In this issue

3  Introduction to multi-dimensional clustering in UDB V8
8  No black boxes!
14  Calling the DSNWZP stored procedure from a REXX client program to display DSNZPARM parameters
20  CAF interface with caller in amode 24 or 31 and more
50  DB2 news

© Xephon plc 2003
Subscriptions and back-issues
A year’s subscription to DB2 Update, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

DB2 Update on-line
Code from DB2 Update, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions
When Xephon is given copyright, articles published in DB2 Update are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our Notes for Contributors from www.xephon.com/nfc.
Introduction to multi-dimensional clustering in UDB V8

This article discusses the concept of Multi-Dimensional Clustering (MDC), which was introduced in UDB DB2 V8. It is presented in a series of questions and answers, which will, hopefully, leave you better informed about MDCs – when to use them (and, perhaps more importantly, when not to use them) and how to set them up.

I ran all the SQL on a Windows 2000 laptop running DB2 8.1. I used the SALES table in the SAMPLE database as a reference table. The DDL for this table is (>db2look -d sample -e -t sales):

```sql
CREATE TABLE "DB2ADMIN"."SALES2" ("SALES_DATE" DATE,"SALES_PERSON" VARCHAR(15),"REGION" VARCHAR(15),"SALES" INTEGER ) IN "USERSPACE1" ;
```

WHAT WAS THE SITUATION PRIOR TO V8?

Prior to V8 you could store data on disk in only one order. So in a table containing, say, account number, name, and postcode, you can choose whether to store the data in account number order or postcode order, but not both. This is important when it comes to retrieving data. You would certainly have an index on all three columns, but there would be only one ‘clustering’ index (the index which determines the physical order on disk). So if our clustering index was account number, if we wanted to retrieve data for a range of account numbers, the underlying data pages could be sequentially scanned from the index pointers. If on the other hand we wanted to retrieve data for a range of postcodes, then we would of course use the postcode index, but the underlying data pages would not be sequential (as the page order on disk was defined by the account number value, not the postcode value).

WHAT DO MDCS GIVE ME?

As stated above, prior to V8 you could store data on disk using
only a single clustering index. This is a physical limitation – data can be stored on disk in only one way! What MDC offers you is the ability to effectively see data as if it were stored using many clustering indexes. This does not mean that the data is stored more than once on disk! – what DB2 does is to use a storage method (described later) to store the data, so you see the data as being clustered on one or more indexes.

SO HOW DO MDCS WORK?

The Administration Guide gives a full description of MDCs, so I will limit myself here to the information you need to get them working, and introduce you to some of the terminology. When you create a table you specify which columns you want to make up your MDC index – it is usually more than one column. Each column is called a dimension. The intersection of these dimensions is called a cell. These cells will contain the values for the appropriate combination of the dimensions. So, if you take the SALES table and create an MDC based on the SALES_DATE and REGION columns, then the cell which is the intersection of SALES_DATE and REGION for particular values of SALES_DATE and REGION will contain pointers to the other data for those values. This will become a lot clearer when we look at an example later on!

HOW DO I CREATE MDCS?

You create MDCs when you create the table by adding an ORGANIZE BY line:

```
CREATE TABLE "DB2ADMIN"."SALES2" (
  "SALES_DATE" DATE , "SALES_PERSON" VARCHAR(15) ,
  "REGION" VARCHAR(15) , "SALES" INTEGER )
organize by(sales_date,region)
IN "USERSPACE1" ;
```

You can’t alter a table to have MDCs – you need to specify them when you create the table.

Let’s look at an example

Using the SALES table, what we will do is create two test tables:
SALES1 and SALES2. The SALES1 table will not contain any MDCs, but will have indexes on the SALES_DATE and REGION columns. The SALES2 table will not have any indexes defined as such, but will have a single MDC defined on the columns SALES_DATE+REGION.

DDL for SALES1:

```
CREATE TABLE "DB2ADMIN"."SALES1"  (
  "SALES_DATE" DATE ,"SALES_PERSON" VARCHAR(15) ,"REGION" VARCHAR(15)
  ,"SALES" INTEGER )
IN "USERSPACE1" ;
create index s1a on sales1 (sales_date);
create index s1b on sales1 (region);
```

DDL for SALES2:

```
CREATE TABLE "DB2ADMIN"."SALES2"  (
  "SALES_DATE" DATE ,"SALES_PERSON" VARCHAR(15) ,"REGION" VARCHAR(15)
  ,"SALES" INTEGER )
organize by(sales_date,region)
IN "USERSPACE1" ;
```

If we look at the indexes created for both tables using the query:

```
>db2 select substr(tabname,1,10), substr(indname,1,18),
substr(colnames,1,40), indextype from syscat.indexes where tabname =
'SALES<n>'
```

For table SALES1 we have our two indexes:

```
  1          2                  3                                INDEXTYPE
  ---------- ------------------ -------------------------------- ---------
 SALES1     S1A                +SALES_DATE                      REG
 SALES1     S1B                +REGION                          REG
```

And for table SALES2 we have:

```
  1          2                  3                                INDEXTYPE
  ---------- ------------------ -------------------------------- ---------
 SALES2     SQL021126181051540 +REGION+SALES_DATE               BLOK
 SALES2     SQL021126181051780 +REGION                          DIM
 SALES2     SQL021126181051870 +SALES_DATE                      DIM
```

We have three indexes – one block index and two dimension indexes.

So if we run a query such as SELECT SALES from SALES2 where REGION = ‘Quebec’, then the optimizer will use the
SQL021126181051780 index (you can see this by running the query using:

```
>db2expln -d sample -t -q "select SALES from SALES2 where REGION = 'Quebec'"
```

I have not found a way of assigning a name to a particular MDC index – DB2 generates the name automatically for you.

Getting back to the SALES1/SALES2 tables – I seeded both of these tables from the SALES table using the `>db2 insert into sales<n> select * from sales` command.

The query we want to test out is paraphrased from the example in the *Administration Guide (Performance)* to demonstrate the benefits of using MDCs:

```
>db2 select sum(sales) from sales where month(sales_date)=3 and region = 'Quebec'
```

As the SALES table contains only 41 rows (and hence the initial number of rows in SALES1/2 is 41), I ran a bat file to copy the SALES table into the SALES1/2 tables many times. Therefore, for each iteration I doubled the size of the SALES1/2 tables, and for SALES1 REORGed on the SALES_DATE index. I did not REORG the SALES2 table at any point, but after every iteration I ran runstats on each table. I then ran the above query against each table. What I looked for was the optimizer cost in timerons for the query. The results are shown in Figure 1.

<table>
<thead>
<tr>
<th>Rows in SALES1/2 Table</th>
<th>Optimizer cost of non MDC query (SALES1)</th>
<th>Optimizer cost of MDC query (SALES2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>82</td>
<td>50</td>
<td>105</td>
</tr>
<tr>
<td>5248</td>
<td>254</td>
<td>106</td>
</tr>
<tr>
<td>1,0496</td>
<td>455</td>
<td>107</td>
</tr>
<tr>
<td>2,0992</td>
<td>867</td>
<td>108</td>
</tr>
<tr>
<td>4,1984</td>
<td>1,668</td>
<td>129</td>
</tr>
<tr>
<td>83,968</td>
<td>3,268</td>
<td>174</td>
</tr>
<tr>
<td>167,936</td>
<td>6,477</td>
<td>261</td>
</tr>
</tbody>
</table>

*Figure 1: Optimizer costs*
What Figure 1 shows us is that, for smaller tables, there is no benefit in using MDCs. However, as the number of rows in the table increases, you can see the benefit in cost terms of using MDCs. These results are specific to the table, the data in the table, and the query run. The query I used lent itself to using the MDC that I specified. This means that before deciding on whether to use MDCs or not, you need to have some idea about the queries that will be run against the table and what type of data you have in your table.

HOW DO I DECIDE HOW TO USE MDCS?
I don’t think there is a set of rules which exactly defines whether you should use MDCs or not. One thing I have found is that you want any cells that you create to be populated by more than one value. For example, if you look at the EMPLOYEE table, then you wouldn’t want to create an MDC on the single column EMPNO because this has a cardinality of 1 (there is a unique value of EMPNO for each row in the table).

SHOULD I CONVERT ALL MY INDEXES TO MDCS?
I would say definitely not!! See the comments I made in the How do I decide when to use MDCs question. You need to make sure that your data lends itself to having MDCs.

WHAT ARE THE ADVANTAGES OF USING MDCS?
As you can see from the discussion so far, one of the major benefits of using MDCs is the reduction in SQL runtime costs. The Administration Manual also states that you do not have to REORG tables which use MDCs – which must be good news for availability.

WHAT ARE THE DISADVANTAGES OF USING MDCS?
You cannot just convert all your indexes to be MDCs – their implementation must be carefully planned and monitored.
FINAL THOUGHTS

MDCs are a very valuable tool when it comes to reducing SQL runtime costs. Their implementation should be carefully planned because inappropriate use could result in an increase in runtime costs! I hope I have shown how to decide when to use them and how to implement them. They are certainly a welcome feature in UDB DB2 and well worth trying out.

C Leonard
Freelance Consultant (UK)  © Xephon 2003

No black boxes!

Before I even begin here I had better define what I mean by a ‘black box’. If I plan to recommend that you prohibit them we had better both understand what it is we are talking about proscribing.

Simply put, a black box is a database access program that sits in between your application programs and DB2. It is designed so that all application programs call the black box for data instead of writing SQL statements that are embedded into a program. The general idea behind such a contraption is that it will simplify DB2 development because programmers will not need to know how to write SQL. Instead, the programmer just calls the black box program to request whatever data is required. SQL statements become calls – and every programmer knows how to code a call, right?

This approach is commonly referred to as a ‘black box’ approach because the data access interface shields the developers from the ‘complexities’ of SQL. The SQL is contained in that black box and programmers do not need to know how the SQL works – just how to call the black box for data. Black boxes usually are introduced into an organization when management gets the notion that it would be quicker and easier for programmers to request data from a central routine than to teach them all SQL.
But there are a number of reasons why this approach is not sound. Let’s examine them.

IGNORANCE (OF SQL) IS NOT A VIRTUE

The basic premise of implementing black box technology is that it is better for programmers to be ignorant of SQL. This means that your company will be creating DB2 applications using developers with little or no understanding of how SQL works. So what may seem like simple requests to a non-educated programmer may actually involve very complex and inefficient SQL ‘behind the scenes’ running in the black box. So innocuous requests for data can perform quite poorly.

When programmers are knowledgeable about SQL they can at least understand the complexity of their data requests and formulate them to perform better. For example, SQL programmers will understand when data must be joined and thereby can form their data requests in such a way as to join efficiently (and perhaps to minimize joining in certain circumstances). With no knowledge of SQL the programmer will have no knowledge of joining – and, more importantly, no true means at his or her disposal to optimize their data requests.

As much as 80% of all database performance problems can be traced back to inefficient application code. Basic SQL is simple to learn and easy to start using. But SQL tuning and optimization is an art that can take years to master.

Be sure to train your application development staff in the proper usage of SQL – and let them write the SQL requests in their programs. Develop and publish SQL guidelines in a readily accessible place (such as your corporate intranet or portal). These guidelines should outline the basics elements of style for DB2 SQL programming. For example, at a very high level, the following rules of thumb need to be understood by your development staff:

• Simpler may be better for rapid understanding, but complex SQL is usually more efficient – SQL joins outperform program
joins, SQL WHERE clauses outperform program filtering, and so on.

- Let SQL do the work, not the program – the more work that can be done by DB2 in its database engine the better your applications will perform.

- Retrieve the absolute minimum number of rows required, never more – it is better to eliminate rows in SQL WHERE clauses than it is to bring the data into the program and bypass it there. The less data that DB2 needs to read and send to your program the better your applications will perform.

- Retrieve only those columns required, never more – additional work is required by DB2 to send additional columns to your programs. Minimizing the number of columns in your SELECT statements will improve application performance.

- When joining tables, always provide join predicates. In other words, avoid Cartesian products.

- Favour Stage 1 predicates – another name for Stage 1 predicates is sargable predicates. A Stage 1 predicate is evaluated earlier in the process than a Stage 2 predicate, and therefore causes less data to be sent along for further processing by DB2. Stage 1 predicates tend to change with each new version of DB2 so make sure you know which version of DB2 you are using, which predicates are Stage 1, and which predicates are Stage 2. Refer to Figure 1 for a detailed depiction of Stage 1 versus Stage 2 processing.

- Favour indexable predicates – when a predicate is indexable then DB2 can use an index to satisfy that predicate. Not so, for a non-indexable predicate. Therefore, indexable predicates give DB2 more leeway for using indexes – which usually results in better performance.

- Avoid tablespace scans for large tables.

