# From IMS Legacy …
# … to Infinity & Beyond

Dougie Lawson

dl1ims@gmail.com

With thanks to Winsopia

# Agenda

- Where we are today
- Where we are heading tomorrow
- IMS Catalog – New with IMS V12 (in 2012)
  - HALDB, OSAM, Model Utility and a collection of Jar files
  - Catalog populate utility
  - ACBGEN changes
  - IMS PROCLIB changes for IMS
- IMS ODBM – Open Database Manager
  - IMS PROCLIB definitions for ODBM
- IMS Connect changes
  - IMS PROCLIB changes for ODBM with CONNECT

# Agenda (continued)

- Java programming
  - JDBC
  - Java DL/I
- Java SQL for IMS
  - Things that differ from Java for Db2
- COBOL with SQLIMS
- CICS with ODBM
- Batch programs with ODBM
- Db2 Stored Procedures with ODBM
- Managed ACBs

# Where we are today

- I’m assuming the system is built like this:
  - IMS V15.1 or V15.2
  - IMS DB/DC
  - HDAM
  - HIDAM
  - Maybe the odd DEDB
  - IMS Connect
  - Some HALDB PHDAM, PHIDAM, PSINDEX
  - No catalog
  - No ODBM
  - Db2 with some JDBC/SQLJ coming in through DDF
  - Db2 stored procedures running IMS Connect
  - CICS DBCTL with CICS as a DRA client

# Where we are heading tomorrow

- The system will look like this:
  - IMS V15.2
  - IMS DB/DC
  - HDAM
  - HIDAM
  - Maybe the odd DEDB
  - IMS Connect
  - Some HALDB PHDAM, PHIDAM, PSINDEX
  - IMS Catalog
  - IMS ODBM
  - Db2 with some JDBC/SQLJ coming in through DDF
  - Db2 Stored procedures running DL/I
  - Db2 Stored procedures running IMS Connect
  - CICS using ODBM
  - z/OS Connect

# IMS Catalog

- New with IMS V12
- There's a good RedPiece http://www.redbooks.ibm.com/redpapers/pdfs/redp4812.pdf
- Holds IMS meta data in an OSAM HALDB
  - DBRC optional but highly recommended
  - Backups and recovery just like every other OSAM HALDB
- Required for
  - Java
  - SQLIMS
  - Managed ACBs
- Needs radical changes to your source code management system like Endevor as ACBGEN changes
- Needs PROCLIB updates in IMS

# Why we need a Catalog

- Java was there in IMS V7.1 (2001)
- System Jar files supplied with IMS base code (they still are)
- DL/I databases needed to be mapped to a Java object
- Read the DBD, create a metadata Jar file
- Distribute Jar file to client system running Java program
- The IMS DBA changes the database
  - ACBGEN, promote to production with Online Change
  - Rebuild Jar file
  - Re-distribute the current version of the Jar to ALL clients
  - Hope that the Jar file matches the ACB running in IMS – there's no checking
- ftp://www.redbooks.ibm.com/redbooks/SG247856/SG247856.zip still has the materials from the old DL/I Model utility RedBook SG24-7856

# The Catalog solves the Jar file problem

- System Jar files can now query IMS for metadata
- IMS DL/I GUR call gets Catalog record in XML
- XML processed by System Jar files to dynamically create the Java object for the database
- If the database changes the Catalog gets updated
- The client gets the database structure as it is right now
- No more DL/I Model Utility
- No need to distribute metadata Jar files and keep them current with the ACB used by IMS
- We can run our own GUR calls
  - In COBOL
  - With the DL/I Test Program DFSDDLT0
  - XML is ASCII data
  - There are published DTDs in the IMS Sample library for the XML
    - DFS3XDBD for DBD records in IMS15.SDFSSMPL(DFS3XDBD)
    - DFS3XPSB for PSB records in IMS15.SDFSSMPL(DFS3XPSB)

# IMS PROCLIB updates

- IMS.PROCLIB(DFSPBxxx)

  RRS=Y,

# IMS PROCLIB updates

- IMS.PROCLIB(DFSVSMxx)

```
IOBF=(24576,50)
IOBF=(8192,50)
IOBF=(4096,50)
SBONLINE,MAXSB=100
```

# IMS PROCLIB updates

- IMS.PROCLIB(DFSCGxxx)

```
RMENV=Y,
```

# Resource Manager PROCLIB updates

- IMS.PROCLIB(IMSRM)

```
//IEFPROC  EXEC PGM=BPEINI00,REGION=3000K,
//  PARM=('BPECFG=BPECONFG','BPEINIT=CSLRINI0','RMINIT=002')
//*       'ARMRST=N','RMNAME=RM2')
//*
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*
```

