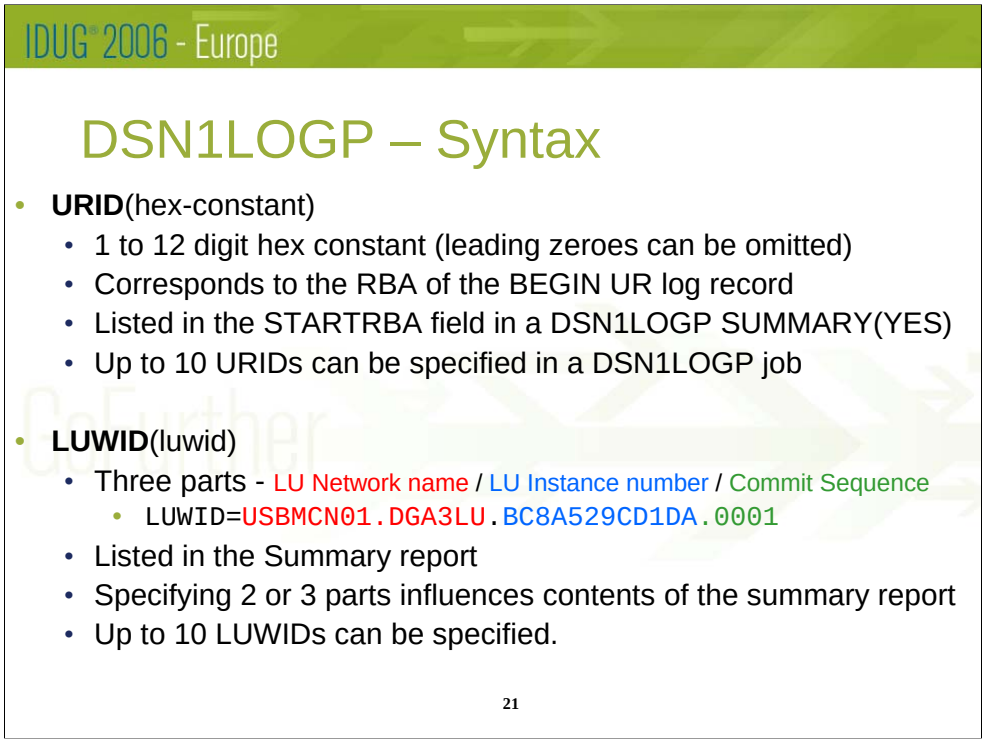
19

Obtain the DBID and OBID values from SELECTs against SYSIBM.SYSDATABASE, SYSIBM.SYSTABLESPACE, and/or SYSIBM.SYSINDEXES. Be sure to obtain HEX values or convert the decimal to HEX before plugging into DSN1LOGP SYSIN.



Reminder that there is a 50 line limit on SYSIN input, so to be able to specify 100 page numbers, one would have to put multiple PAGE keywords on any given line.

I personally prefer not to omit leading zeroes. I'm usually cut and pasting anyway and getting all 8 or 10 digits is a nit. The Professor of my first computer programming class in College used to say “Parenthesis are free” in relation to complex IF statements. The same applies here… “Leading zeroes are free”.



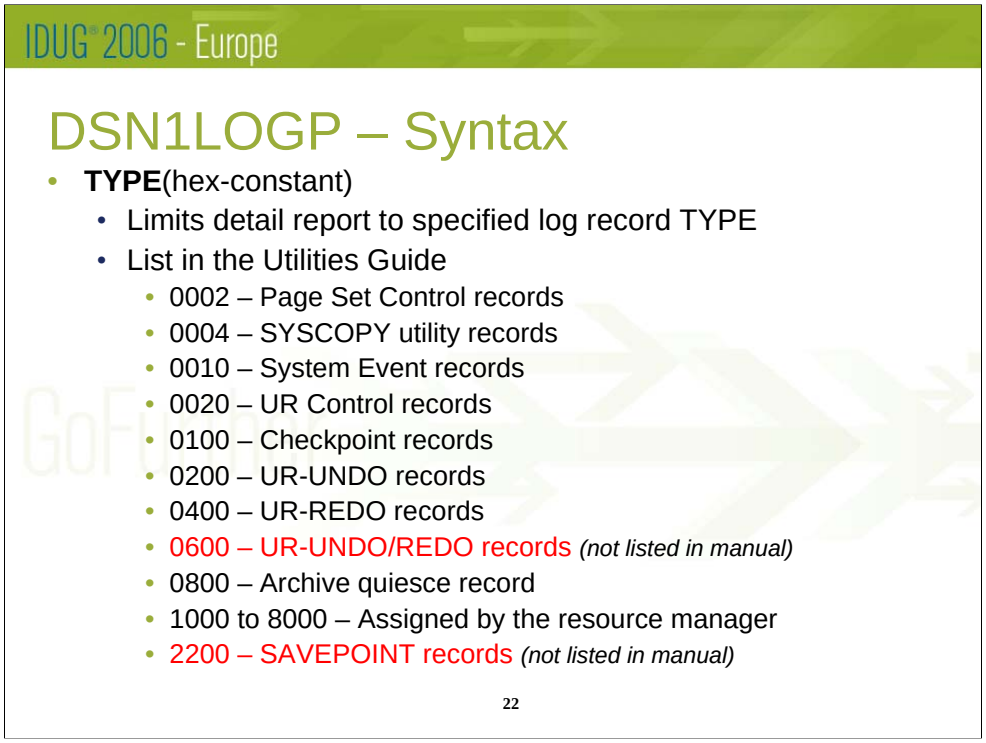
The LU Network name consists of a one to eight character network ID followed by a period and a one to 8 character LU name.

The LU Instance consists of 12 hexadecimal characters (Seems to be timestamp related in some instances)

The Commit Sequence number a 4 hexadecimal character field usually starting with 0001 and incrementing from there.

From a SYSSUMRY output it looks like 4 parts… but the first two are the LU Network name:

LUWID=USBMCN01.DGA3LU.BC8A529CD1DA.0001



The types listed in the Utilities manual are in black. The 0600 type is common on the log where both the UNDO and REDO portion of the change (INSERT, UPDATE, DELETE) are contained in a single log record.

Some records are a combination of types… For example, SAVEPOINT records have TYPE(2200).