- Avoid sorting if possible by creating indexes for ORDER BY and GROUP BY operations.
And, let’s face it, even when using a black box some technicians in your organization still have to understand SQL – namely the writer(s) of the black box code. Because all of the SQL is coded in the black box program (or programs), someone has to be capable of writing efficient and effective SQL inside of the black box program. Which brings us to our next consideration.

SHORTCUTS MAKE FOR POOR PERFORMANCE

The SQL programmers in charge of writing the black box code will inevitably introduce problems into the mix. This is because of simple human nature – and because of most technicians’ desire to find shortcuts. But SQL shortcuts can lead to poor performance.

The black box inevitably will deviate from the standards and procedure of good SQL development. For example, let’s assume that there are three application programs and each one of them

![Figure 1: Stage 1 versus Stage 2 processing](image-url)
needs to retrieve customer information by area code. Program 1 needs the customer name and address, program 2 requires customer ID, name, and phone number, and program 3 requires customer ID, name, and type. This is properly coded as three different SQL requests (each one in its own program). For program 1 we would write:

```sql
SELECT FIRST_NAME, LAST_NAME, ADDRESS, CITY, STATE, ZIP
FROM CUSTOMER_TAB
WHERE AREA_CODE = :HV-AC;
```

For program 2 we would write:

```sql
SELECT CUST_ID, FIRST_NAME, LAST_NAME, PHONE_NUM
FROM CUSTOMER_TAB
WHERE AREA_CODE = :HV-AC;
```

And for program 3 we would write:

```sql
SELECT CUST_ID, FIRST_NAME, LAST_NAME, CUST_TYPE
FROM CUSTOMER_TAB
WHERE AREA_CODE = :HV-AC;
```

Of course, all of these SQL statements are remarkably similar, aren’t they? If we were in charge of writing the black box for these requests we would likely consolidate these three SQL statements into one statement like this:

```sql
SELECT FIRST_NAME, LAST_NAME, ADDRESS, CITY, STATE, ZIP, PHONE_NUM, CUST_TYPE
FROM CUSTOMER_TAB
WHERE AREA_CODE = :HV-AC;
```

Then our query will work for all three of these requests. When program 1 calls the black box we execute the query and return just the customer name and address; for program 2 we return just customer ID, name, and phone number; and for program 3 the black box returns only customer ID, name and type. We’ve coded a shortcut in our black box.

“So what?” you may ask. Well, this is bad program design because we are violating one of our SQL coding guidelines. Remember, SQL statements should retrieve only those columns required; never more. This is so because additional work is required by DB2 to send additional columns to your programs.
Minimizing the number of columns in your SELECT statements will improve application performance.

By coding shortcuts such as these into the black box you are designing poor performance into your DB2 applications. And a black box will use shortcuts. The example given here is a simple one, but even more complex shortcuts are possible in which WHERE clauses are coded so that they can be bypassed with proper host variables. For example, perhaps sometimes we need to query by area code and other times by area code and customer type. Well, we could code the CUST_TYPE predicate as a range something like this:

```
WHERE CUST_TYPE >= :HV1 and CUST_TYPE <= :HV2;
```

When we want to query for CUST_TYPE we simply provide the same value to both HV1 and HV2; when we do not want to query for CUST_TYPE we choose a larger value for HV1 than for HV2 (for example, 1 and 0). This effectively blocks out the CUST_TYPE predicate. Using tricks like this it is possible to cram a lot of different SQL statements into one – with the results usually being worse performance than if they were separate SQL statements.

**EXTRA CODE MEANS EXTRA WORK**

Additionally, when you code a black box, your application will require more lines of code to be executed than without the black box. It is elementary when you think about it. The call statement in the calling program is extra and the code surrounding the statements in the black box that ties them together is extra. None of this is required if you just plug your SQL statements right into your application programs.

This extra code must be compiled and executed. When extra code is required – no matter how little or efficient it may be – extra CPU will be expended to run the application. More code means more work. And that means degraded performance.

**SQL IS ALREADY AN ACCESS METHOD**

The final argument I will present here is a bit of a philosophical
one. When you code a black box you are basically creating a data access method for your programs. To access data each program must call the black box. But SQL is already an access method – so why create another one?

Not only is SQL an access method but it is a very flexible and comprehensive access method at that. You will not be able to create an access method in your black box that is as elegant as SQL – so why try?

SUMMARY

Do not implement data access interfaces that are called by application programs instead of coding SQL requests as needed in each program. When a black box is used, the tendency is that short cuts are taken. The black box inevitably deviates from proper SQL development guidelines, requires additional work and additional code, and is just another access method that is not required. Do not get lost in the black box – instead, train your programmers to code efficient SQL statements right in their application programs. Your applications will thank you for it!

Craig S Mullins
Director, Technology Planning
BMC Software (USA)

© Craig S Mullins 2003

Calling the DSNWZP stored procedure from a REXX client program to display DSNZPARM parameters

INTRODUCTION

You can easily get a listing of your DB2 subsystems DSNZPARM and DSNHDECP modules by using the IBM-supplied stored procedure DSNWZP.

This article explains how to call the DSNWZP stored procedure from a REXX client program.
DSNWZP REXX CLIENT PROGRAM

/* REXX */
/* THIS REXX PROCEDURE CALLS THE DSNWZP IBM STORED PROCEDURE */
/* TO EXTRACT ACTIVE DSNZPARAM PARAMETERS */
/* */
/* RESULT STRING RETURNED BY DSNWZP: */
/* */
/* - "RECORDS" WITHIN THE STRING ARE DELIMITED BY THE LINE FEED */
/* (LF - X'25') CHARACTER */
/* */
/* - FIELDS WITH EACH "RECORD" ARE DELIMITED BY A FORWARD SLASH */
/* */
PARSE ARG SSID COMMAND /* GET THE SSID TO CONNECT TO */
/* AND THE DB2 COMMAND TO BE */
/* EXECUTED */

.parseFloat ARG SSID COMMAND /* GET THE SSID TO CONNECT TO */
/* AND THE DB2 COMMAND TO BE */
/* EXECUTED */

/* HEADER */
LINEO.1 = CALLING DSNWZP FOR DB2 SUBSYSTEM SSID "-" DATE('U') TIME()
LINEO.2 = " "
"EXECIO * DISKW SYSPRINT (STEM LINEO."
/* SET UP THE HOST COMMAND ENVIRONMENT FOR SQL CALLS. */
"SUBCOM DSNREXX" /* HOST CMD ENV AVAILABLE? */
IF RC THEN /* NO--MAKE ONE */
Ø
S_RC = RXSUBCOM('ADD','DSNREXX','DSNREXX')
/* CONNECT TO THE DB2 SUBSYSTEM. */
ADDRESS DSNREXX "CONNECT" SSID
IF SQLCODE ≠ Ø THEN CALL SQLCA
/* SAY "*** CONNECT = OK ***" */
PROC = 'DSNWZP'
RESULTSIZE = 32768
RESULT = LEFT('',RESULTSIZE,'')
/* CALL THE STORED PROCEDURE DSNWZP */
/* THE OUTPUT VARIABLE (RESULT) WILL CONTAIN THE RETURN AREA */
ADDRESS DSNREXX "EXECSQL",
"CALL" PROC "( :RESULT)"
IF SQLCODE < Ø THEN CALL SQLCA
/* SAY "*** CALL = OK ***" */
/* EXTRACT DSNZPARAM PARAMETERS */
K = 1
I = INDEX(RESULT, X2C(25))
DO WHILE ( I /= Ø)
   R.K = SUBSTR(RESULT, 1, I - 1)
   K = K + 1
   L = LENGTH(RESULT)
   RESULT = RIGHT(RESULT, L - I)
   I = INDEX(RESULT, X2C(25))
END

MACRO_O = ""
DO I = 1 TO K - 1
   R = R.I
   IF INDEX(R, '/') /= Ø THEN
      DO
         DO J = 1 TO 6
            II = INDEX(R, '/ ')
            P.J = SUBSTR(R, 1, II - 1)
            LI = LENGTH(R)
            R = RIGHT(R, LI - II)
         END
         P.7 = R
      END
      MACRO_N = SUBSTR(P.2, 1, 00009)
      IF MACRO_N /= MACRO_O THEN
         DO
            SAY MACRO_O MACRO_N
            LINEO.1 = " "
            LINEO.2 = MACRO_N
            "EXECIO 2 DISKW SYSPRNT (STEM LINEO.)"| , MACRO_O = MACRO_N
         END
         LINEO.1 = " "| ,
         SUBSTR(P.3, 1, 009)| ,
         SUBSTR(P.7, 1, 040)| ,
         SUBSTR(P.6, 1, 040)
         "EXECIO 1 DISKW SYSPRNT (STEM LINEO.)"| ,
      END
   END
END

ADDRESS DSNREXX "DISCONNECT"
IF SQLCODE /= Ø THEN CALL SQLCA
/* SAY "** DISCONNECT = OK **" */

ADDRESS DSNREXX "DISCONNECT"
IF SQLCODE /= Ø THEN CALL SQLCA
/* SAY "** DISCONNECT = OK **" */

ADDRESS DSNREXX "DISCONNECT"
IF SQLCODE /= Ø THEN CALL SQLCA
/* SAY "** DISCONNECT = OK **" */
S_RC = RXSUBCOM('DELETE', 'DSNREXX', 'DSNREXX') /* REMOVE CMD ENV */
RETURN

/**************************************************************************
/* ROUTINE TO DISPLAY THE SQLCA */
/**************************************************************************
SQLCA:
TRACE 0
SAY 'SQLCODE = ' SQLCODE
SAY 'SQLERRMC = ' SQLERRMC
SAY 'SQLERRP = ' SQLERRP
SAY 'SQLERRD = ' SQLERRD.1 ', ',
    SQLERRD.2 ', ',
    SQLERRD.3 ', ',
    SQLERRD.4 ', ',
    SQLERRD.5 ', ',
    SQLERRD.6
SAY 'SQLWARN = ' SQLWARN.0 ', ',
    SQLWARN.1 ', ',
    SQLWARN.2 ', ',
    SQLWARN.3 ', ',
    SQLWARN.4 ', ',
    SQLWARN.5 ', ',
    SQLWARN.6 ', ',
    SQLWARN.7 ', ',
    SQLWARN.8 ', ',
    SQLWARN.9 ', ',
    SQLWARN.10
SAY 'SQLSTATE=' SQLSTATE
EXIT

IMPLEMENTATION

Installing DSNWZP

The DSNWZP stored procedure is supplied by IBM.

You should execute DB2.SDSNSAMP(DSNTIJSG) JCL in order to install DSNWZP:

/* */
DROP PROCEDURE SYSPROC.DSNWZP RESTRICT;
CREATE PROCEDURE SYSPROC.DSNWZP
  (OUT P10 VARCHAR (32000) CCSID EBCDIC)
  PROGRAM TYPE MAIN
  EXTERNAL NAME DSNWZP
  COLLID DSNWZP
  LANGUAGE ASSEMBLE
  RUN OPTIONS 'TRAP(ON), TERMTHDAC(UADUMP)' 
  PARAMETER STYLE GENERAL
  NO WLM ENVIRONMENT
COMMIT ON RETURN NO;
COMMIT;
GRANT EXECUTE ON PROCEDURE SYSPROC.DSNWZP TO PUBLIC;
/
//STEP002 EXEC PGM=IKJEFT01, DYNAMNBR=20
//SYSTSPRT DD SYSPUT=* 
//SYSPRPC DD DISP=SHR,DSN=MY.REXXLIB
//SYSPRINT DD SYSPUT=2
//SYSTSIN DD *
DSN SYSTEM (DB2S)
BIND PACKAGE (DSNWZP) MEMBER (DSNWZP) 
  ACTION (REPLACE) ISOLATION (CS) ENCODING (EBCDIC) 
  CURRENTDATA (NO) VAL (BIND) 
  LIBRARY ('DB2.SDSNDBRM') 
BIND PLAN (DSNWZP) PKLIST (DSNWZP, DSNWZP) 
  ISOLATION (CS) ENCODING (EBCDIC) ACTION (REPLACE)
/

JCL to call DSNWZP

//STEP1 EXEC PGM=IKJEFT01, DYNAMNBR=60
//SYSTSPRT DD SYSPUT=* 
//SYSPROC DD DISP=SHR,DSN=MY.REXXLIB
//SYSPRINT DD SYSPUT=2
//SYSTSIN DD *
DSNWZPCL DB2S
/