# Resource Manager PROCLIB updates

- IMS.PROCLIB(CSLRI002)

```
ARMRST=N,
RMNAME=RM2,
IMSPLEX(NAME=PLEX2)
<SECTION=REPOSITORY>
REPOSITORY=(NAME=IMSREPO,TYPE=IMSRSC,GROUP=R002REPO,
AUDITACCESS=NOAUDIT)
```

# Repo PROCLIB updates

- IMS.PROCLIB(IMSREPO)

```
//IT2AREPO PROC RGN=512M,BPERCFG=BPERSCFN,
//             FPQRCFG=IMSRSCF,SOUT='*'
//*
//RESPPROC  EXEC PGM=BPEINI00,REGION=&RGN,
//  PARM='BPEINIT=FRPINI00,BPECFG=&BPERCFG,FRPCFG=&FPQRCFG'
//*
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//*
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//*
//FRPPRINT DD  SYSOUT=&SOUT
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
```

# Repo PROCLIB updates

- IMS.PROCLIB(IMSRSCF)

```
XCF_THREADS=8
MAX_COMMUNICATION_RETRY=32
MBR_CORE_MAX=1024
IMSPLEX(NAME=PLEX2)
RSNAME=IMSRS
PRIMARY_CATALOG_REPOSITORY_INDEX=(
IMS.REPO.CATPRI.RID)
PRIMARY_CATALOG_REPOSITORY_MEMBER=(
IMS.REPO.CATPRI.RMD)
SECONDARY_CATALOG_REPOSITORY_INDEX=(
IMS.REPO.CATSEC.RID)
SECONDARY_CATALOG_REPOSITORY_MEMBER=(
IMS.REPO.CATSEC.RMD)
VSAM_BUFNO=128
VSAM_BUFSIZE=8
XCF_GROUP_NAME=R002REPO
AUDIT=NO
AUDIT_FAIL=CONTINUE
AUDIT_LEVEL=HIGH
AUDIT_DEFAULT=NOAUDIT
```

# Catalog and Repo PROCLIB updates

- IMS.PROCLIB(DFSDFxxx)

```
*-----------------------------------------------------------------*
* IMS CATALOG SECTION                                             *
*-----------------------------------------------------------------*
<SECTION=CATALOG>
  CATALOG=Y
  ALIAS=DFSC
/*****************************************************************/
/* DYNAMIC RESOURCE DEFINITION SECTION                           */
/*****************************************************************/
 <SECTION=DYNAMIC_RESOURCES>
AUTOIMPORT=AUTO
AUTOEXPORT=AUTO
IMPORTERR=ABORT
REPOERR=ABORT
/*****************************************************************/
/* REPOSITORY SECTION                                            */
/*****************************************************************/
<SECTION=REPOSITORY>
REPOSITORY=(TYPE=IMSRSC)
```

# Catalog Populate Utility

- Runs like ACBGEN and builds staging ACBLIB as always
- Can run as DLIBATCH
  - Needs DBRC=Y, data sharing & IRLM if IMS online is using the Catalog
  - Needs a special DFSDFxxx member in IMS.PROCLIB
  - Ideal for folks who love counting commas
  - `DLI,DFS3PU00,DFSCP001,,,,,,,,,,Y,Y,JRLM,,,,,,,,,,,DFSDF=CAT`
- Can run as BMP
  - You need to DBR and START the Catalog HALDB as ACCESS=UP for that
  - Sample BMP JCL on next page