DSN1LOGP – Syntax

- **SUBTYPE**
  - Restricts to SUBTYPE records for TYPE(0200) and (0400)
  - Mutually exclusive with the TYPE keyword
  - List in the Utilities Guide *(too many to include on the slide)*:
    - 01 – Update data page
    - 02 – Format page or update space map
    - 03 – Update space map bits
    - 04 – Update to index space map
    - 05 – Update to index page
    - 06 – DBA table update log record
    - … (see notes)
  - Is everyone practicing Safe DB2?
    - Subtypes 0F and 10 for REPAIR
    - Subtype 82 for START FORCE

23

09 – DBD virtual memory copy

0A – Exclusive lock on page set partition or DBD

0B – Format file page set

0C – Format index page set

0F – Update by repair (first half if 32KB)

10 – Update by repair (second half if 32KB)

11 – Allocating or deallocating a segment entry

12 – Undo/redo log record for modified page or redo log record for formatted page

14 – Savepoint

15 – Other DB2 component log records written for RMID 14

17 – Checkpoint record of a modified page set

19 – Type 2 Index update

1A – Type 2 Index undo/redo or redo log record

1B – Type 2 Index change notification log record

1C – Type 2 Index space map update

1D – DBET log record with exception data

1D – DBET log record with LPL/GRECP data

65 – Data propagation diagnostic log record

81 – Index dummy compensation log record

82 - START DATABASE ACCESS (FORCE) log record

DSN1LOGP - Syntax

- **OFFSET**(hex-constant)
  - Hex constant of a maximum of 8 characters
- **VALUE**(hex-constant)
  - Hex constant of a maximum of 64 characters
  - Must be an even number of characters
- Comments
  - SUBTYPE is required to specify OFFSET/VALUE
  - OFFSET/VALUE must appear in pairs
  - Maximum of 10 pairs can be specified
  - Multiple OFFSET/VALUE treated as ORs
  - **Difficult to use on compressed spaces, columns following variable length rows, and partially logged updates**
  - **V9 – Reordered Row Format**

24

Use of the OFFSET/VALUE can be fairly powerful… But, you need to be aware of the format of the records you are searching and what you are specifically searching for. For user data which is compressed or follows VARCHAR or other variable fields on the row, success at using VALUE/OFFSET may be dodgy at best… But, you have 10 pairs where you can twiddle the offset.

Another point is Version 9 changes user data to Reordered Row Format which will make determining the column offset of user data even more difficult. Luckily, the DB2 Catalog remains in Basic Row Format, as it is today.

DSN1LOGP – Syntax

- **SYSCOPY**(NO/YES)
  - YES limits detail report to only SYSCOPY log records for the events against tablespaces not recorded in SYSCOPY (SYSCOPY, DBD01, SYSLGRNX)
  - Default is NO.
  - SYSCOPY Data remains in EBCDIC in DB2 Version 8

```
0243317FDC93  LRSN(BC842081A9CB)
              TYPE(SYSCOPY UTILITY)
*LRH* 00DA017E 00040010 15800000 00000000 02433163 E93B0426 02433163 E93BBC84
      2081A9CB 0000
0000  00000000 C4E2D5C4 C2F0F640 E2E8E2C3 D6D7E840 00000000 C6F0F5F0 F2F0F302
0020  433173FA 6C000000 17F3F4F9 F0404040 40D5C3E2 C74BC4C5 C2C14BC4 E2D5C4C2
0040  F0F64BE2 E8E2C3D6 D7E84BC4 F0F5F0F2 F0F34BE3 F0F3F1F0 F4F74040 4040F0F3
0060  F1F0F5F1 C3000020 05020303 10511757 214040E3 40000000 00000040 40404040
0080  404040E3 00000000 00000000 44151800 00000000 44151800 00000000 42510000
00A0  00000000 C4C5C2C1 C3D6D7E8 C2D4C3C1 C4D44040
*   =               Z       Z  d
* az
*    DSNDB06 SYSCOPY     F050203
*    %    3490    NCSG.DEBA.DSNDB
*06.SYSCOPY.D050203.T031047    03
*1051C             T
*   T
*    DEBACOPYBMCADM
```

25

The Highlighted “F” was the ICTYPE… this was a full copy of DSNDB06.SYSCOPY

DSN1LOGP – Syntax

- **CHECK**(DATA)
  - Added in Version 7 of DB2
  - Specifies that the utility checks the specified range of pages for page regression
  - Every page has a PGLOGRBA in the page header containing the RBA or LRSN of the last update to the page
  - Data Management log records have in their DM Subheader a field which contains the previous PGLOGRBA before this update
- PQ90432 – Serviceability Enhancement
  - Added INDEX and ALL as options to the CHECK syntax

Looks like a pretty cool report to determine if someone is not practicing safe DB2. There is an example in the manual, but I did not produce a situation to force data into this report.

(Might have been APARed into Version 6 based upon some APARs I saw… the -0 version of the Version 6 Utilities Guide did not have this in the syntax.)

DSN1LOGP – Usage and Examples

- Who updated that table that's not supposed to be updated?
- Who DROPPED that DATABASE?
- Did BADUSER update anything?
- Are GRANTs and REVOKEs being done outside of our control?
- Who FREEd that PLAN or PACKAGE?
- Are SAVEPOINTs being executed on this subsystem?
- Can I find a common quiet point for two tablespaces for Recovery?

DSN1LOGP – Who updated that table?

Prep

- Determine DBID and PSID of the tablespace.
  - KMMSEGDB.KMMSEGS – RDAKMM.KMMSEGT3
  - DBID(0600) PSID(0002)
- If a multi-table tablespace, determine table OBID
  - OBID(000D)

Method

- SUMMARY(ONLY) – Look for updates to DBID/PSID
- DETAIL – Look for updates to DBID/PSID/OBID
- If updates happened… two methods
  - who/what/how in associated URID SUMMARY
  - STARTRBA(urid rba) ENDRBA(urid rba) to see BEGIN UR record