Output from DSNWZP

1CALLING DSNWZP FOR DB2 SUBSYSTEM DB2S  04/01/03 15:53:58

DSN6SYSYP
AUDITST 00000000000000000000000000000000 AUDIT TRACE
CONDBAT 0000000002 MAX REMOTE CONNECTED
CTHREAD 00030 MAX USERS
DLDFREQ 00005 LEVEL ID UPDATE FREQUENCY
PCLOSEN 00005 RO SW I TCH CHKPTS
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDBACK</td>
<td>00020</td>
<td>MAX BATCH CONNECT</td>
</tr>
<tr>
<td>IDFORE</td>
<td>00100</td>
<td>MAX TSO CONNECT</td>
</tr>
<tr>
<td>CHKFREQ</td>
<td>000050000</td>
<td>CHECKPOINT FREQ</td>
</tr>
<tr>
<td>MON</td>
<td>100000000</td>
<td>MONITOR TRACE</td>
</tr>
<tr>
<td>MONSIZE</td>
<td>00008192</td>
<td>MONITOR SIZE</td>
</tr>
<tr>
<td>SYNCVAL</td>
<td>NO</td>
<td>STATISTICS SYNC</td>
</tr>
<tr>
<td>RLFAUTH</td>
<td>SYSIBM</td>
<td>RESOURCE AUTHID</td>
</tr>
<tr>
<td>RLF</td>
<td>YES</td>
<td>RLF AUTO START</td>
</tr>
<tr>
<td>RLFERR</td>
<td>NOLIMIT</td>
<td>RLST ACCESS ERROR</td>
</tr>
<tr>
<td>RLFTBL</td>
<td>01</td>
<td>RLST NAME SUFFIX</td>
</tr>
<tr>
<td>MAXDBAT</td>
<td>00002</td>
<td>MAX REMOTE ACTIVE</td>
</tr>
<tr>
<td>DSSSTIME</td>
<td>00005</td>
<td>DATASET STATS TIME</td>
</tr>
<tr>
<td>EXTSEC</td>
<td>NO</td>
<td>EXTENDED SECURITY</td>
</tr>
<tr>
<td>SMFACCT</td>
<td>110000000000000000000000000000000</td>
<td>SMF ACCOUNTING</td>
</tr>
<tr>
<td>SMFSTAT</td>
<td>100000000000000000000000000000000</td>
<td>SMF STATISTICS</td>
</tr>
<tr>
<td>ROUTCDE</td>
<td>100000000000000000000000000000000</td>
<td>WTO ROUTE CODES</td>
</tr>
<tr>
<td>STORMXAB</td>
<td>00000</td>
<td>MAX ABEND COUNT</td>
</tr>
<tr>
<td>STORPROC</td>
<td>DB2SSPAS</td>
<td>DB2 PROC NAME</td>
</tr>
<tr>
<td>STORTIME</td>
<td>00180</td>
<td>TIMEOUT VALUE</td>
</tr>
<tr>
<td>STATIME</td>
<td>00030</td>
<td>STATISTICS TIME</td>
</tr>
<tr>
<td>TRACLOC</td>
<td>00016</td>
<td>RO SWITCH TIME</td>
</tr>
<tr>
<td>TRACSTR</td>
<td>000000000000000000000000000000000</td>
<td>TRACE AUTO START</td>
</tr>
<tr>
<td>TRACTBL</td>
<td>00016</td>
<td>TRACE SIZE</td>
</tr>
<tr>
<td>URCHKTH</td>
<td>000</td>
<td>UR CHECK FREQ</td>
</tr>
<tr>
<td>WLMENV</td>
<td></td>
<td>WLM ENVIRONMENT</td>
</tr>
<tr>
<td>LOBVALA</td>
<td>00000002048</td>
<td>USER LOB VALUE STORAGE</td>
</tr>
<tr>
<td>LOBVALS</td>
<td>00000002048</td>
<td>SYSTEM LOB VALUE STORAGE</td>
</tr>
<tr>
<td>LOGAPSTG</td>
<td>000</td>
<td>LOG APPLY STORAGE</td>
</tr>
<tr>
<td>DBPROTCL</td>
<td>PRIVATE</td>
<td>DATABASE PROTOCOL</td>
</tr>
<tr>
<td>PTASKROL</td>
<td>YES</td>
<td>EXTRA BLOCKS REQ</td>
</tr>
<tr>
<td>EXTRAREQ</td>
<td>00100</td>
<td>EXTRA BLOCKS SRV</td>
</tr>
<tr>
<td>EXTRASRV</td>
<td>00100</td>
<td>DEFAULT BUFFER POOL</td>
</tr>
<tr>
<td>TBSBPOOL</td>
<td>BP10</td>
<td>FOR USER DATA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEFAULT BUFFER POOL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FOR USER INDEXES</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LIMIT BACKOUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BACKOUT DURATION</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UR LOG WRITE CHECK</td>
</tr>
<tr>
<td>DSN6LOGP</td>
<td></td>
<td>NUMBER OF COPIES</td>
</tr>
<tr>
<td>TWOACTV</td>
<td>2</td>
<td>NUMBER OF COPIES</td>
</tr>
<tr>
<td>OFFLOAD</td>
<td>YES</td>
<td>RECORDING MAX</td>
</tr>
<tr>
<td>TWOBSDS</td>
<td>2</td>
<td>DEALLOC PERIOD</td>
</tr>
<tr>
<td>TWOARCH</td>
<td>2</td>
<td>READ TAPE UNITS</td>
</tr>
<tr>
<td>MAXARCH</td>
<td>0000000010</td>
<td></td>
</tr>
<tr>
<td>DEALLCT</td>
<td>00000:00000</td>
<td></td>
</tr>
<tr>
<td>MAXRTU</td>
<td>00002</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>OUTBUFF</td>
<td>000000400</td>
<td>OUTPUT BUFFER</td>
</tr>
<tr>
<td>WRTHRSH</td>
<td>00020</td>
<td>READ COPY2 ARCHIVE</td>
</tr>
<tr>
<td>ARC2FRST</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DSN6ARVP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLKSIZE</td>
<td>000028672</td>
<td>BLOCK SIZE</td>
</tr>
<tr>
<td>CATALOG</td>
<td>YES</td>
<td>CATALOG DATA</td>
</tr>
<tr>
<td>ALCUNIT</td>
<td>TRK</td>
<td>ALLOCATION UNITS</td>
</tr>
<tr>
<td>PROTECT</td>
<td>NO</td>
<td>ARCHIVE LOG RACF</td>
</tr>
<tr>
<td>ARCWTOR</td>
<td>NO</td>
<td>WRITE TO OPER</td>
</tr>
<tr>
<td>COMPACT</td>
<td>NO</td>
<td>COMPACT DATA</td>
</tr>
<tr>
<td>TSTAMP</td>
<td>NO</td>
<td>TIMESTAMP ARCHIVES</td>
</tr>
<tr>
<td>QUIESCE</td>
<td>00005</td>
<td>QUIESCE PERIOD</td>
</tr>
<tr>
<td>ARCRETN</td>
<td>00003</td>
<td>RETENTION PERIOD</td>
</tr>
<tr>
<td>ARCPF1</td>
<td>SDB2.SAVE.LOG01</td>
<td>ARCH LOG 1 PREFIX</td>
</tr>
<tr>
<td>ARCPF2</td>
<td>SDB2.SAVE.LOG02</td>
<td>ARCH LOG 2 PREFIX</td>
</tr>
<tr>
<td>PRIQTY</td>
<td>0000000150</td>
<td>PRIMARY QUANTITY</td>
</tr>
<tr>
<td>SECQTY</td>
<td>000000015</td>
<td>SECONDARY QTY</td>
</tr>
<tr>
<td>UNIT</td>
<td>SYSDA</td>
<td>DEVICE TYPE 1</td>
</tr>
<tr>
<td>UNIT2</td>
<td>NONE</td>
<td>DEVICE TYPE 2</td>
</tr>
<tr>
<td>SVOLARC</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>ARCWRTC</td>
<td>1011000000000000</td>
<td>WTOR ROUTE CODE</td>
</tr>
</tbody>
</table>

Editor’s note: the output would continue for more examples.

Systems Programmer
(France) © Xephon 2003

CAF interface with caller in amode 24 or 31 and more

When you want to execute a batch program in a job that needs to use DB2, there are four different ways to do it:

1. If your program also has to use an IMS database then you must use the DB2-DLI interface supplied within DB2. This is because DL/I is going to assume the integrity of the unit of work and the first program to start is a DL/I.

2. If your program also has to use MQSeries queues then you need an interface with RRS.
3 TSO tmp program IKJEFT01.

4 Call Attachment Facility (CAF). If your program accesses only DB2 and sequential files or VSAM, then you have the choice between the TSO tmp program, IKJEFT01, or the CAF (Call Attachment Facility). TSO tmp is easy to do, CAF is a little bit more complicated but can help programmers a lot.

In our shop we had an old CAF interface and a lot of old programs with old routines, all written in the years of 24-bit mode addressing.

Now, more and more, programs need to access both worlds – the old routines and DB2.

The CAF supplied by IBM is in amode 31, and our old stuff is in amode 24, so we get into trouble. We could re-linkededit the CAF supplied by IBM, but then we must maintain two different libraries for the CAF because of problems with other DB2 software that requires a CAF in amode 31.

In order to ease the migration and compatibility between our two worlds, we’ve reviewed our CAF interface to add some features as described here.

It is nearly transparent to the application programmers because they have nothing to do except during the link-edit, where they must specify the DB2 interface that they’re going to use.

The interface performs the following actions:

- It creates a stub to see whether the caller is in amode 24 or amode 31 and switches the amode. This means that we can keep the original CAF supplied by IBM with our old programs and the DB2 software.

- The connection with the DB2 subsystem has a name that may come from a load module in a library, from a parameter on the JCL EXEC card within the ‘parm=’ field, or in the DSNHDECP module found in the SDSNEXIT. When DB2 is down, it waits until DB2 comes up again.
• During the create thread it tries at least three times with different plan names. The plan name may come from the program name – the first two characters concatenated with ‘000BPL’. The plan name may equal the program name. Or the plan name may equal the DBRM supplied in the first SQL statement met.

• The program doesn’t have to take care of the connection.

• In case of trouble with an SQL statement, it will print with the DSNTIAR routine. This is done in a DD statement, allocated dynamically, whose name is CAFMSG.

• If the SQL code cannot continue, it will abend with a message in the syslog.

The JCL used to assemble the interface is:

```
//ASM01 EXEC PGM=ASMA90, REGION=1024K,
//       PARM='NODECK'
//SYSLIB DD DSN=SYS1.MODGEN, DISP=SHR
//       DD DSN=SYS1.MACLIB, DISP=SHR
//       DD DSN=SYS1.DSN710.SDSNMACS, DISP=SHR
//       DD DSN=SYS1.DSN710.SDSNSAMP, DISP=SHR
//SYSTUT1 DD UNIT=SYSDA, SPACE=(CYL,(1,1)), DISP=(NEW,DELETE)
//SYSTUT2 DD UNIT=SYSDA, SPACE=(CYL,(1,1)), DISP=(NEW,DELETE)
//SYSTUT3 DD UNIT=SYSDA, SPACE=(CYL,(1,1)), DISP=(NEW,DELETE)
//SYSLIN DD DSN=&&OBJ, DISP=(, PASS), UNIT=SYSDA, SPACE=(CYL,(1,1))
//SYSPRINT DD SYSOUT=*, DCB=(RECFM=FBM, LRECL=121, BLKSIZE=3509)
//SYPUNCH DD SYSOUT=*, DCB=(RECFM=FB, LRECL=80, BLKSIZE=3200)
//SYSLIB DD DSN=yourlibraryasm(ZCAF000), DISP=SHR
//*
//LINK EXEC PGM=IEWL,
//       PARM='XREF, LET, LIST, AMODE=31, REUS, RMODE=ANY, SIZE=(750K, 200K)'
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DSN=SYS1.CSSLIB, DISP=SHR
//SYSLMOD DD DSN=SYS1.DSN710.RUNLIB, DISP=SHR
//SYSLIN DD DSN=&&OBJ, DISP=(OLD, DELETE)
//*
NAME ZCAF000(R)
//ASM01 EXEC PGM=ASMA90, REGION=1024K,
//       PARM='NODECK'
//SYSLIB DD DSN=SYS1.MODGEN, DISP=SHR
//       DD DSN=SYS1.MACLIB, DISP=SHR
//       DD DSN=SYS1.DSN710.SDSNMACS, DISP=SHR
//       DD DSN=SYS1.DSN710.SDSNSAMP, DISP=SHR
```

// SYSLIN DD DSN=&&OBJ,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1))
// SYSPRINT DD SYSOUT=*,DCB=(RECFM=FBM,LRECL=121,BLKSIZE=3509)
// SYSPUNCH DD SYSOUT=*,DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
// SYSLIN DD *

TITLE 'ZCAFSSID DDB2 LOADMODULE WITH SSID NAME '