# Catalog Populate Utility BMP JCL

```
// JCLLIB ORDER=IMS.SDFSPROC
//ACB3BMP EXEC PGM=DFS3UACB,REGION=0M
//STEPLIB  DD  DSN=IMS.SDFSRESL,DISP=SHR
//PROCLIB  DD  DSN=IMS.PROCLIB,DISP=SHR
//DFSRESLB DD  DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSABEND DD  SYSOUT=*
//IMS      DD  DSN=IMS.PSBLIB,DISP=SHR
//         DD  DSN=IMS.DBDLIB,DISP=SHR
//IMSACB   DD  DSN=IMS.ACBLIB,DISP=OLD
//SYSUT3   DD  UNIT=SYSDA,SPACE=(80,(100,100))
//SYSUT4   DD  UNIT=SYSDA,
//         SPACE=(256,(100,100)),
//         DCB=KEYLEN=30
//SYSIN    DD  *
         BUILD PSB=DFSIVP33
/*
```

```
//IMSACB01 DD  DSN=*.IMSACB,DISP=OLD
//ACBCATWK DD  SPACE=(CYL,(1,1)),UNIT=SYSDA
//SYSINP   DD  DUMMY
//DFS3PPRM DD *
BMP,DFS3PU00,DFSCP001,,,N00000,,,,,,,,IMS,,,,'',,,
/*
//
```

# ACBGEN changes

- Most sites will have ACBGEN built into their source manager
  - SCLM
  - Endevor
  - Zigi (aka Easy Github for z/OS)
- The ACBGEN may run on a development system
- The Catalog runs on the production system
- The process will need to change
  - You will need to work with your SCLM/Endevor folks
- We’re not running DFS3PU00 every day
- Regular ACBGEN needs to use DFS3UACB on the production system
- Managed ACBs enables the use of Catalog export and import
  - DFS3CCE0 – Catalog export
  - DFS3CCI0 – Catalog import
  - DFS3LU00 – Library builder

# IMS Open Database Manager (ODBM)

- Gives an open interface to IMS databases
- Can be used by Db2 Stored Procedures
- Will be used by IMS Connect for remote database connectors
  - Java DL/I
  - JDBC
  - DRDA native clients

# ODBM PROCLIB updates

- IMS.PROCLIB(IMSODBM)

```
//IMSODBM  PROC RGN=64M,TME=NOLIMIT,SOUT=*,
//              BPECFG=BPECONFG,
//              ODBMINIT=ODM,
//              CSLDI=CSLDINI0,
//              PARM1='RRS=Y,ARMRST=N'
//*
//ODBM     EXEC PGM=BPEINI00,REGION=&RGN,TIME=&TME,
//  PARM='BPECFG=&BPECFG,BPEINIT=&CSLDI,ODBMINIT=&ODBMINIT,&PARM1'
//STEPLIB  DD   DSN=IMS.SDFSRESL,DISP=SHR
//         DD   DSN=SYS1.CSSLIB,UNIT=SYSALLDA,DISP=SHR
//PROCLIB  DD   DSN=IMS.PROCLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
```

# ODBM PROCLIB updates

- IMS.PROCLIB(CSLDIODM)

```
<SECTION=GLOBAL_DATASTORE_CONFIGURATION>
IDRETRY=5                  /* RETRY CONNECTION 5 TIMES BEFORE QUIT    */
MAXTHRDS=10                /* 10 THREADS MAX TO ANY IMS DATASTORE     */
TIMER=30                   /* 30 SECONDS BETWEEN ID RETRY ATTEMPTS    */
FPBUF=3                    /* 10 DEDB BUFFERS PER THREAD              */
FPBOF=3                    /* 10 OVERFLOW BUFFERS PER THREAD          */
CNBA=90                    /* (FPBUF+FPBOF)*MAXTHRDS <= CNBA          */
DIAGDPSBMSG=LONG           /* NO ABEND210 LOTS OF MESSAGES            */
<SECTION=LOCAL_DATASTORE_CONFIGURATION>
ODBM(NAME=ODBM
    DATASTORE(
      NAME=IMS
    )
)
```

# ODBM PZP module

```
PZP      TITLE 'DATABASE RESOURCE ADAPTER STARTUP PARAMETER TABLE'
DFSPZP00 CSECT
         EJECT
         DFSPRP DSECT=NO,                                              X
               FUNCLV=3,           X        CCTL FUNCTION LEVEL        X
               DDNAME=CCTLDD,      XXXXXXXX DDN FOR CCTL RESLIB DYNALOC X
               DSNAME=IMS.SDFSRESL,                                    X
               DBCTLID=IT2A        NAME OF DBCTL REGION               X
               USERID=IMSUSER,     XXXXXXXX NAME OF USER REGION       X
               MINTHRD=001,        XXX      MINIMUM THREADS           X
               MAXTHRD=005,        XXX      MAXIMUM THREADS           X
               TIMER=60,           XX       IDENTIFY TIMER VALUE - SECS X
               FPBUF=0001,         XXXX     FP FIXED BFRS PER THREAD  X
               FPBOF=0001,         XXXX     FP OVFLW BFRS PER THREAD  X
               SOD=A,              X        SNAP DUMP CLASS           X
               TIMEOUT=060,        XXX      DRATERM TIMEOUT IN SECONDS  X
               CNBA=002,           XXX      TOTAL FP NBA BFRS FOR CCTL  X
               IMSPLEX=PLEX2,      xxxxx    IMSPLEX  FOR ODBM          X
               ODBMNAME=ODIT2A     xxxxxx   ODBM NAME TO USE
         END
```