From SPUFI:

```
---------+---------+---------+---------+---------+---------+---------+--
  SELECT DBNAME, NAME, HEX(DBID), HEX(PSID)
    FROM SYSIBM.SYSTABLESPACE
      WHERE DBNAME = 'KMMSEGDB' AND NAME = 'KMMSEGS';
---------+---------+---------+---------+---------+---------+---------+--
DBNAME    NAME
---------+---------+---------+---------+---------+---------+---------+--
KMMSEGDB  KMMSEGS   0600  0002
DSNE610I NUMBER OF ROWS DISPLAYED IS 1
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+--

  SELECT DBNAME, TSNAME, CREATOR, NAME, HEX(DBID), HEX(OBID)
    FROM SYSIBM.SYSTABLES
      WHERE DBNAME = 'KMMSEGDB' AND TSNAME = 'KMMSEGS';
---------+---------+---------+---------+---------+---------+---------+--
DBNAME    TSNAME    CREATOR   NAME
---------+---------+---------+---------+---------+---------+---------+--
KMMSEGDB  KMMSEGS   RDAKMM    KMMSEGTB            0600  0003
KMMSEGDB  KMMSEGS   RDAKMM    KMMSEGT2            0600  0008
KMMSEGDB  KMMSEGS   RDAKMM    KMMSEGT3            0600  000D
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
DSNE616I STATEMENT EXECUTION WAS SUCCESSFUL, SQLCODE IS 100
---------+---------+---------+---------+---------+---------+---------+--
```

DSN1LOGP – Is this table being updated?

```
STARTRBA(024A96811954) ENDRBA(024AB15B8FFF)
DBID(0600) OBID(0002)
DATAONLY(YES)

STARTRBA(024A96811954) ENDRBA(024AB15B8FFF)
DBID(0600) OBID(0002)
DATAONLY(YES)
SUBTYPE(01)
OFFSET(003C) VALUE(000D)

024AAC993D20  URID(024AAC993C24)  LRSN(BCA41AA03C4D)  DBID(0600)
              OBID(0002)  PAGE(00000B88)  TYPE( UNDO  REDO )
              SUBTYPE(UPDATE IN-PLACE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIREPR)
*LRH* 00460046 06000001 0E80024A AC993C24 024AAC99 3D200426 024AAC99 3D20BCA4
      1AA03C4F 0000
*LG** 88060000 0200000B 8800024A AC993D20 2D00
0000  000E0107 000D8200 011B04D2 025A
```


The first STARTRBA/ENDRBA control cards returned THOUSANDS of lines of output because this segmented space had tables which were being “validly” updated. So, I further qualified with an OFFSET/VALUE pair looking for the OBID of the table which should not have been updated.

Offset derived by *LRH* being an x'26' bytes. The *LG** being x'12' bytes. And the OBID is at offset 4 into the Header portion of the DM (Data Management) record. So… add those together and you get offset x'003C'

Much of the DB2 Log content format is documented in hlq.DSNMACS(DSNDQJ00)

Shameless plug… my prior IDUG presentation – BITs and Pieces of the DB2 Log provides detail mapping of these type records. Send me an eMail if you'd like a copy.

DSN1LOGP – URID for errant update

```
STARTRBA(024AAC993C24) ENDRBA(024AAC993C24)


024AAC993C24  URID(024AAC993C24)  LRSN(BCA41AA03C4C)
              TYPE(UR CONTROL)  SUBTYPE(BEGIN UR)
*LRH* 00900034 00200001 0380024A AC993C24 00000000 00000426 00000000 0000BCA4
      1AA03C4C 0000
0000  003C0000 0000D000 00000000 00000700 0000D9C4 C1D2D4D4 40404040 4040D9C4
0020  C1D2D4D4 4040BCA4 1AA03C4C 9812C4E2 D5C5E2D7 C3E2C2C1 E3C3C840 4040E3E2
0040  D6404040 40400000 00000000 0000001A 0001E4E2 C2D4C3D5 F0F1C4C5 C2C1D3E4
0060  4040BCA4 1AA0273A 0001
*                   RDAKMM       RD
*AKMM    u   <q DSNESPCSBATCH    TS
*O                  USBMCN01DEBALU
*   u
```


I'll be using SUMMARY output from this point forward… But, this is what a BEGIN UR record looks like in the log. The information contained within is used to populate the non RBA/LRSN fields of the DSN1LGOP SUMMARY report.

The PLAN in this UR was SPUFI with Cursor Stability.

DSN1LOGP – Who DROPPED that DATABASE?

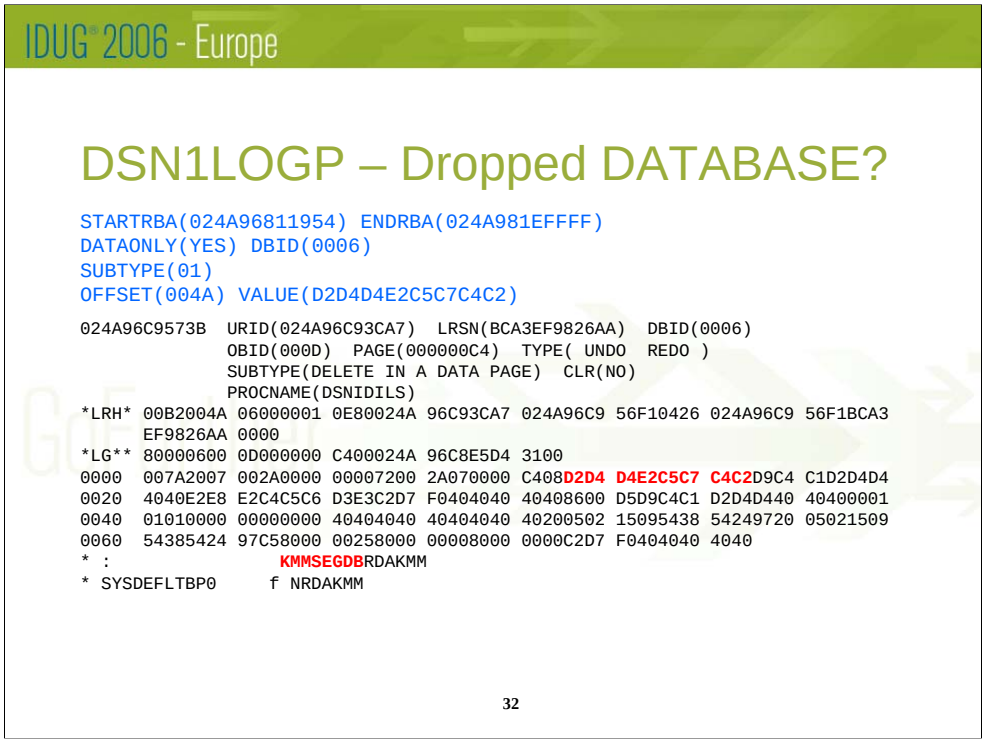
Prep

- Determine DBID/PSID/OBID of SYSIBM.SYSDATABASE
  - In tablespace DSNDB06.DSNDBAUT
  - DBID 0006, PSID 000D, OBID 002A
- Determine DATABASE name of interest
- Determine Range

Methods to look for DELETE from SYSIBM.SYSDATABASE

- DETAIL on DBID(0006) PSID(000D)
- DETAIL on DBID(0006) SUBTYPE(01) VALUE/OFFSET

SYSIBM.SYSDATABASE is in DSNDB06.DSNDBAUT. A similar select from the DB2 Catalog can be done as we did for the user tablespace.