ZCAFSSID CSECT
ZCAFSSID AMODE 31
ZCAFSSID RMODE ANY
SSID DC CL4'DB2W' *
END ZCAFSSID *

// LINK EXEC PGM=IEWL,
//   PARM='XREF,LET,LIST,AMODE=31,RMODE=ANY,SIZE=(750K,200K)'
// SYSPRINT DD SYSOUT=* 
// SYSLIN DD DSN=SYS1.DSN710.RUNLIB,DISP=SHR
// DD *
NAME ZCAFSSID(R)

Link-edit statment for a program is:

// SYSLIB DD DSN=SYS1.DSN710.RUNLIB,DISP=SHR
// LKED.SYSLIN DD *
// INCLUDE SEQOBJ
// INCLUDE SYSLIB(ZCAF000)
// INCLUDE SYSLIB(DSNTIAR)
NAME userpgm(R)

JCL to execute your program is:

// DB2RUN PROC MEM=PGMLOAD, DB2=DBMX
// RUN EXEC PGM=MEM, PARM='DB2: &DB2'
// STEPLIB DD DSN=yourlibraryloadmod,DISP=SHR
// DD DSN=SYS1.dsn710.SDSNexit,DISP=SHR
// DD DSN=SYS1.dsn710.SDSNLOAD,DISP=SHR
// DD DSN=SYS1.dsn710.RUNLIB,DISP=SHR
// SYSPRINT DD SYSOUT=* 
// *CAFMSG DD SYSOUT=* allocated within the CAF.
// SYSUDUMP DD SYSOUT=* 
// PEND
// RUNDPD1 EXEC DB2RUN, MEM=Z5800B, DB2=DPDX

ZCAF000

TITLE 'ZCAF000 DB2 CAF INTERFACE FOR SQL IFI CAF'
* EXTERNAL ROUTINES :
* .DSNALI  DB2
* .DSNHLI2 DB2
* .DSNWLI2 DB2
* .DSNHLI DB2
* .DSNELI DB2

ZCAF000  CSECT
ZCAF000  AMODE     31
ZCAF000  RMODE     ANY
PRINT     GEN
DSNTIACN
PRINT     GEN
EJECT
ENTRY     DSNHLI
ENTRY     DSNHLI2
ENTRY     DSNWLI
ENTRY     DSNWLI2
ENTRY     DSNDDB2
* EXTRN     CAFCONNA
* EXTRN     CAFOPENA
* EXTRN     CAFCHEKA
* EXTRN     SQLCHEKA
* EXTRN     CAFCVTA
* R0     SYSTEM USE
* R1     SYSTEM USE
* R2     R1 PARAMETER LIST
* R3     WORK
* R4     ADR PARAMETER LIST FROM CALLER SQLPLIST OR IFCA
* R5     ADR MVS & DB2 CONTROL BLOCKS
* R6     FREE
* R7     WORK REGISTER
* R8     FREE
* R9     WORK REGISTER ADR SQLCA
* R10    SAVE AREA & BASE REGISTER FOR COMMON DATA
* R11    BASE REGISTER FIRST
* R12    RESERVED FOR PL/1
* R13    SAVE AREA
* R14    RETURN ADDRESS
* R15    RETURN CODE
EJECT
ZCAF000$  DC   C'ZCAF000'   *
DC   AL1(7)   *
DC   CL8'&SYSDATE'   *
DC   CL1' '   *
DC   CL8'&SYSTIME'   *
DC   CL1' '   *
EJECT   *

***********************************************************************
DSNHLI ENTRY POINT FOR SQL CALL. EXEC SQL ...

CAFDSNH CSECT

CAFDSNH AMODE 31

CAFDSNH RMODE ANY

DSNHLI DS DF ENTRY POINT IF PRECOMP WO CAF

DSNHLI2 DS DF ENTRY POINT FOR CAF

USING *,R15 R15 CURRENT BASE REGISTER

STM R14,R12,12(R13) SAVE REGS IN CALLER'S SAVEAREA

LA R10,DSNHLI SAVE R10 <- ADR DSNHLI'S SAVEAREA

ST R13,4(R10) LINK SAVEAREA CALLER IN DSNHLI

ST R10,8(R13) LINK SAVEAREA IN CALLING PGM

LR R13,R10 ESTABLISH OWN SAVE AREA

DSNR11A DS DF YES THEN CALLER IS IN RMODE 24

BSM R12,R11 BRANCH R11, R12 <- AMODE CALLER

DROP R15 R15 OUT OF USAGE

USING DSNMHMODE,R11 R11 CURRENT BASE REGISTER

DSNRMODE DS DF

L R7,R15 R7 <- R15 ENTRY POINT ADDRESS

SRL R7,24 R7 <- SHIFT BIT 8-31 OUTSIDE

LA R11,DSNMHMODE R11 <- BRANCH ADDRESS IN AMODE

LA R12,DSNHREST R12 <- RETURN ADDRESS

LTR R7,R7 ? R7 = 0

BZ DSNHR11A NO THEN CALLER IS IN RMODE ANY

O R11,=XL4'80000000' R11 <- SET HIGH BIT TO 1

DSNHR11A DS DF

BSM R12,R11 BRANCH R11, R12 <- AMODE CALLER

DROP R15 R15 OUT OF USAGE

USING DSNMHMODE,R11 R11 CURRENT BASE REGISTER

DSNRMODE DS DF

L R10,=A(CACVT) R10 <- ADR OF PTRDEF COMMON DATA

L R10,0(R10) R10 <- PTRDEF COMMON DATA

USING CACVT,R10 R11 CURRENT BASE REGISTER

LR R4,R1 R4 <- R1 ADR PARAMETER LIST

ST R1,PARSAVE PARSAVE <- R1 ADR PARAMETER LIST

MVI PGMRM,C'4' PGM CALLER RMODE ANY ABOVE 16MB

LTR R7,R7 ? R7 > 0

BP DSNHRMBL YES THEN CALLER IS IN RMODE ANY

MVI PGMRM,C'8' NO THEN CALLER RMODE IS 24

DSNHRMBL DS DF

MVI PGMAM,C'3' PGM CALLER AMODE 31

LR R7,R12 R7 <- R12 ADR(AMODE+RETURN)

SRL R7,31 R7 <- SHIFT BIT 1-31 OUTSIDE

LTR R7,R7 ? R7 = 0

BNZ DSNHINIT NO THEN CALLER IS IN AMODE 31

MVI PGMAM,C'2' PGM CALLER AMODE 24

NI PARSAVE,X'80' CLEAR PARSAVE HIGH ODER BYTE

DSNHINIT DS DF

L R4,PARSAVE R4 <- ADR PARAMETER LIST

XC RETRYC,RETRYC SET RETRY COUNTER TO ZERO

CLI STATUS,STATOPEN ? DB2 STATUS OPEN

BE DSNHLIO YES PROCESS SQL STMT
CLI        STATUS, STATFIRS   ? FIRST CALL
BE        DSNHLIØ   YES THEN CONNECT TO DB2
CLI        STATUS, STATDISC   ? DB2 STATUS DISCONNECTED
BNE       DSNHLI1   NO THEN OPEN THREAD

******************************************************************************

DSNHLIØ  EQU       *                   *
L         R2,Ø(,R4)   R2 <- Ø(R4)
USING     SQLPLDS,R2   R2 MAP SQLPL
MVC       PLANPGM,SQLPROGN   PLANPGM <- SQLPROGN
MVC       PLANDBRM,PLANPGM   SAVE PLAN NAME FOR TSO OR REXX
DROP       R2   SQLPLSDS OUT
L         R15,=A(CAFCONNA)   R15 <- ADR OF PTRDEF CONNDB2
L         R15,Ø(,R15)   R15 <- PTRDEF CONNDB2
BASSM     R14,R15   CALL CONNDB2

******************************************************************************

DSNHLI1  EQU       *                   *
CLI        STATUS, STATCONN   ? DB2 STATUS CONNECTED
BE        DSNHLI3   YES THEN PROCESS SQL STMT
CLI        STATUS, STATCLOK   ? DB2 STATUS CLOSE OK
BE        DSNHLI3   YES THEN PROCESS SQL STMT
CLI        STATUS, STATCLKO   ? DB2 STATUS CLOSE KO
BNE       DSNHLI4   NO THEN *

******************************************************************************

DSNHLI3  EQU       *                   *
L         R15,=A(CAFOPENA)   R15 <- ADR OF PTRDEF OPENDB2
L         R15,Ø(,R15)   R15 <- PTRDEF OPENDB2
BASSM     R14,R15   CALL OPENDB2

******************************************************************************

DSNHLI4  EQU       *                   *
CLI        STATUS, STATOPEN   ? STATUS DB2 OPENED
BNE       DSNHINIT   NO THEN RETRY

******************************************************************************

DSNHLIO  EQU       *                   *
MVC       LASTFUNC,SQL   LASTFUNC <- SQL
LR        R1,R4   R1 <- R4 ADR SQL STMT PARM LIST
L         R15,EPHLI   R15 <- DSNHLI ENTRY POINT
BASSM     R14,R15   PROCESS SQL STMT, R1 LOADED
LTR        R15,R15   ? R15 = Ø
BZ        DSNHLI01   YES THEN CHECK SQLCODE
ST        R15,RETCODE   RETCODE <- R15
L         R15,=A(CAFCHEKA)   R15 <- ADR OF PTRDEF CAFCHEK
L         R15,Ø(,R15)   R15 <- PTRDEF CAFCHEK
BASSM     R14,R15   CALL CAFCHEK
CLI        LASTEP,C'R'   ? RETRY SQL STMT
BE        DSNHLIO   YES THEN DSNHLIO

******************************************************************************

DSNHLIO1 EQU       *                   *
L         R1,PARSAVE   R4 <- PARSAVE
L         R2,Ø(,R1)   R2 <- Ø(R1)
USING     SQLPLDS,R2   R2 MAP SQLPL
L         R2,SQLCODEP   ADDRESS OF SQLCA IN SQL PARMLIST
DROP       R2   SQLPLDS OUT
USING     SQLCADS,R2   R2 MAP SQLCA
L R7,SQLCODE R7 <- SQLCODE
LTR R7,7 ? R7 >= 0, SQLCODE >= 0
BNM DSNHLI5 YES THEN RETURN TO CALLER
DS ØH NO THEN ANALYZE SQLCODE
L R15,=A(SQLCHEKA) R15 <- ADR OF PTRDEF SQLCHEK
L R15,Ø(,R15) R15 <- PTRDEF SQLCHEK
BASSM R14,R15 CALL SQLCHEK
CLI LASTEP,C'R' ? RETRY STATEMENT
BE DSNHLI0 YES THEN RETRY PREVIOUS STATEMENT

******************************************************************************

DSNHLI5 EQU * *
DROP R2 SQLCA OUT
BSM R0,R12 RESET CALLER'S AMODE

******************************************************************************

DSNHREST DS ØH *
L R13,4(,R13) R13 <- 4(R13) CALLING SAVEAREA
L R14,12(,R13) R14 <- 12(R13)
LM R1,R12,24(R13) R1-R12 <- 24(R13)
BR R14 RETURN

******************************************************************************

DSNHSAVE DS 18F SAVE AREA FOR DSNHLI ENTRY POINT
DC CL4'DSNH' EYE CATCHER
LTORG DROP R11,R10 R11 OUT OF USAGE

******************************************************************************

EJECT

******************************************************************************

* CONNECT TO DB2 IF STATUS IS ? *

******************************************************************************

CAFCONN CSECT *
CAFCONN AMODE 31 *
CAFCONN RMODE ANY *
CONND2 DS ØH *
USING *,R15 R15 CURRENT BASE REGISTER
STM R14,R12,12(R13) SAVE REGS IN CALLER'S SAVEAREA
LR R11,R15 R11 <- R15 ENTRY POINT
DROP R15 R15 OUT
USING CONND2,R11 R11 CURRENT BASE REGISTER
LA R10,CONNSAVE R10 <- ADR DSNHLI'S SAVEAREA
ST R13,4(R10) LINK SAVEAREA CALLER IN DSNHLI
ST R10,8(R13) LINK SAVEAREA IN CALLING PGM
LR R13,R10 ESTABLISH OWN SAVE AREA