# IMS Connect – changes for ODBM

- Connect gives us
  - Transaction processing
  - TCP/IP MSC links
  - TCP/IP CICS ISC links
  - Callout processing
  - Command processing

- We’re going to add
  - ODBM for database processing
  - Java DL/I support
  - DRDA support
  - JDBC support

# IMS Connect PROCLIB updates

- IMS.PROCLIB(HWSCFG00)

```
HWS(    ID=HWS2A,
        RRS=Y,
        SMEMBER=XIMS
)

ODACCESS(
        DRDAPORT= ( ID=4394 )
        IMSPLEX=( MEMBER=IMSHWSD TMEMBER=PLEX2)
        ODBMAUTOCONN=Y
)
```

# DBD Changes

- DBD needs external names
- They're the names for the "table" and "columns"
- IBM supplied DFSIVD2 (aka IVPDB2) does have the new parameters

```
SEGM     NAME=A1111111,                                    C
            EXTERNALNAME=PHONEBOOK,                        C
            PARENT=0,                                      C
            BYTES=40,                                      C
            RULES=(LLL,LAST)

FIELD    EXTERNALNAME=FIRSTNAME,NAME=FNAME,                C
            BYTES=10,                                      C
            START=11,                                      C
            TYPE=C,                                        C
            DATATYPE=CHAR
```

# PSB Changes

- Needs to have a PCBNAME
  - That's used for the SQL statement FROM clause
  - SELECT LASTNAME, FIRSTNAME FROM PCB01.PHONEBOOK
  - IBM supplied sample DFSIVP2 doesn't have a PCBNAME
  - Assembler label field is mapped to PCBNAME=xxxx if PCBNAME is not coded

```
            PCB   TYPE=TP,MODIFY=YES
PCB01       PCB   TYPE=DB,DBDNAME=IVPDB2,PROCOPT=A,KEYLEN=10
            SENSEG NAME=A1111111,PARENT=0,PROCOPT=A
            PSBGEN LANG=ASSEM,PSBNAME=DFSIVP2
            END
```

# IMS Java Programming

- JDBC
  - Ideal for programmers who know SQL but don't know IMS
- Java DL/I
  - More familiar territory for COBOL/IMS programmers

- https://www.ibm.com/support/knowledgecenter/SSEPH2_15.1.0/com.ibm.ims15.doc.apr/ims_javaprogrammingreference.htm has a full scale Javadoc for everything in imsudb.jar

- The IMS call compare trace
  `/TRA SET ON PSB sample1 COMP.`
  will capture DL/I database calls from your Java programs.

# IMS Java Programming (JDBC)

- JDBC
  - Easy and familiar
  - Uses SQL just like Db2 JDBC
  - Some differences in the SQL language – the IMS books list the allowed SQL language elements
    - https://www.ibm.com/support/knowledgecenter/SSEPH2_15.2.0/com.ibm.ims152.doc.apg/ims_sqlkeywordsjdbc.htm
  - IMS turns the SQL into DL/I calls
  - Hierarchy looks like SQL tables with foreign keys
  - There is support for a type-2 (local) or type-4 (remote) connection
    - With IMS Connect it's always going to be a type-4 connection

# Create an IMS Connection

```
static String hostname = "192.168.xxx.xxx";
static String userid = “userid";
static String password = “password”;
static String psbLabel = "DFSIVP33";
static int portNum = 4394;

private static PSB createAnImsJDBCConnection() throws Exception {
        IMSDataSource ds = new IMSDataSource();
        ds.setHost(hostname);
        ds.setPortNumber(portNum);
        ds.setDriverType(IMSConnectionSpec.DRIVER_TYPE_4);
        ds.setUser(userid);
        ds.setPassword(password);
        ds.setDatabaseName(psbLabel);
        connection = ds.getConnection();
        return connection;
        }
```