This is on from a DB2 Version 7 subsystem. If you were to do the same thing on an New Function Mode (or enabling with DSNDBAUT converted) DB2 Version 8 subsysem, the offset would change and the VALUE would have to be specified in UNICODE.

Offset determined by adding *LRH* length of x'26', *LG** length of x'12', DM Header length of x'08', DB2 Row header length of x'06, and on 'LINK' fullword in the SYSDBASE row (per the diagnosis guide). The database NAME is the first column of the SYSDATABASE row… so, there you go.

DSN1LOGP – DROP DATABASE URID

```
STARTRBA(024A96C93CA7) ENDRBA(024A981EFFFF)
SUMMARY(ONLY) FILTER
URID(024A96C93CA7)
```

```
DSN1151I DSN1LPRT UR  CONNID=BATCH     CORRID=RDAKMMCR      AUTHID=RDAKMM    PLAN=BMCTEP2
    START DATE=05.059 TIME=10:22:24  DISP=COMMITTED          INFO=COMPLETE
    STARTRBA=024A96C93CA7 ENDRBA=024A96C9613C STARTLRSN=BCA3EF9801D9 ENDLRSN=BCA3EF982855
    NID=*  LUWID=USBMCN01.DEBALU.BCA3EF924DD9.0001
```

DSN1LOGP – Did BADUSER do any updates?

Prep

- Determine BADUSER's primary authid

Method

- SUMMARY(ONLY) for range. Search for BADUSER.
- Cannot use VALUE/OFFSET (which is for TYPE(0200/0400) records). URID records are TYPE(0020). SUMMARY is the only option.

34

The PRIMARY AUTHID used for the thread attach is recorded in the BEGIN UR record, regardless of SET SQLID statements executed. So, to look for BADUSER behavior, you must know their PRIMARY logon ID.

Some DSN3@ATH or DSN3@SGN alter the PRIMARY AUTHID as part of the connection process, which may add a level of difficulty for identifying a specific user.

And, some multi tier ERP applications connect all users from the server to DB2 as a single AUTHID for which at this point there is not a way to easily distinguish who was responsible for what in the log.

DSN1LOGP – BADUSER activity example

```
STARTRBA(024AC2EF9000) ENDRBA(024AC6766FFF)
SUMMARY(ONLY)
```

Small range… 7000+ lines of output

```
DSN1151I DSN1LPRT UR  CONNID=TSO    CORRID=BADUSER    AUTHID=BADUSER     PLAN=DSNESPCS
   START DATE=05.059 TIME=14:29:20  DISP=COMMITTED          INFO=COMPLETE
   STARTRBA=024AC666C4C7 ENDRBA=024AC666C76B STARTLRSN=BCA426C98AED ENDLRSN=BCA426C98C3E
   NID=*  LUWID=USBMCN01.DEBALU.BCA4267B984C.0001
   COORDINATOR=*  PARTICIPANTS=*
   DATA MODIFIED:
        DATABASE=0600=KMMSEGDB  PAGE SET=0002=KMMSEGS
```

35

Could be prohibitive to investigate over a large range of log…

DSN1LOGP – Look for GRANTs and/or REVOKEs

Prep - Determine DBID/PSID/OBID of Authorization Tables

- SYSIBM.SYSCOLAUTH - Table (Column) privileges
- SYSIBM.SYSTABAUTH - Table privileges
- SYSIBM.SYSDBAUTH - Database privileges
- SYSIBM.SYSRESAUTH - Collection/Use privileges
- SYSIBM.SYSROUTINEAUTH – Function/Procedure privileges
- SYSIBM.SYSSCHEMAAUTH - Schema privileges
- SYSIBM.SYSSEQUENCEAUTH- Sequence privileges
- SYSIBM.SYSPACKAUTH - Package privileges
- SYSIBM.SYSPLANAUTH - Plan privileges
- SYSIBM.SYSUSERAUTH - System privileges


```
---------+---------+---------+---------+---------+---------+---------+--
  SELECT DBNAME, NAME, HEX(DBID), HEX(PSID)
    FROM SYSIBM.SYSTABLESPACE
      WHERE DBNAME = 'DSNDB06'
        AND NAME IN ( 'SYSDBASE', 'SYSDBAUT', 'SYSGPAUT', 'SYSOBJ',
                      'SYSPKAGE', 'SYSPLAN', 'SYSUSER' )
        ORDER BY DBNAME, NAME;
---------+---------+---------+---------+---------+---------+---------+--
DBNAME    NAME
---------+---------+---------+---------+---------+---------+---------+--
DSNDB06   SYSDBASE  0006  0009
DSNDB06   SYSDBAUT  0006  000D
DSNDB06   SYSGPAUT  0006  000E
DSNDB06   SYSOBJ    0006  0120
DSNDB06   SYSPKAGE  0006  0079
DSNDB06   SYSPLAN   0006  000A
DSNDB06   SYSUSER   0006  000F
DSNE610I NUMBER OF ROWS DISPLAYED IS 7
---------+---------+---------+---------+---------+---------+---------+--
SELECT DBNAME, TSNAME, CREATOR, NAME, HEX(DBID), HEX(OBID)
    FROM SYSIBM.SYSTABLES
      WHERE DBNAME = 'DSNDB06' AND NAME LIKE 'SYS%AUTH'
        ORDER BY DBNAME, TSNAME, NAME;
---------+---------+---------+---------+---------+---------+---------+--
DBNAME    TSNAME    CREATOR   NAME
---------+---------+---------+---------+---------+---------+---------+--
DSNDB06   SYSDBASE  SYSIBM    SYSCOLAUTH          0006  0020
DSNDB06   SYSDBASE  SYSIBM    SYSTABAUTH          0006  001C
DSNDB06   SYSDBAUT  SYSIBM    SYSDBAUTH           0006  002B
DSNDB06   SYSGPAUT  SYSIBM    SYSRESAUTH          0006  002C
DSNDB06   SYSOBJ    SYSIBM    SYSROUTINEAUTH      0006  013E
DSNDB06   SYSOBJ    SYSIBM    SYSSCHEMAAUTH       0006  0139
DSNDB06   SYSPKAGE  SYSIBM    SYSPACKAUTH         0006  008F
DSNDB06   SYSPLAN   SYSIBM    SYSPLANAUTH         0006  0025
DSNDB06   SYSUSER   SYSIBM    SYSUSERAUTH         0006  002D
DSNE610I NUMBER OF ROWS DISPLAYED IS 9
```