******************************************************************************

L R10,=A(CAFCVTA) R10 <- ADR OF PTRDEF COMMON DATA
L R10,Ø(,R10) R10 <- PTRDEF COMMON DATA
USING CAFCVT,R10 R11 CURRENT BASE REGISTER
MVI CONSLBK,C' ' SET MESSAGE TO BLANK
MVC CONSOOTH,CONSLBK *
CLI STATUS,STATOPEN ? DB2 STATUS = C (OPENED)
BE CONNDB99 YES THEN RETURN
CLI STATUS,STATCONN ? DB2 STATUS = C (CONNECTED)
BE CONNDB99 YES THEN RETURN
CLI       DSNDCAF,C‘Y’        ? DSNDCAF = Y
BE        CONNDB5Ø    YES THEN CONNDB5Ø LOAD CAF
*****************************************************************************
*  LOAD CAF AND CONNECT TO DB2 IF UNSUCCESSFUL THEN DISCONNECT & DELETE*
*****************************************************************************
CONNDB1Ø DS ØH                  *
L         R5,CVTPTR      R5 <- ADR CVT
USING     CVT,R5            R5 MAP CVT
L         R5,CVTCBP       R5 <- TCBP
DROP      R5                CVT OUT
L         R5,4(RØ,R5)      POINT TO TCB PSATOLD
USING     TCB,R5            R5 MAP TCB TASK CB
L         R7,TCBFSA       POINT TO FIRST SAVE AREA
L         R3,TCBTIO       POINT TO TIODS
L         R5,TCBJSCB      POINT TO JSCB
DROP      R5                TCB OUT
USING     IEZJSCB,R5       R5 MAP JSCB JOBSTEP CB
MVC       EXECPGM,JSCBPGMN EXECPGM <- PGM NAME FROM JSCB
MVC       PLANPGM,JSCBPGMN PLANPGM <- PGM NAME FROM JSCB
MVC       PLANPKGH,JSCBPGMN PLANPKG <- PGM NAME HEADER 1-2
DROP      R5                IEZJSCB OUT
USING     TIOT1,R3          R3 MAP TIOT
MVC       JOBN,TIOCNJOB    JOBN <- JOBNAME FROM TIOT
MVC       STPN,TIOCSTEP    STEP <- STEP & PROC NAME
DROP      R3                TIOT OUT
CLC       PLANPGM(6),=C‘IKJEFT’ ? TSO TMP
BNE       CONNDB2R        NO THEN TEST REXX
MVI       ENVIR,ENVTSO    SET INDICATOR TO TSO
B         CONNDB2X         SKIP TO NEXT
CONNDB2R EQU       *
CLC       PLANPGM(6),=C‘IRXJCL’ ? REXX BATCH
BNE       CONNDB2X        NO THEN SKIP TO BATCH
MVI       ENVIR,ENVREXX   SET INDICATOR TO REXX
CONNDB2T EQU       *
MVC       PLANPGM,PLANDBRM PLAN <- FOUND IN SQL PARM LIST
*****************************************************************************
EJECT
*****************************************************************************
CONNDB2X EQU       *
LOAD       EP=ZCAFSSID,ERRET=CONNDB2Ø
LR         R5,RØ            RØ <- RØ ADR ZCAFSSID LOADMOD
CLI        Ø(R5),C’ ’       ? SSID BLANK (FIRST CHAR)
BE         CONNDB2Ø         YES THEN LOAD DSNHDECP
MVC        SS1D,Ø(R5)       MOVE SS1D NAME
B          CONNDB5Ø         BRANCH TO CONNECT
CONNDB2Ø EQU       *
LR         R5,R7            R7 <- R7 (TCBFSA) ADR FIRST SA
L          R5,24(R5)        R5 <- ADR PARAMETER LIST
L          R5,Ø(R5)         R5 <- PARAMETER LIST
XR         R7,R7            R7 <- Ø
LH R7,0(R5) R7 <- LENGTH PARM LIST
C R7,RC8 ? R7 < 8
BL CONNDB25 YES THEN SKIP IT
LA R5,2(R5) R5 <- ADR PARM, SKIP LENGTH
CLI 0(R5),C'/' ? START WITH / FOR PL1
BNE CONNDB15 NO THEN TEST FIRST FOUR BYTES
LA R5,1(R5) YES THEN R5 <- R5 + 1
CONNDB15 EQU *
CLC 0(4,R5),=C'DB2=' ? PARM KEYWORD DB2=
BE CONNDB16 YES THEN KEEP IT
CLC 0(4,R5),=C'DB2:' ? PARM KEYWORD DB2:
BNE CONNDB25 NO THEN SKIP IT
CONNDB16 EQU *
CLI 4(R5),C' ' ? SSID BLANK (FIRST CHAR)
BE CONNDB25 YES THEN LOAD ZCAFSSID
MVC SSID,4(R5) NO TAKE IT AS DB2 SSID
B CONNDB5Ø BRANCH TO CONNECT
***********************************************************************
EJECT
***********************************************************************
CONNDB25 EQU *
* LOAD EP=DSNHDECP, ERRET=CONNDB27
LR R5,RØ R5 <- RØ ADR DSNHDECP LOADMOD
ST R5,EPDECP EPDECP <- ADR DSNHDECP
USING DECP,R5 R5 MAP DSNHDECP MODULE
* CLC R1 BRVAL, DECPREL ? CORRECT RELEASE LEVEL
* BNE CONNDBAB NO THEN ABEND
CLI DECPSSID,C' ' ? SSID BLANK (FIRST CHAR)
BE CONNDB27 YES THEN TAKE FROM ASSEMBLY
MVC SSID, DECPSSID NO MOVE SSID NAME FROM DECP
DROP R5 DECP OUT
B CONNDB5Ø BRANCH TO CONNECT
CONNDB27 EQU *
* MVC MSGABND, =CL4' SSID' *
CLI SSID,C' ' ? SSID BLANK (FIRST CHAR)
BE CONNDBAB YES THEN ABEND
*******************************************************************************
EJECT
*******************************************************************************
CONNDB5Ø EQU *
* MVC MSGABND, =CL4' HL12' BUILD ERROR MESSAGE
LOAD EP=DSNL1I2, ERRET=CONNDBAB
ST RØ, EPHLI EPHLI <- ADR DSNL1I2
LR R7, RØ R1 <- RØ ADR(AMODE+EPHLI)
SRL R7,31 R7 <- SHIFT BIT 1-31 OUTSIDE
MVI HL1AM, C'2' PGM DSNL1I AMODE 24
LTR R7, R7 ? R7 = 0
BZ CONNDB2A YES THEN DSNL1I IS IN AMODE 24
MVI HL1AM, C'3' ELSE DSNL1I IS IN AMODE 31
MVC MSGABND, =CL4' WL12' BUILD ERROR MESSAGE
LOAD EP=DSNWLI2, ERRET=CONNDBAB
ST R0, EPWLI EPWLI <- ADR DSNWLI2
LR R7, R0 R1 <- R0 ADR(AMODE+EPWLI)
SRL R7, 31 R7 <- SHIFT BIT 1-31 OUTSIDE
MVI WLIAM, C'2' PGM DSNWLI AMODE 24
LTR R7, R7 ? R7 = 0
BZ CONNDB2B YES THEN DSNWLI IS IN AMODE 24
MVI WLIAM, C'3' ELSE DSNWLI IS IN AMODE 31

CONNDB2B EQU *

MVC MSGABND, =CL4' ALI' BUILD ERROR MESSAGE
LOAD EP=DSNALI, ERRET=CONNDBAB
ST R0, EPALI EPALI <- ADR CAF
LR R7, R0 R1 <- R0 ADR(AMODE+EPALI)
SRL R7, 31 R7 <- SHIFT BIT 1-31 OUTSIDE
MVI ALIAM, C'2' PGM DSNALI AMODE 24
LTR R7, R7 ? R7 = 0
BZ CONNDB2C YES THEN DSNALI IS IN AMODE 24
MVI ALIAM, C'3' ELSE DSNALI IS IN AMODE 31

CONNDB2C EQU *

******************************************************************************
EJECT
******************************************************************************

CONNDB7Ø EQU *

CLC PGMCAFAR, PGMCAFER ? INVALID AMODE RMODE PGM & CAF
BNE CONNDBOK NO THEN PROCESS CONNECTION
L R15, CAFCODE R15 <- -2000
MVC REASCODE(8), =CL8'*ARMODE'*
B CONNDBAB BRANCH TO ABEND

CONNDBOK DS OF *

MVC LASTFUNC, CONN LASTFUNC <- CONNECT
MVC MSGABND, =CL4' CONN' BUILD ERROR MESSAGE CONNECT
LA R1, CONNDBP *
B CONNDBC *

CONNDBP DS OF *

DC A(CONN) CONNECT
DC A(SSID) DB2 SSID
DC A(TECB) TERMINATION ECB
DC A(SECB) START-UP ECB
DC A(RIBPTR) CAF RELEASE INFORMATION BLOCK
DC A(RETCODE) RETURN CODE
DC A(REASCODE) REASON CODE
DC A(SRDURA) CURRENT DEGREE CONNECT->DISCON
DC A(EIBPTR+X'80000000') ENVIRONMENT INFORMATION BLOCK

CONNDBC EQU *

L R15, EPALI ADDRESS DSNALI BEFORE CALL
BASSM R14, R15 CALL DSNALI & SAVE-SWITCH AMODE

******************************************************************************
* CHECK RETURN CODE AND REASON CODE FROM CALL ATTACH *
******************************************************************************
CTR R15, R15
BNZ CONNDB75 NO THEN CHECK REASCODE
MVI STATUS, STATCONN YES CHG STATUS TO CONNECT
B CONNDB99 RETURN TO CALLER

CONNDB75 EQU *
L R15, =A(CAFCHEKA) R15 <- ADR OF PTRDEF CAFCHEK
L R15, Ø(R15) R15 <- PTRDEF CAFCHEK
BASSM R14, R15 CALL CAFCHEK
CLI LASTEP, C 'R' ? RETRY CONNECTION
BE CONNDB7Ø YES THEN CONNDB7Ø
CLI LASTEP, C 'O' ? ALREADY CONNECTED
BNE CONNDBAB NO THEN ERROR
B CONNDB99 RETURN TO CALLER

CONNDBAB EQU *

* ZCAFCON ESSID MSGA RETC REAC JOBNAME_
MVC CONSMGST, =CL8 'ZCAFCON '_
MVI CONSMSGL, C 'E' ERROR LEVEL
MVC CONSSSID, SSI D DB2 SSI D
MVC CONSJOBN, JOBN JOBNAME
MVC CONSPLAN, PLAN OE PLAN
MVC CONSTYPE, PLANTYPE PLAN TYPE
MVC CONSABND, MSGABND ABEND REASON
MVC CONSSTPN, STPN PROC STEP NAME
MVC CONSRETC, RETCODE RETURN CODE
MVC CONSREAS, REASCODE REASON CODE
WTO MF=(E, CONSOLE) *
ABEND X ' CAF ', , STEP, REASON=REASCODE

CONNDB99 DS ØH *
L R13, 4(R13) R13 <- 4(R13) CALLING SAVEAREA
L R14, 12(R13) R14 <- 12(R13)
LM R1, R12, 24(R13) R1-R12 <- 24(R13)
BR R14 RETURN

CONNSAVE DS 18F SAVE AREA FOR DSNHLI ENTRY POINT
DC CL4 'CONN' EYE CATCHER
CAFCONN A(X '80000000' +CAFCONN)

LTORG
DROP R11, R10 R11 OUT OF USAGE
***********************************************************************
EJECT
***********************************************************************

* ISSUE A CAF OPEN CALL (CREATE THREAD) *
***********************************************************************

CAFOPEN CSECT *
CAFOPEN AMODE 31 *
CAFOPEN RMODE ANY *
OPENDB2 DS ØH *
USING *, R15 R15 CURRENT BASE REGISTER
STM R14, R12, 12(R13) SAVE REGS IN CALLER'S SAVEAREA
LR R11, R15 R11 <- R15 ENTRY POINT
DROP R15 R15 OUT
USING OPENDB2, R11
R11 CURRENT BASE REGISTER
LA R10, OPENSEAVE R10 <- ADR DSNHLI'S SAVEAREA
ST R13, 4(R10) LINK SAVEAREA CALLING IN DSNHLI
ST R10, 8(R13) LINK SAVEAREA IN CALLING PGM
LR R13, R10 ESTABLISH OWN SAVE AREA

***********************************************************************
L R10, =A(CAFCVTA) R10 <- ADR OF PTRDEF COMMON DATA
L R10, Ø(, R10) R10 <- PTRDEF COMMON DATA
USING CAFCVT, R10 R11 CURRENT BASE REGISTER
CLI STATUS, STATOPEN ? DB2 STATUS = (OPENED)
BE OPENDB99 YES THEN RETURN
MVC LASTFUNC, OPEN LASTFUNC < OPEN THREAD
MVC PLANOPEN, PLANPKG PLANOPEN < PLAN PACKAGE
MVI PLANTYPE, C'K' PLANTYPE < K FOR PACKAGE

OPENDB2A DS ØH *
LINK EP=DSNALI, PARAM=(OPEN, SSID, PLANOPEN, RETCODE, REASCODE), VL=1
L R15, RETCODE R15 <- RETCODE
LTR R15, R15 ? R15 = Ø
BNZ OPENDB2Ø NO THEN CALL CHEKCAF