# IMS Java Programming (JDBC)

```
private static void executeAndDisplaySqlQuery() throws Exception {
        Connection connection = createAnImsJDBCConnection();
        String sql = "SELECT LASTNAME, FIRSTNAME, EXTENSION,
                        ZIPCODE FROM TELEPCB.PHONEBOOK";
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(sql);
        ResultSetMetaData rsmd = rs.getMetaData();
        int colCount = rsmd.getColumnCount();
        System.out.printf("%-16s| %-20s\n", "Column Name", "Column Value");
        while (rs.next()) {
                for (int i = 1; i <= colCount; i++) {
                        System.out.printf("%-16s| %-20s\n", rsmd.getColumnName(i), rs.getString(i));
                        }
                }
                connection.commit();
                connection.close();
        }
```

# IMS Java Programming

- Java DL/I
  - Needs an understanding of the IMS hierarchy
  - Needs an understanding of IMS segment search arguments (SSAs)
  - May need an understanding of command codes
  - Can issue some IMS system services call
  - Can use an AIB or PSB
    - Will usually use AIB
  - https://www.ibm.com/docs/api/v1/content/SSEPH2_15.1.0/com.ibm.ims15.javadoc.doc/topics/overview-summary.html

# Create an IMS Connection

```
static String hostname = "192.168.xxx.xxx";
static String userid = “userid";
static String password = “password”;
static String psbLabel = "DFSIVP33";
static int portNum = 4394;
private static PSB createAnImsPSBConnection() throws Exception {
        PSB psb = null;
        IMSConnectionSpec imsConnSpec = IMSConnectionSpecFactory.createIMSConnectionSpec();
        imsConnSpec.setDatastoreServer(hostname);
        imsConnSpec.setPortNumber(portNum);
        imsConnSpec.setDatabaseName(psbLabel);
        imsConnSpec.setUser(userid);
        imsConnSpec.setPassword(password);
        imsConnSpec.setDriverType(IMSConnectionSpec.DRIVER_TYPE_4);
        psb = PSBFactory.createPSB(imsConnSpec);
        return psb;
        }
```

# IMS Java Programming (Java DL/I)

```
private static void readASpecificRecordWithDliGu() throws Exception {
        PSB psb = createAnImsPSBConnection();
        PCB pcb = psb.getPCB("TELEPCB");
        SSAList ssaList = pcb.getSSAList("PHONEBOOK");
        ssaList.addInitialQualification("PHONEBOOK", "LASTNAME", SSAList.EQUALS, "BAGGINS");
        Path path = ssaList.getPathForRetrieveReplace();
        pcb.getUnique(path, ssaList, false); // DL/I GU with SSA

        System.out.printf("%-12s| %-12s\n", "Field name", "Field value");
        System.out.printf("%-12s| %-12s\n", "FIRSTNAME", path.getString("FIRSTNAME"));
        System.out.printf("%-12s| %-12s\n", "LASTNAME", path.getString("LASTNAME"));
        System.out.printf("%-12s| %-12s\n", "EXTENSION", path.getString("EXTENSION"));
        System.out.printf("%-12s| %-12s\n", "ZIPCODE", path.getString("ZIPCODE"));

        psb.commit();
        psb.close();
}
```

# IMS Java imsudb.jar and modern JDK/Java

- My Java programming was done in March on my old laptop
- My new laptop has a more modern version of Java
- I kept getting
  - Exception in thread "main" java.lang.NoClassDefFoundError: javax/xml/bind/JAXBContext
- IBM Software Support solution
  - Convert to a Maven project
  - Add some strange commands to pom.xml
  - https://github.com/DougieLawson/IMS_assembler/blob/master/pom.xml

# COBOL with SQLIMS

- Looks a lot like Db2 SQL

```
001000**************************************************
001100* SQL INCLUDE FOR SQLCA *
001200**************************************************
001300     EXEC SQLIMS
001400     INCLUDE SQLIMSCA
001500     END-EXEC.
001600**************************************************
001700* SQL INCLUDE FOR SQLDA *
001800**************************************************
001900     EXEC SQLIMS
002000     INCLUDE SQLIMSDA
002100     END-EXEC.
```