DSN1LOGP – Look for GRANTs and/or REVOKEs

Method

- Search for DBID(0006) for catalog changes
- VALUE/OFFSET search for Auth table updates
- Analyze details

Pitfalls/Complications

- Partially logged updates
- GRANTEETYPE = 'P' for plans and packages
- Extra information (DBID(0006)) or multiple runs

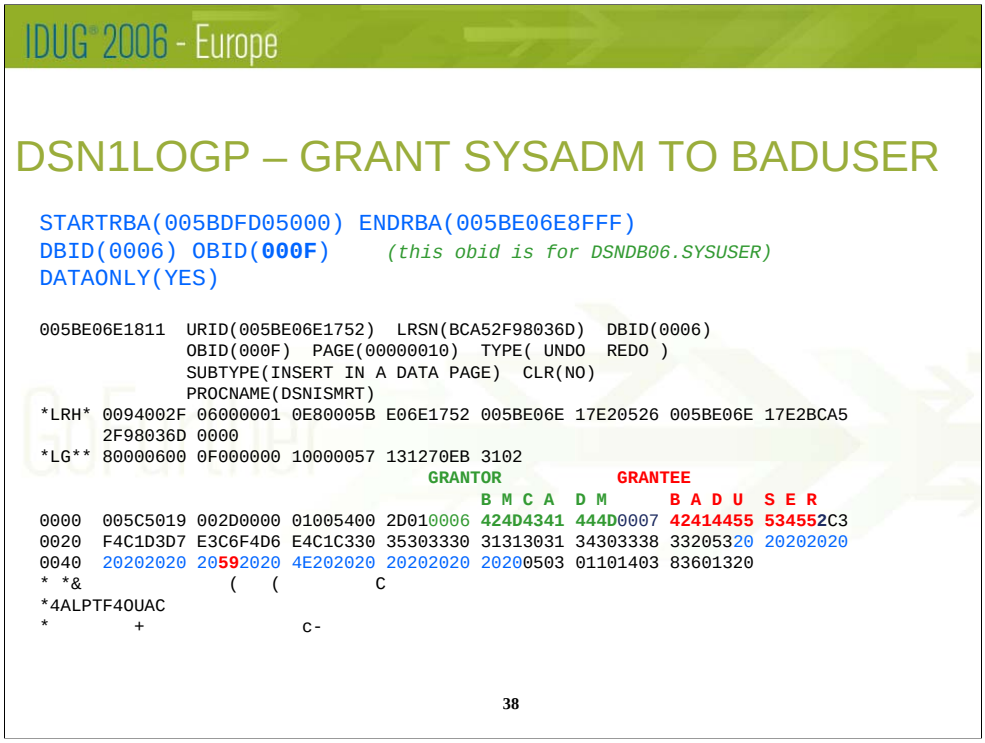
37

There are two major pitfalls/complications in trying to audit AUTH table changes via the log.

1) Some tables have rows where if the same GRANTOR grants or revokes a privilege from a GRANTEE with existing privileges or leaving some privileges, the associated column will be updated. DB2 can produce a partially logged update which will not contain the GRANTOR or GRANTEE columns in the log. You could possibly use DSN1PRNT to print the page and row to determine these values. SYSIBM.SYSUSERAUTH is one table which fits this paradigm.

2) When a BIND or FREE is done, DB2 will record rows in some authorization indicating the PLAN or PACKAGE has access to the resource. In these rows, the GRANTEETYPE is equal to 'P'. This is really just noise in the log to be filtered out of Authorization auditing. Most of the AUTH tables have a GRANTEETYPE column. Tables with GRANTEETYPE = 'P':

1) SYSIBM.SYSCOLAUTH
2) SYSIBM.SYSDBAUTH
3) SYSIBM.SYSPACKAUTH
4) SYSIBM.SYSPLANAUTH
5) SYSIBM.SYSRESAUTH
6) SYSIBM.SYSROUTINEAUTH
7) SYSIBM.SYSSEQUENCEAUTH
8) SYSIBM.SYSTABAUTH
9) SYSIBM.SYSUSERAUTH



This was a GRANT SYSADM TO BADUSER logged in SYSIBM.SYSUSERAUTH. I could get by with only OBID(000F) of the DSNDB06.SYSUSER space because it has only the one table.

This was also from a Version 8 system to show you that grotting data out of UNICODE catalog log can be bothersome in an EBCDIC world… The log processing tools on the market allow for the ability to do something like this:

```
WHERE
   TABLE NAME LIKE SYSIBM."SYS%AUTH"
   AND (SYSIBM.SYSCOLAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSDBAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSPACKAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSPLANAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSRESAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSROUTINEAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSSEQUENCEAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSTABAUTH.GRANTEETYPE <>  'P'
    OR  SYSIBM.SYSUSERAUTH.GRANTEETYPE <>  'P')
```

DSN1LOGP – Who FREEd that PLAN?

Prep

- Determine DBID/PSID/OBID of SYSIBM.SYSPLAN and/or SYSIBM.SYSPACKAGE
  - SYSPLAN – 0006 / 000A / 0022
  - SYSPACKAGE – 0006 / 0079 / 0080
- Determine PLAN and/or PACKAGE names of interest

Method

- Pretty much the same that we've done on the previous examples.

The OFFSET(003C) example was looking for the SYSIBM.SYSPLAN Table OBID, producing output for all SYSPLAN table activity on a fairly busy development system.