OPENDB2B DS ØH *
MVI STATUS, STATOPEN YES CHG STATUS < OPEN
MVI CONSBLK, C' ' SET MESSAGE TO BLANK
MVC CONSBLK, CONSBLK *
MVC CONSMSGT, =CL8'ZCAF001'
MVI CONSMSGL, C'I' ERROR LEVEL
MVC CONSSSID, SSID DB2 SSID
MVC CONSJOB, JOBNAME
MVC CONSEPGM, PLANPKG EXEC PGM
MVC CONSPKG, PLANOPEN PLAN
MVC CONSTYPE, PLANTYPE PLAN TYPE
MVC CONSSTPN, STPN PROC STEP NAME
WTO MF=(E, CONSOLE) *
B OPENDB99 RETURN TO CALLER

OPENDB2Ø EQU *

OPENDB2A *
L R15, =A(CAFCHEKA) R15 <- ADR OF PTRDEF CAFCHEK
L R15, Ø(, R15) R15 <- PTRDEF CAFCHEK
BASSM R14, R15 CALL CAFCHEK
CLI LASTEP, C'R' ? RETRY STATMENT
BE OPENDB2A YES THEN RETRY PREVIOUS STATMENT
CLI LASTEP, C'Ø' ? ALREADY OPENED
BE OPENDB2B YES THEN GO ON
CLC REASCODE, F30040 ? RETRY STATMENT
BE OPENDB25 YES THEN RETRY WITH OTHER PLAN
CLC REASCODE, F30034 ? RETRY STATMENT
BE OPENDB25 YES THEN RETRY WITH OTHER PLAN
B OPENDBAB NO THEN END WITH ABEND

OPENDB25 EQU *

CLI PLANTYPE, C'K' ? OPEN WITH PLANPKG PACKAGE
BE OPENDB3Ø YES THEN TRY WITH PLANPKG
CLI       PLANTYPE, C'P'       ? OPEN WITH PLANPGM PLAN
BE        OPENDB34            YES THEN TRY WITH PLANDBRM
B         OPENDBAB            ABEND UNABLE TO GET A PLAN

OPENDB30 EQU *           *
MVC       PLANOPE, PLANPGM   PLANOPEN <- PLANPGM
MVI       PLANTYPE, C'P'    PLANOPEN <- P FOR PLAN
B         OPENDB2A          *

OPENDB34 EQU *           *
MVC       PLANOPE, PLANDBRM PLANOPEN <- PLANDBRM
MVI       PLANTYPE, C'D'   PLANOPEN <- D FOR DBRM
B         OPENDB2A          *

**************************************************************

OPENDBAB EQU *           *
* ZCAFOPE ESSID MSGA      RETC REAC JOBNAME_
MVC       CONSMG T, =CL8'ZCAFOPE'
MVI       CONSMSGL, C'E'   ERROR LEVEL
WTO       MF=(E, CONSOLE)   *
ABEND     X'CAF', ,STEP, REASON=RETCODE

OPENDB99 DS ØH               *
L         R13, 4(R13)        R13 <- 4(R13) CALLING SAVEAREA
L         R14, 12(R13)       R14 <- 12(R13)
LM        R1, R12, 24(R13)   R1-R12 <- 24(R13)
BR        R14                RETURN

OPENS AVE DS 18F       SAVE AREA FOR DSNHLI ENTRY POINT
DC        CL4'OPEN'        EYE CATCHER
CAFOPENA DC A (X'80000000' +CAFOPEN)

**************************************************************

CAFCHEK  CSECT             *
CAFCHEK  AMODE     31       *
CAFCHEK  RMODE     ANY      *
CHEKCAF  DS        ØH      *
USING     * , R15       R15 CURRENT BASE REGISTER
STM       R14, R12, 12(R13) SAVE REGS IN CALLER'S SAVEAREA
LR        R11, R15        R11 <- R15 ENTRY POINT
DROP      R15              R15 OUT
USING     CHEKCAF, R11    R11 CURRENT BASE REGISTER
LA        R10, CHEKSAVE    R10 <- ADR DSNHLI'S SAVEAREA
ST        R10, 4(R10)      LINK SAVEAREA CALLING IN DSNHLI
ST        R10, 8(R13)      LINK SAVEAREA IN CALLING PGM
LR        R13, R10        ESTABLISH OWN SAVE AREA

**************************************************************

L         R10, =A(CAFCVTA)  R10 <- ADR OF PTRDEF COMMON DATA
L         R10, Ø(R10)      R10 <- PTRDEF COMMON DATA
USING     CAFCVTA, R10      R10 CURRENT BASE REGISTER

MV1       LASTEP,C'0'       LASTEP <- 0 RETURN CODE

** IF RETCODE = 0

L       R15, RETCODE       R15 <- RETURN CODE
LTR      R15, R15       ? R15 = 0
BZ      CHEKRTRN       YES THEN RETURN
MV1      CONSLBK, C' '       SET MESSAGE TO BLANK
MV1      CONSOTh, CONSLBK       *

** IF TECB IS POSTED WITH ABTERM OR FORCE

TM       TECB, POSTBIT       ? TECB POSTED
BZ       CHEKRCØ       NO THEN CHECK RETCODE
CLC      TECB+1(3), QUIESC6       ? STOP DB2 MODE (QUIESC6)
BE       CHEKRCØ       YES THEN CHECK RETCODE
MV1      CHECKMSG, =CL25 'STOP DB2 FORCE OR ABTERM'
MV1      LASTEP, C'5'       LASTEP <- 5 STOPPED
B       CHEKWTO       BRANCH TO WTO

** IF RETCODE > 0

CHEKRCØ       DS       ØH       *

** BUILD ERROR MESSAGE

L       RØ, REASCODE       RØ <- REASON CODE
UNPK      REASED(9), REASCODE(5)
TR      REASED(8), HEXTAB       TRANSLATE TO HEXA IN DISPLAY

ZCAFØØØ D SSID LAST----FUNC +RTCODE REASEDXXX JOBNAME_ PLANNAME
MV1      CONSMGT, =CL8 'ZCAFCHK'
MV1      CONSMGL, C' E'
MV1      CONSSSID, SSID       DB2 SSID
MV1      CONSFUNC, LASTFUNC
L       R7, RETCODE       R7 <- RETCODE
CVD      R7, DW       DW <- R7 RETCODE IN DECIMAL
MV1      RLEN6, REDIT       MOVE MASK IN MESSAGE
ED      RLEN6, DW+4       EDIT RETCODE
MV1      RLEN6+1, C' +'       SET DEFAULT SIGN
BC       2, CHEKCAF6       NO THEN KEEP +
MV1      RLEN6+1, C' -'       YES THEN SET -
CHEKCAF6       EQU       *
MV1      CONSRETC( L' RLEN6-1 ), RLEN6+1
MV1      CONSREAS, REASED       REASON CODE
MV1      CONSJOB, JOBN       JOBNAME
MV1      CONSPGM, PLANPGM       EXEC PGM
MV1      CONSSTPN, STPN       STEP & PROC NAME
MV1      CONSPLAN, PLANOE       PLAN NAME
MV1      CONSTYPE, PLANL6       PLAN TYPE
WTO       MF=(E, CONSOLE)       *
LR       R15, R7       R15 <- RETCODE

******************************************************************************

EJECT

** IF RETCODE = 4

C       R15, RC4       ? R15 = 4
BNE      CHEKC8       NO THEN TEST RC = 8
MV1      LASTEP, C'4'       LASTEP <- 4 RETURN CODE
CLC      REASCODE, C10823       ? RELEASE MISMATCH LEVEL
BE   CHEKRC4R   YES THEN SET MESSAGE
CLC   REASCODE, C10824   ? READY TO RESTART
BNE   CHEKRC4U   NO THEN UNKNOWN REASCODE
MVI   LASTEP, C’R’   LASTEP <- CHG TO RETRY
B   CHEKRTRN   RETURN
CHEKRC4R DS   ØH   *
MVC   CHECKMSG, =CL25’RC=4’   RELEASE DB2/CAF ERR’
B   CHEKWTO   BRANCH TO WTO
CHEKRC4U DS   ØH   *
MVC   CHECKMSG, =CL25’RC=4’   UNKNOWN REASONCODE’
B   CHEKWTO   BRANCH TO WTO
***********************************************************************
EJECT

** IF RETCODE = 8
CHEKRC8 DS   ØH   *
MVI   LASTEP, C’8’   LASTEP <- 8 RETURN CODE
C   R15, RC8   ? R15 = 8
BE   CHEKRC8C   YES THEN TEST REASCODE
MVI   LASTEP, C’C’   LASTEP <- C RETURN CODE
C   R15, RC12   ? R15 = 12
BNE   CHEKRC2H   NO THEN TEST RC 2000

** IF RETCODE = 8, 12
CHEKRC8C DS   ØH   *
CLC   REASCODE, F30002   HUNT FOR F30002
BE   CHEKDOWN   YES THEN CHECK DOWN
CLC   REASCODE, F30011   HUNT FOR F30011
BE   CHEKDOWN   YES THEN CHECK DOWN
CLC   REASCODE, F30012   HUNT FOR F30012
BE   CHEKDOWN   YES THEN CHECK DOWN
CLC   REASCODE, F30049   ? ALREADY CONNECTED
BNE   CHEKRC8D   RETURN TO CALLER
MVI   LASTEP, C’Ø’   YES THEN ACCEPT & GO ON
B   CHEKRTRN   RETURN TO CALLER

CHEKRC8D DS   ØH   *
CLC   REASCODE, F30055   ? MAX CONNECTIONS REACHED
BNE   CHEKRTRN   NO THEN RETURN TO CALLER
MVI   LASTEP, C’R’   YES THEN CHG TO RETRY
B   CHEKRTRN   YES THEN RETURN TO CALLER

** IF OPEN CALL
CLC   LASTFUNC, OPEN   ? OPEN CALL
BNE   CHEKWTO   NO THEN SKIP TRANSLATE
CLC   REASCODE(2), F3XXX   ? REASCODE TO TRANSLATE
BNE   CHEKWTO   NO THEN SKIP TRANSLATE

** TRANSLATE SQLCA ONLY FOR OPEN AND REASCODE ØØF3****
L   R2, Ø(, R4)   R2 <- ADR SQL PARAMETER LIST
USING   SQLPLDS, R2   R2 MAP SQL PARAMETER LIST
L   R2, SQLCODEP   ADDRESS OF SQLCA IN SQL PARMLIST
DROP   R2   SQLPLDS OUT
USING   SQLCADS, R2   R2 MAP SQLCA
LINK   EP=DSNALI, PARAM=(TRANSLAT,(2)), VL=1

DROP R2 SQLCA OUT
C R0,C10205 ? DID TRANSLATE FAIL
BNE CHEKWT0 YES THEN WTO
MVC CHECKMSG,=CL25'RC=8 CAF TRANSLATE ERROR'
B CHEKWT0 BRANCH TO WTO
CHEKDOWN DS ØH *
MVC CHECKMSG,=CL25'RC=8 DB2 DOWN WAIT WAKEUP'
WTO MF=(E, CONSOLE) *
WAIT ECB=SECB WAIT SOME SECONDS
MVC CHECKMSG,=CL25'RC=8 DB2 UP AGAIN, RETRY '
MVI LASTEP,C'R' LASTEP <- CHG TO RETRY
B CHEKRTRN RETURN TO CALLER
***********************************************************************
EJECT *
** IF RETCODE = 2Ø0
CHEKRC2H DS ØH *
MVI LASTEP,C'H' LASTEP <- H RETURN CODE
CLC RETCODE,RC2ØØ ? DB2 RC = 2Ø0
BNE CHEKRC24 NO THEN TEST DB2 RC = 2Ø4
CLC REASCODE,C10201 ? ALREADY CONNECTED
BNE CKC1Ø1Ø2 YES THEN ACCEPT & GO ON
MVI STATUS,STATCONN ? DB2 STATUS CONNECTED
MVI LASTEP,C'Ø' WE ACCEPT THE CODE & GO ON
B CHEKRC20 YES THEN ACCEPT & GO ON
CKC1Ø1Ø2 DS ØH *
CLC REASCODE,C10202 ? ALREADY OPENED
BNE CKC1Ø1Ø3 YES THEN ACCEPT & GO ON
MVI STATUS,STATOPEN ? DB2 STATUS CONNECTED
MVI LASTEP,C'Ø' WE ACCEPT THE CODE & GO ON
B CHEKRC20 YES THEN ACCEPT & GO ON
CKC1Ø1Ø3 DS ØH *
CLC REASCODE,C10203 ? NOT OPENED
BNE CKC1Ø1Ø4 YES THEN ACCEPT & GO ON
MVI STATUS,STATCONN ? DB2 STATUS CONNECTED
B CHEKRC20 YES THEN ACCEPT & GO ON
CKC1Ø1Ø4 DS ØH *
CLC REASCODE,C10204 ? NOT CONNECTED
BNE CHEKRC2W YES THEN ACCEPT & GO ON
MVI STATUS,STATFIRS ? DB2 STATUS CONNECTED
B CHEKRC20 YES THEN ACCEPT & GO ON
CHEKRC2W DS ØH *
MVC CHECKMSG,=CL25'RC=2ØØ STOP PROCESSING '
B CHEKWT0 BRANCH TO WTO
CHEKRC20 DS ØH *
B CHEKRTRN NO NORMAL RETURN TO CALLER
** IF RETCODE = 2Ø4
CHEKRC24 DS ØH *
CLC RETCODE,RC2Ø4 ? DB2 RC = 2Ø4
BNE CHEKRCUK NO THEN DB2 RC = UNKNOWN
MVI LASTEP,C'S' YES LASTEP <- CHG TO USER ERROR
MVC CHECKMSG, =CL25' RC=2O4 CAF SYSTEM ERROR '
B CHEKWTO BRANCH TO WTO