# COBOL with SQLIMS

- No static SQL everything is dynamic SQL
- Prepare, describe, execute

```
010000*
010100     EXEC SQLIMS
010200       DECLARE TELE1 CURSOR FOR DYSQL
010300     END-EXEC.
010400*
010500     MOVE "SELECT LASTNAME, FIRSTNAME FROM PCB01.PHONEBOOK"
010600                TO SQL-STMT-TEXT.
010700     MOVE 198 TO SQL-STMT-LENGTH.
010800*
010900     EXEC SQLIMS
011000       PREPARE DYSQL FROM :SQL-STATEMENT
011100     END-EXEC.
011200 *
011300     EXEC SQLIMS
011400       DESCRIBE OUTPUT DYSQL INTO :SQLIMSDA
011500     END-EXEC.
011600*
```

# COBOL with SQLIMS

- All calls run with an AIB
- Errors are passed back in SQLCODE and SQLSTATE
- Some new SQL codes just for IMS

```
015400*****************************************************************
015500* DISPLAY ERROR MESSAGE
015600*****************************************************************
015700 DB-SQL-ERROR.
015800
015900     DISPLAY "ERROR OCCURRED IN WHENEVER SQLIMSERROR".
016000     MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
016100     DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
016200     DISPLAY "SQLIMSERRMC= " SQLIMSERRMC.
016300     DISPLAY "SQLIMSSTATE= " SQLIMSSTATE.
016400     GO TO RETURN-TO-CALLER.
016500
```

- Watch out for packed decimal with SQLIMSCODE
  - SQLIMSCODE PIC S9(9) COMP-5.
  - SQLIMSCODE = 0500J  - J = x’D1’ – so that’s sqlcode “-5001”

# CICS with ODBM

- Nothing changes
- CICS continues to create a DRA connection
- The new DFSPZP parameters are ignored
- I’m not going to consider Java DL/I or JDBC from CICS

# Batch programs with ODBM

- All calls use AIB
- Interface is AERTDLI not AIBTDLI
- ABENDs in batch AERTDLI programs don’t cause ABENDU0113 in IMS

# Batch programs with ODBM

```
004000      MOVE +660 TO IOAREA-LENGTH
004100      MOVE "INIT" TO SUB-FUNCTION
004200      MOVE "IT2A" TO RESOURCE-NAME1
004300      CALL "AERTDLI" USING CIMS, AIB.
004400      IF CALL-RETURN-CODE NOT = 0 THEN
004500        DISPLAY  "INIT-RC:", CALL-RETURN-CODE,
004600          " RSN:", REASON-CODE,
004700          " RSNM1:", RESOURCE-NAME1,
004800          " RSNM2:", RESOURCE-NAME2
004900        CALL "CEE3ABD" USING CALL-RETURN-CODE, TIMING
005100      END-IF
```

# Batch programs with ODBM

```
005200     MOVE "DFSSAM02" TO RESOURCE-NAME1
005300     MOVE "IT2A" TO RESOURCE-NAME2
005400     CALL "AERTDLI" USING APSB, AIB.
005500     IF CALL-RETURN-CODE NOT = 0 THEN
005600       DISPLAY  "APSB-RC:", CALL-RETURN-CODE,
005700         " RSN:", REASON-CODE,
005800         " RSNM1:", RESOURCE-NAME1,
005900         " RSNM2:", RESOURCE-NAME2
006000       CALL "CEE3ABD" USING CALL-RETURN-CODE, TIMING
006200     END-IF
006300     MOVE "DBPCB01 " TO RESOURCE-NAME1
006400     CALL "AERTDLI" USING GU, AIB, DB-AREA, SSA.
```

# Db2 Stored Procedures with ODBM

- Connection to ODBM is with an CIMS INIT call
- PSB is allocated with an APSB call
- Use a DPSB when you want to change PSB
- Call CIMS to terminate a thread
  - TERM – terminates this DB thread
  - TALL – terminates ALL DB thread and destroys the DRA environment
- All calls use an AIB
- Interface is AERTDLI with AIB – not AIBTDLI just like ODBM batch
- An ABEND in a stored proc won’t cause ABENDU0113 in IMS

# Managed ACBs


## Toy Story 2

The Toys are back!

In a possible future presentation at this theatre

# My samples

- I've put all of the sample COBOL programs for this presentation on Github (as ASCII – so you can browse them on Github)

https://github.com/DougieLawson/IMS_assembler

- Not just assembler, now

# Questions and Feedback

Any Questions

Or send me an email or post questions on IMS-L

dl1ims@gmail.com