For the OFFSET(0052) example, I was searching for my specific PLAN NAME – ALPBKMM. The offset x'0052' was derived by

LRH - x'26' (Log Record Header – DSNDQJ00)

*LG** - x'12' (Data Manager Log Record Subheader – DSNDQJ00)

DMH – x'08' (Segment Header – LGBENTRY in DSNDQJ00)

Row Header – x'06'

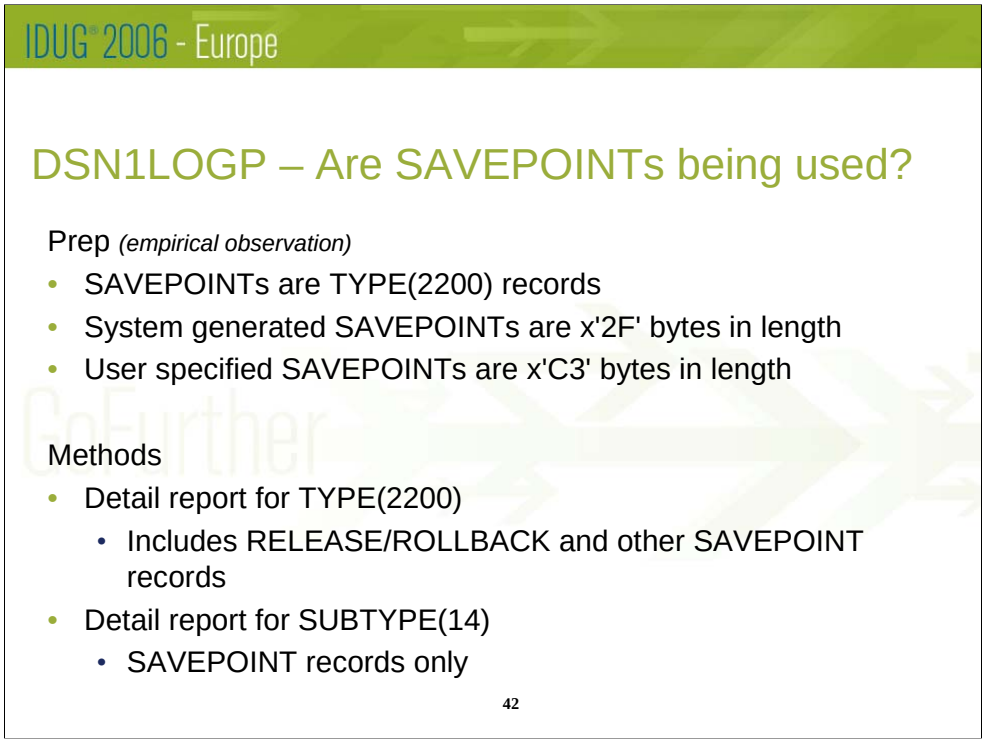
LINK Fullwords – x'0C' (documented in the Diagnosis Guide)

NAME is the first column in the SYSPLAN row -

DSN1LOGP – FREE PLAN detail

```
024B089EAC87  URID(024B086DE941)  LRSN(BCA557913E31)  DBID(0006)
              OBID(000A)  PAGE(00000C1C)  TYPE( UNDO  REDO )
              SUBTYPE(DELETE IN A DATA PAGE)  CLR(NO)
              PROCNAME(DSNIDILS)
*LRH* 00E00052 06000001 0E80024B 086DE941 024B089E AC350426 024B089E AC35BCA5
      57913E31 0000
*LG** 00000600 0A00000C 1C00024B 088955EC 3900
0000  00A82001 00220000 0000A000 2201000C 1C02000C 1E0E0029 ED15C1D3 D7C2D2D4
0020  D440C1D3 D7C4C5E5 4040F0F5 F0F3F0F1 C2E2E8E8 F0F9F2F7 F5F8F5F6 80007CB8
0040  D2800018 82E4C3D5 D5D5D5C3 D9C4C1D2 D4D44040 C1D3D7C4 C5E54040 84008000
0060  D5404040 40404040 40404040 40404040 408000F1 4040C4C5 40404040 40404040
0080  40200503 01092758 561582D5 D50000C4 20050301 09275840 13684040 40404040
00A0  40408000 0025D5D2
* y                              ALPBKM
*M ALPDEV  050301BSYY09275856  @
*K   bUCNNNNCRDAKMM  ALPDEV  d
*N                 1  DE
*          bNN  D
*      NK
```


Again, in DB2 Version 8, the format of the SYSPLAN row changes for long name and Unicode support. This data will no longer be in EBCDIC.



This was actually generated from a discussion with a person as to whether or not SAVEPOINTs were being executed on their system. Our product behaved in a manner that was consistent with a user specified SAVEPOINT. However, the contact was fairly certain that they were not being used. By looking for TYPE(2200) records, we were able to identify SAVEPOINT names and by looking at the associated URID, determine under which PLAN and AUTHID they were being executed.

I tried adding VALUE/OFFSET to the SUBTYPE specification for a x'00C3' at offset zero (checking the length). But, it did not suppress the shorter system generated SAVEPOINTs.

DSN1LOGP – SAVEPOINT example

```
003C0074BA6A  MEMBER(DGA3    )  URID(003C0074B9DA)
              LRSN(BCA53A0E96FA)  TYPE( UNDO )
              SUBTYPE(SAVEPOINT)
*LRH* 002F0090 22000014 0E80003C 0074B9DA 003C0074 B9DA0426 003C0074 B9DABCA5
      3A0E96FA 0003
0000  00E7D9E4 C9000059 87
--------------------------------------------------------------------------------
0183BFE68C79  MEMBER(DGA1    )  URID(0183BFE68293)
              LRSN(BCA56AA71024)  TYPE( UNDO )
              SUBTYPE(SAVEPOINT)
*LRH* 00C304AB 22000014 0E800183 BFE68293 0183BFE6 87CE0426 0183BFE6 87CEBCA5
      6AA71024 0001
0000  A0D2D4D4 E2E5D7E3 F2404040 40404040 40404040 40404040 40404040 40404040
0020  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040
0040  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040
0060  40404040 40404040 40404040 40404040 40404040 40404040 40404040 40404040
0080  40BCA56A A710246D D0C4C7C1 40404040 40404040 40404040 40404000 00
* KMMSVPT2
*
*
*
*  v x  _ DGA
```


The first SAVEPOINT record is one which DB2 took on your behalf. The data in the log record appears to be a counter which increments.