CHEKRCUK EQU * *
MVC CHECKMSG, =CL25 ' RC=??? UNKNOWN DB2 RC '
CHEKWTO EQU * *
WTO MF=(E, CONSOLE) *
B CHEKRTMN NO NORMAL RETURN TO CALLER

** IF IFI CALL
CHEKRTMN DS ØH *
L R13,4(R13) R13 <- 4(R13) CALLING SAVEAREA
L R14,12(R13) R14 <- 12(R13)
LM R1,R12,24(R13) R1-R12 <- 24(R13)
BR R14 RETURN

CHEKSSAVE DS 18F SAVE AREA FOR DSNHLI ENTRY POINT
DC CL4' CHEK' EYE CATCHER
CAFCHEKA DC A( X' 80000000' + CAFCHEK)

***********************************************************************
EJECT
***********************************************************************
* CHECK SQLCODE FROM SQL STMT
***********************************************************************

SQLCHECK CSECT *
SQLCHECK AMODE 31 *
SQLCHECK RMODE ANY *
CHEKSQL DS ØH *
USING *,R15 R15 CURRENT BASE REGISTER
STM R14,R12,12(R13) SAVE REGS IN CALLER'S SAVEAREA
LR R11,R15 R11 <- R15 ENTRY POINT
DROP R15 R15 OUT
USING CHEKSQL,R11 R11 CURRENT BASE REGISTER
LA R10,SQLCSAVE R10 <- ADR DSNHLI'S SAVEAREA
ST R13,4(R10) LINK SAVEAREA CALLING IN DSNHLI
ST R10,8(R13) LINK SAVEAREA IN CALLING PGM
LR R13,R10 ESTABLISH OWN SAVE AREA

***********************************************************************

L R10,=A(CAFCVT) R10 <- ADR OF PTRDEF COMMON DATA
L R10,Ø(R10) R10 <- PTRDEF COMMON DATA
USING CAFCVT, R10 R11 CURRENT BASE REGISTER

***********************************************************************

MVI LASTEP, C'?' LASTEP <- '? UNKNOWN ACTION
L R2,Ø(R4) R2 <- Ø(R4)
USING SQLPLDS, R2 R2 MAP SQLPL
L R2,SQLCODEP ADDRESS OF SQLCA IN SQL PARMLIST
DROP R2 SQLPLDS OUT
USING SQLCADS, R2 R2 MAP SQLCA
XR R6, R6 R6 <- Ø
L R7, SQLCODE R7 <- SQLCODE

* BUILD EXCEPTION TABLE COUNTER FOR SQLCODE BETWEEN -1 AND -999 *
* THIS AVOIDS FILLING THE SPOOL WITH A LOT OF DSNTIAR MESSAGES *

LPR R7, R7       R7 <- ABS(R7)
BCTR R7, R0      R7 <- R7 - 1
M R6, RC2       R7 <- R7 * 2
A R7, CPTSQLA    R7 <- R7 + ADR CPTSQLA
LH R6, Ø(R7)     R6 <- # SQLCODE ALREADY GOT
CH R6, RC1000    ? R6 > 100
BH CHEKSQLØ      YES THEN CHECK FURTHER
AH R6, RC1       R6 <- R6 + 1 INCREASE COUNTER
STH R6, Ø(R7)    CPTSQLA <- R6 SAVE IT
CHEKSQLØ EQU *
CH R6, RC1000    ? R6 > 100
BH CHEKSQL2      YES THEN SKIP DSNTIAR

******************************************************************************
* EJECT *
******************************************************************************
* FORMAT SQLCA WITH DSNTIAR AND PRINT IT *
******************************************************************************

DSNTIARØ DS ØH

CLI SWOPEN, C'Y' ? CAFMSG ALREADY OPEN
BE DSNTIAR2 YES THEN SKIP OPEN
XC S99AREA, S99AREA CLEAR S99AREA SET TO X'00'
LA R7, S99AREA  R7 <- S99AREA
USING S99RBP, R7 R7 MAP S99AREA
LA R1, 4(R7)    R1 <- R7 + 4
ST R1, S99RBPTR S99RBPTR <- R1 ADDRESS GETMAIN
OI S99RBPTR, S99RBPND S99RBPTR <- FLAG FIRST BYTE
DROP R7

LR R7, R1       R7 <- R1
USING S99RB, R7 R7 MAP S99RB
MVI S99RBLN, X'14' S99RBLN <- LENGTH 20 BYTES
MVI S99VERB, S99VRBAL S99VERB <- ALLOCATION VERB
LA R1, LS99RB, (R7) R1 <- PTR TO S99599X
USING S99RBX, R1 R1 MAP S99TUPL
LA R1, LS99RBX, (R1) R1 <- PTR TO S99
ST R1, S99TXTPP S99TXTPP <- S99 TEXT UNITS
LR R7, R1       R7 <- R1
DROP R7

USING S99TUPL, R7 R7 MAP S99TUPL
LA R1, S99DDNMK R7 <- ADR DDNAME KEY
ST R1, Ø(R7)    R1 <- ADR DDNAME KEY
LA R7, 4(R7)    R1 <- R1 + 4
LA R1, S99SYSOK R7 <- ADR SYSOUT KEY
ST R1,Ø(R7) R1 <- ADR SYSOUT KEY
OI Ø(R7),X'8Ø' FLAG LAST BYTE
LA R1,S99AREA R1 <- S99AREA
LR R7,R1 R7 <- R1
DYNALLOC *
LTR R15,R15 ? IS IT OK
BNZ DSNTIAR8 *
GETMAIN RC,LV=LCAFMSG,LOC=24 GETMAIN STORAGE FOR DCB BELOW
LTR R15,R15 ? GETMAIN OK
BNZ DSNTIAR8 NO THEN SKIP PRINTING
ST R1,ACAFMSGD ACAFMSGD <- R1 ADDRESS GETMAIN
LR R12,R1 R12 <- R1
MVC Ø(LCAFMSG,R12),CAFMSG * INIT DCB BELOW 16MB
OPEN ((12), (OUTPUT)), MODE=31
LTR R15,R15 ? OPEN OK
BNZ DSNTIAR8 NO THEN SKIP PRINTING
MVI SWOPEN,C'Y' SWOPEN <- Y SET SWITCH
DSNTIAR2 DS ØH *
L R12,ACAFMSGD R12 <- ACAFMSGD DCB CAFMSG
LA R1,RECS*RECL R1 <- RECS * RECL MSG AREA LEN
STH R1,MESSAGEL MESSAGEL <- R1
LINK EP=DSNTIAR, PARAM=((2), MESSAGE, LRECL), VL=1
LTR R15,R15 CHECK THE RETURN CODE
BZ DSNTIAR4 THE LENGTH IS OK, CONTINUE
MVC MESSAGE, MSGRETCD INITIALIZE THE MESSAGE
MVC MESSAGE, EDMASK INITIALIZE THE MESSAGE
CVD R15, PACK PACK <- R15 INTO DECIMAL
ED MESSAGE, PACK+FOUR CONVERT TO CHARACTERS
LA R4, MESSAGEX POINT TO THE OUTPUT DATA
L R12, ACAFMSGD R12 <- ACAFMSGD DCB CAFMSG
PUT (12), (4) INDICATE RETURN CODE FORMAT ERR
DSNTIAR4 DS ØH *
LA R7, RECS POSSIBLE NUMBER OF MESSAGES
LA R4, MESSAGET POINT TO THE TEXT
L R12, ACAFMSGD R12 <- ACAFMSGD DCB CAFMSG
DSNTIAR6 DS ØH *
PUT (12), (4) PRINT THE ERROR MESSAGE
A R4, LRECL R4 <- R4+LRECL POINT NEXT MSG
CLC ZERO(RECL,R4), BLANKS ? NEXT MESSAGE BLANK
BE DSNTIAR8 YES PRINT IS COMPLETE
BCT R7, DSNTIAR6 ? R7 <- R7-1 > Ø LOOP TO GET MSGS
DSNTIAR8 DS ØH *
* CLOSE (CAFMSG), MODE=31 *
**********************************************************************
EJECT *
CHEKSQL2 DS ØH TEST SQLCODE CRASH CODE
LA R7, CRASHNE R7 <- # CRASH CODE
LA R4, CRASHCT R4 <- ADR CRASH CODE TABLE
CHEKSQL4 DS ØH *
CLC SQLCODE, Ø(R4) ? IS SQLCODE IN CRASH CODE TABLE

BE  CHEKSQL6         YES THEN ABEND
BH  CHEKSQL8         NO AND SQLCODE HIGHER THEN OK
LA  R4, 4(R4)       R4 <- R4 + 4 NEXT CRASH CODE
BCT R7, CHEKSQL4     R7 <- R7 - 1 > 0 LOOP
B   CHEKSQL8        NO SQLCODE IN CRASH CODE TABLE

CHEKSQL6 DS ØH       *
  * ZCAFSQL E SSID MSGA RETC REAC JOBNAME_
  MVC CONSMSGT, =CL8 'ZCAFSQL '
  MVI CONSMGSL, 'E'
  MVC CONSSID, SID DB2 SID
  MVC CONSABND, =CL4 'SQLC'
  L  R7, SQLCODE     R7 <- RETCODE
  LPR R7, R7         R7 <- ABS(R7)
  MVC RLENG, EDIT    MOVE MASK IN MESSAGE
  ED RLENG, DW+4     EDIT RETCODE
  MVI RLENG+1, 'C'    SET DEFAULT SIGN
  BC 2, CHEKSQL$     NO THEN KEEP +
  MVI RLENG+1, 'C'    YES THEN SET -
  MVC µ RETC=(L'RLENG-1), RLENG+1
  MVC µ OBXN, JOBN
  MVC µ STPN, STPN
  MVC µ SQLSTATE
  WTO MF=(E, CONSOLE) *
  MVC µ SQLERRM, SQLERRM
  WTO MF=(E, CONSOLE) *
  ABEND X'CAF',, STEP, REASON=(R7)

CHEKSQL8 DS ØH       *
CHEKSQL9 DS ØH       *
  L  R13, 4(R13)     R13 <- 4(R13) CALLING SAVEAREA
  L  R14, 12(R13)    R14 <- 12(R13)
  LM R1, R12, 24(R13) R1-R12 <- 24(R13)
  BR R14             RETURN
SQLCSAVE DS 18F     SAVE AREA FOR DSNHLI ENTRY POINT
  DC CL4 'SQLC' EYE CATCHER

SQLCHEKA DC A('X'80000000' +SQLCHEK)
******************************************************************************

EJECT
  *
CRASHCT DC F' 804, 805, 818, 902, 906'
  DC F' 922, 923, 924, 927, 991'
CRASHNE EQU (*-CRASHCT)/4
RECS EQU 10    # OF RECORDS IN MESSAGE AREA
RECL EQU 121   LRECL FOR OUTPUT
DS 0F
LRECL DC AL4(RECL) FLAG TELLING DSNTIAR LRECL FOR MSG
  *
PRINT NOGEN
CAFMSG DCB D$ORG=PS, MACRF=PM, DDNAME=CAFMSG
  * RECFM=FB, LRECL=RECL, BLKSIZE=RECL*RECS
PRINT GEN