The second one was one an explicitly named SAVEPOINT of KMMSVPT2. This was on a Version 7 DB2 subsystem.

DSN1LOGP – Find a Quiet Point

Prep

- Analyze REPORT RECOVERY first
  - SYSLGRNGX entries can encompass many Units of Recovery
- Determine the DBID/PSIDs of the spaces you care about
- Have a spread sheet available or lots of paper and sharpen your pencil
  - Any break between overlapping Units of Recovery are physical Quiet Points to which you could use a a recovery point.

Method

- My favorite brute force utility… IEHIBALL

DSN1LOGP – Summary output again

```
DSN1151I DSN1LPRT  UR  CONNID=CIP4W    CORRID=ENTRRED10139  AUTHID=CICSPPAE  PLAN=RE002PP
  START DATE=05.042 TIME=18:46:06  DISP=ABORTED     INFO=PARTIAL
  STARTRBA=BA4EE9C718A7 ENDRBA=BA50263D9204 STARTLRSN=BC8EA2839DD4 ENDLRSN=BC90E81F98C7
    NID=*
  LUWID=*  COORDINATOR=*
  PARTICIPANTS=*
   DATA MODIFIED:
        DATABASE=0111=PVRESA    PAGE SET=00AE=SVLGDOS
        DATABASE=0111=PVRESA    PAGE SET=00B1=CVLGDOS0
        DATABASE=0111=PVRESA    PAGE SET=0023=YVLGDOS2
        DATABASE=0111=PVRESA    PAGE SET=00F2=YVLGDOS3
        DATABASE=0111=PVRESA    PAGE SET=0073=SVLGVRD
        DATABASE=0111=PVRESA    PAGE SET=0076=CVLGVRD0
        DATABASE=0111=PVRESA    PAGE SET=0064=SVERDOS
        DATABASE=0111=PVRESA    PAGE SET=0108=YVDOSS22
        DATABASE=0111=PVRESA    PAGE SET=0042=YVDOSS23
        DATABASE=0111=PVRESA    PAGE SET=007F=YVDOSS24
        DATABASE=0111=PVRESA    PAGE SET=0101=SVDOSS2
```


IEHIBALL… Look for the DATABASE and TABLESPACES (PAGE SET) for which you have interest. For non-data sharing, extract the STARTRBA and ENDRBA. For data sharing, use STARTLRSN and ENDLRSN.

SORT based upon START point. Collapse overlapping entries into a single entry. Any gaps left over are quiet points.

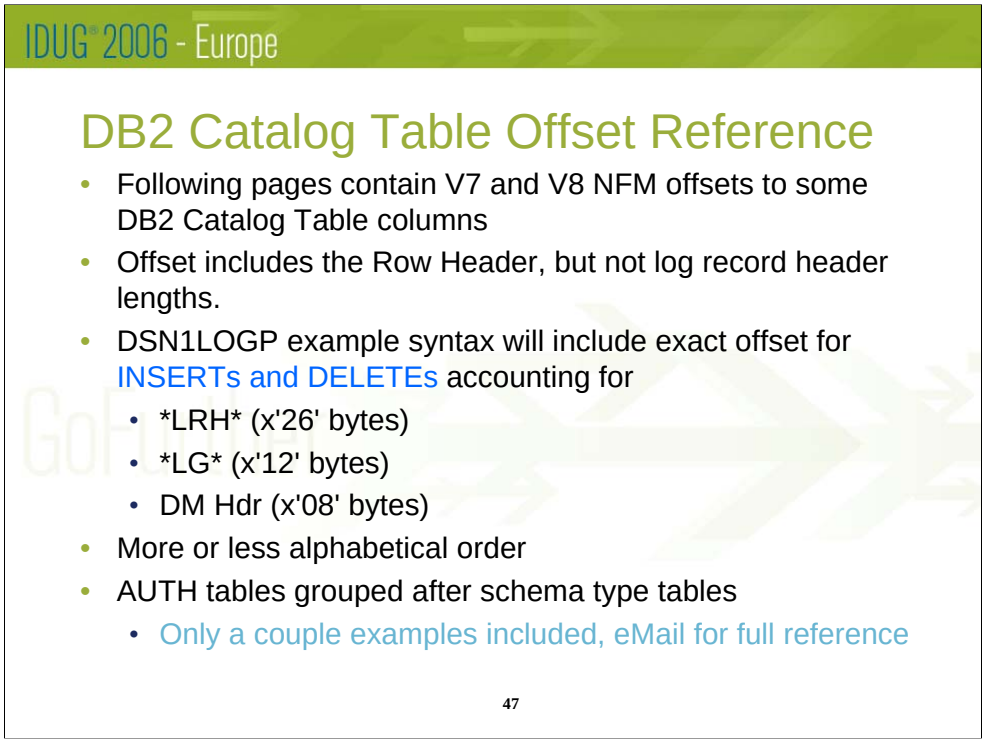
The next thing to consider… Physical quiet points are not necessarily Logical quiet points. You can have processes which do intermittent commits. So, you may want to consider PLAN names, correlation IDs, et cetera when a gap is found.

Other Usage ideas

- SYSCOPY analysis
  - Regular Utility monitoring
  - IMAGE COPY discovery after a DROP TABLESPACE
- REVOKE SYSADM cascading revokes
  - Find the SYSUSERAUTH DELETE
  - Report on the URID to see other activity
- Reverse engineer DDL after a DROP to recreate objects
- Ideas from you???

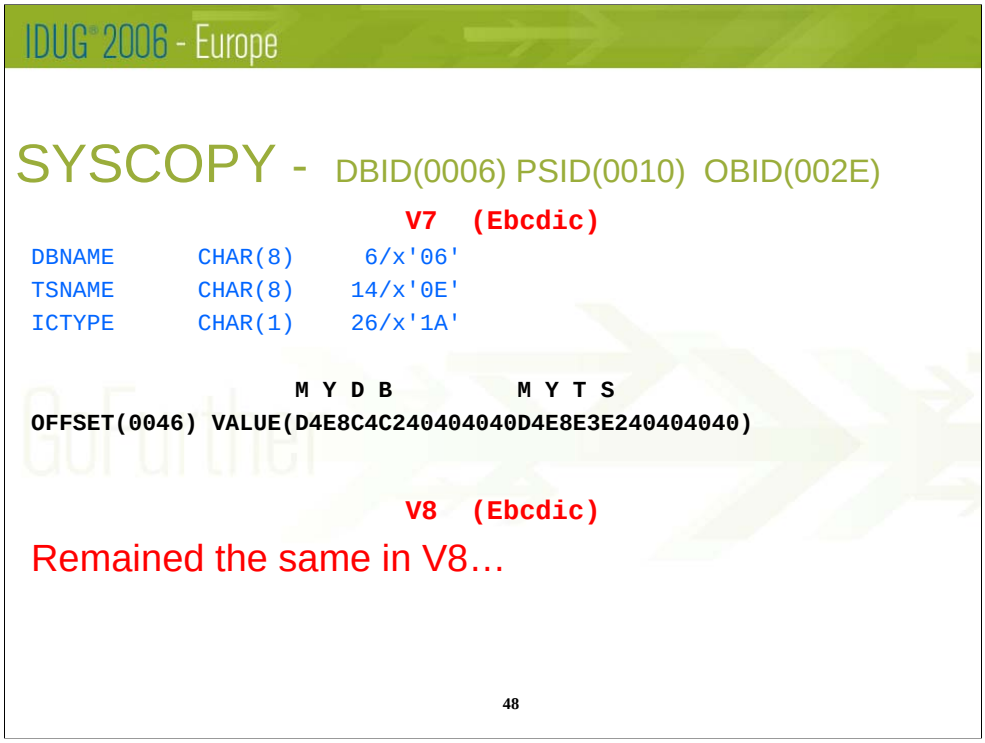
46

The log tools or DB2 administration products on the market which produce DDL basically just take the DB2 Catalog rows and generate the DDL representing the data either in the log or the Catalog tablespaces. Both the Diagnosis Reference manual and SQL Reference manual contain descriptions of the DB2 Catalog tables and their column contents and values which can be used to recreate the objects.



Offsets taken from the DB2 Table "Catalog Formats" chapter of the Diagnosis Guide and Reference… accounts for both the row header as well as any hash/link fields which exist in the older DB2 Catalog tables.

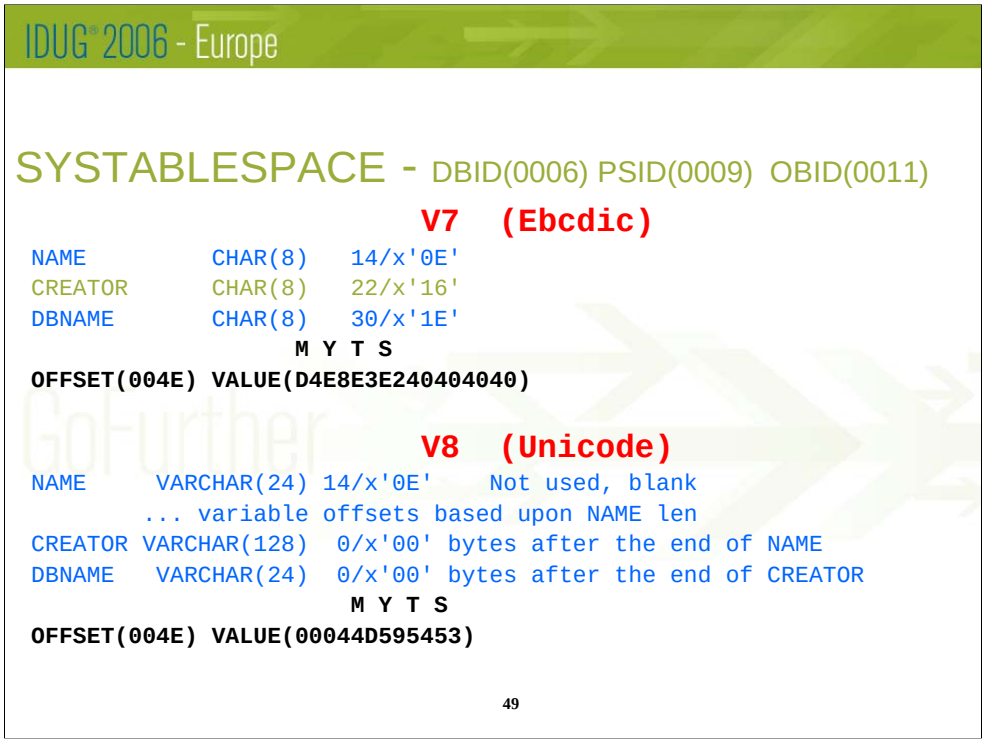
I'd like to stress again, the finding the offset for UPDATEs is different either due to being either partially logged beginning at the first changed byte or if fully logged either due to page movement or DATA CAPTURE CHANGES, there is an additional UPDATE appendage and other considerations.



SYSIBM.SYSCOPY is in DSNDB06.SYSCOPY…

The offset(0046) was computed by

```
LRH len       =  38 / x'26'
LG len        =  18 / x'12'
DM Hdr        =   8 / x'08'
DBNAME offset =  06 / x'06'
               ------------
                 70 / x'46'
```



SYSIBM.SYSTABLESPACE is in DSNDB06.SYSDBASE…

The offset(004E) for the V7 NAME was computed by

```
LRH len       =  38 / x'26'
LG len        =  18 / x'12'
DM Hdr        =   8 / x'08'
NAME   offset =  14 / x'0E'
               ------------
                 78 / x'4E'
```

The creator name being between the (tablespace)NAME and DBNAME adds a difficulty to compare for both values in a single offset clause unless you know the creator and include it as part of the VALUE clause.

SYSUSERAUTH - DBID(0006) PSID(000F) OBID(002D)

V7 (Ebcdic)

```
GRANTOR      CHAR(8)    6/x'06'
GRANTEE      CHAR(8)   14/X'0E'
```

V8 (Unicode)

```
GRANTOR  VARCHAR(128)   6/x'06'
      ... variable offsets based upon GRANTOR len
GRANTEE  VARCHAR(128)   0/x'00' bytes after the end of GRANTOR
```

This is the table which contains SYSADM, BINDADD, etc. type GRANTs represented as a bunch of single Y/N columns indicating what authorities are involved.

SYSIBM.SYSUSERAUTH is in DSNDB06.SYSUSER…

Ask 'em if you have 'em.

Session: B15

DSN1LOGP – It could save your job one day

Ken McDonald

BMC Software

Ken_McDonald@bmc.com