LCAFMSG EQU *'CAFMSG' LENGTH OF THE DCB
ACAFMGS GD DS F ADDRESS GETMAIN FOR DCB CAFMSG
ARECMSGD DS F ADDRESS GETMAIN FOR REC CAFMSG
SWOPEN DC CL1'N' SWITCH CAFMSG OPEN
RESTEV DC H'1' RETURN CODE FOR SEVERE ERROR
EDMASK DC XL8'402020202020202120' EDIT MASK FOR CONVERTING ROWS
BLANKTWO DC CL(RECL+ONE)'Ø' BLANK LINE FOR SPACING
BLANKS EQU BLANKTWO+ONE BLANKS FOR CLEARING AN AREA
MSGRETCD DC CL(RECL)' RETURN CODE FROM MESSAGE ROUTINE DSNTIAR'

LTORG

PACK DS D AREA FOR NUMERIC CONVERSION
PARM DS 4A PARAMETER AREA

*MESSEGX DS CL(RECL) MESSAGE TEXT
ORG MESSAGEX
DS CL42 SPACING
MESSAGE DS CL8 RETURN CODE
ORG

MESSAGE DS H,CL(RECS*RECL) MESSAGE FOR OUTPUT
ORG MESSAGE
MESSAGEL DS H LENGTH OF THE VARCHAR FIELD
MESSAGE DS ØCL(RECL) MESSAGE TEXT
DS CL12 SPACE
ORG

*. *

DS F
S99AREA DS CL100
S99DDNMD DC XL6'001000100010006'
S99DDNMDC DC CL8'CAFMSG'
S99SYSOK DC XL6'01800000000'
S99SYSOC DC CL1'*'
S99SYSOX DC XL6'00180001001'
S99SYSOY DC CL1'*'
LS99RB EQU 20
LS99RBX EQU 24
*. *

DS ØF *
CPTSQLA DC AL4(CPTSQL) *
CPTSQL DC 999'H' Ø' *
*. *

DROP R10,R11
EJECT

***********************************************************************
* DSNWLI ENTRYPONT FOR IFI CALLS.                                   *
***********************************************************************

CAFDSNW CSECT *
CAFDSNW AMODE 31 *
CAFDSNW RMODE ANY *
DSNWLI DS ØH *

DSNWLI2 DS ØH *
USING *, R15 R15 CURRENT BASE REGISTER
STM R14, R12, 12(R13) SAVE REGS IN CALLER’S SAVEAREA
LA R10, DSNWSAVE R10 <- ADR DSNWLI’S SAVEAREA
ST R10, 8(R10) LINK SAVEAREA CALLING IN DSNWLI
ST R10, 8(R13) LINK SAVEAREA IN CALLING PGM
LR R13, R10 ESTABLISH OWN SAVE AREA
***********************************************************************
LR R7, R15 R7 <- R15 ENTRY POINT ADDRESS
SRL R7, 24 R7 <- SHIFT BIT 8-31 OUTSIDE
LA R11, DSNWMODE R11 <- BRANCH ADDRESS IN AMODE
LA R12, DSNWREST R12 <- RETURN ADDRESS
LTR R7, R7 ? R7 = Ø
BZ DSNWR11A YES THEN CALLER IS IN RMODE 24
O R11, =XL4'80000000' R11 <- SET HIGH BIT TO 1
DSNWR11A DS ØH *
BSM R12, R11 BRANCH R11, R12 <- AMODE CALLER
DROP R15 R15 OUT OF USAGE
USING DSNWMODE, R11 R11 CURRENT BASE REGISTER
DSNWMODE DS ØH *
***********************************************************************
L R10, =A(CAFCVTA) R10 <- ADR OF PTRDEF COMMON DATA
L R10, Ø(R10) R10 <- PTRDEF COMMON DATA
USING CAFCVTA, R10 R10 CURRENT BASE REGISTER
LR R4, R1 R4 <- R1 ADR PARAMETER LIST
ST R1, PARSAVE SAVE PARAMETER LIST ADDRESS
MVI PGMRM, C'A' PGM CALLER RMODE ANY ABOVE 16MB
LTR R7, R7 ? R7 > Ø
BP DSNWRMBL YES THEN CALLER IS IN RMODE ANY
MVI PGMRM, C'2' NO THEN CALLER RMODE IS 24
DSNWRMBL DS ØH *
MVI PGMAM, C'3' PGM CALLER AMODE 31
LR R7, R12 R7 <- R12 ADR(AMODE+RETURN)
SRL R7, 31 R7 <- SHIFT BIT 1-31 OUTSIDE
LTR R7, R7 ? R7 = Ø
BNZ DSNWINIT NO THEN CALLER IS IN AMODE 31
MVI PGMAM, C'2' PGM CALLER AMODE 24
NI PARSAVE, X'80' CLEAR PARSAVE HIGH ODER BYTE
***********************************************************************
DSWINIT EQU *
L R4, PARSAVE R4 <- ADR PARAMETER LIST
CLI STATUS, STATOPEN ? DB2 STATUS OPEN
BE DSNWLIO YES PROCESS SQL STMT
CLI STATUS, STATFIRST ? FIRST CALL
BE DSNWLIO YES THEN CONNECT TO DB2
CLI STATUS, STATDISC ? DB2 STATUS DISCONNECTED
BNE DSNWLIO NO THEN OPEN THREAD
DSNWLIØ EQU *
L R15, =A(CAFCONNA) R15 <- ADR OF PTRDEF CONNDB2
L R15, Ø(R15) R15 <- PTRDEF CONNDB2
BASSM R14, R15 CALL CONNDB2

DSNWLI1 EQU * *
CLI STATUS, STATCONN ? DB2 STATUS CONNECTED
BE DSNWLI3 YES THEN PROCESS SQL STMT
CLI STATUS, STATCLOK ? DB2 STATUS CLOSE OK
BE DSNWLI3 YES THEN PROCESS SQL STMT
CLI STATUS, STATCLOKO ? DB2 STATUS CLOSE KO
BNE DSNWLI4 NO THEN *

DSNWLI3 EQU * *
L R15, =A(CAFOPENA) R15 <- ADR OF PTRDEF OPENDB2
L R15, Ø(R15) R15 <- PTRDEF OPENDB2
BASSM R14, R15 CALL OPENDB2

DSNWLI4 EQU * *
CLI STATUS, STATOPEN ? STATUS DB2 OPENED
BNE DSNWLI5 NO THEN *

DSNWLI5 EQU * *
BSM R0, R12 *

DSNWREST DS ØH *
L R13, 4(R13) R13 <- 4(R13) CALLING SAVEAREA
L R14, 12(R13) R14 <- 12(R13)
LM R1, R12, 24(R13) R1-R12 <- 24(R13)
BR R14 RETURN

DSNWSAVE DS 18F SAVE AREA
DC CL4'DSNW' EYE CATCATCHER
LTORG

DROP R11, R10 R11 OUT OF USAGE

***********************************************************************
EJECT *
***********************************************************************
* DSNDB2 ENTRYPINT FOR DB2 FUNCTIONs *
***********************************************************************

CAFDSND CSECT *
CAFDSND AMODE 31 *
CAFDSND RMODE ANY *
DSNDB2 DS ØH *
USING *, R15 R15 CURRENT BASE REGISTER
STM R14, R12, 12(R13) SAVE REGS IN CALLER'S SAVEAREA
LA R10, DSNDBSAVE R10 <- ADR DSNHLI'S SAVEAREA
ST R13, 4(R10) LINK SAVEAREA CALLING DSNHLI
ST R10, 8(R13) LINK SAVEAREA IN CALLING PGM
**LR** R13, R1Ø  **ESTABLISH OWN SAVE AREA**

***********************************************************************
**LR** R7, R15  **R7 <- R15 ENTRY POINT ADDRESS**
**SRL** R7, 24  **R7 <- SHIFT BIT 8-31 OUTSIDE**
**LA** R11, DSNDMODE  **R11 <- BRANCH ADDRESS IN AMODE**
**LA** R12, DSNDREST  **R12 <- RETURN ADDRESS**
**LTR** R7, R7  **? R7 = Ø**
**BZ** DSNDR11A  **YES THEN CALLER IS IN RMODE 24**
**O** R11, =XL4'80000000'  **R11 <- SET HIGH BIT TO 1**

**DSNDR11A** DS 0H  *****
**BSM** R12, R11  **BRANCH R11, R12 <- AMODE CALLER**
**DROP** R15  **R15 OUT OF USAGE**
**USING** DSNDMODE, R11  **R11 CURRENT BASE REGISTER**

***********************************************************************
**L** R1Ø, =A(CAFCVTA)  **R1Ø <- ADR OF PTRDEF COMMON DATA**
**L** R1Ø, Ø(R1Ø)  **R1Ø <- PTRDEF COMMON DATA**
**USING** CAFCVT, R1Ø  **R11 CURRENT BASE REGISTER**
**LR** R4, R1  **R4 <- ADR PARAMETER LIST**
**ST** R1, PARSAVE  **SAVE PARAMETER LIST ADDRESS**
**MVI** PGMRM, C' A'  **PGM CALLER RMODE ANY ABOVE 16MB**
**LTR** R7, R7  **? R7 > Ø**
**BP** DSNDRMBL  **YES THEN CALLER IS IN RMODE ANY**
**MVI** PGMAM, C' 2'  **NO THEN CALLER RMODE IS 24**

**DSNDRMBL** DS 0H  *****
**MVI** PGMAM, C' 3'  **PGM CALLER AMODE 31**
**LR** R7, R12  **R7 <- R12 ADR(AMODE+RETURN)**
**SRL** R7, 31  **R7 <- SHIFT BIT 1-31 OUTSIDE**
**LTR** R7, R7  **? R7 = Ø**
**BNZ** DSNDINIT  **NO THEN CALLER IS IN AMODE 31**
**MVI** PGMAM, C' 2'  **PGM CALLER AMODE 24**
**NI** PARSAVE, X'80'  **CLEAR PARSAVE HIGH ODER BYTE**

***********************************************************************
**EJECT**  *****

**DSNDINIT** EQU  *****
**L** R4, PARSAVE  **R4 <- ADR PARAMETER LIST**
**L** R4, Ø(R4)  **R4 <- PARAMETER LIST**
**MVI** DSNDCAF, C' Y'  **DSNDCAF <- Y**
**USING** DB2CAF, R4  **R4 MAP DB2CAF PARM LIST**
**CLI** CAFFUNC, CAFCONNE  **? DB2 CONNECT REQUEST**
**BNE** DSNDOPEN  **NO THEN TEST OPEN**
**MVC** SS1D, CAFSSID  **SSID <- CAFSSID PARM**
**L** R15, =A(CAFCONNA)  **R15 <- ADR OF PTRDEF CONNDB2**
**L** R15, Ø(R15)  **R15 <- PTRDEF OPENDB2**
**BASSM** R14, R15  **CALL OPENDB2**
**B** DSNRTRN  **RETURN TO CALLER**

**DSNDOPEN** DS 0H  *****
**CLI** CAFFUNC, CAFOPENE  **? DB2 OPEN REQUEST**
**BNE** DSNDCLOS  **NO THEN TEST CLOSE**
**MVC** PLANPGM, CAFPLAN  **PLANPGM <- CAFPLAN PARM**
L R15, =A(CAFOPENA) R15 <- ADR OF PTRDEF OPENDB2
L R15, Æ(R15) R15 <- PTRDEF OPENDB2
BASSM R14, R15 CALL OPENDB2
B DSNDRTRN RETURN TO CALLER

DSNDCL05 DS ØH *

CLI CAFFUNC, CAFCLOSE ? DB2 CLOSE REQUEST
BNE DSNDSC NO THEN TEST CLOSE
MVC CLOSOPT, CAFTERM CLOSOPT <- CAFTERM PARM

CLOSDB2 DS ØH *

MVC LASTFUNC, CLOS *
LINK EP=DSNALI, PARAM=(CLOS, CLOSOPT), VL=1
ST R15, CAFRETC *
ST R0, CAFREAC *
B DSNDRTRN RETURN TO CALLER

DSNDCLOS DS ØH *

CLI CAFFUNC, CAFDISC ? DB2 DISCONNECT REQUEST
BNE DSNDSC NO THEN INVALID REQUEST
MVC SSID, CAFSSID SSID <- CAFSSID PARM

CLOSDB2 DS ØH *

LINK EP=DSNALI, PARAM=(DISC), VL=1
ST R15, CAFRETC *
ST R0, CAFREAC *
DELETE EP=DSNALI DELETE CALL ATTACH
DELETE EP=DSNWLI2 DELETE CALL ATTACH
DELETE EP=DSNHLI2 DELETE CALL ATTACH
B DSNDRTRN RETURN TO CALLER

DSNDDIS DS ØH *

MVC CAFRETC, CAFCODE CAFRETC <- -2000
MVI CAFREAC, C'?' CAFREAC <- ???????
MVC CAFREAC+1(7), CAFREAC
DROP R4 DB2CAF OUT

DISCD2 DS ØH *

***************************************************************
BSM R0, R12 *

DSNDRTRN DS ØH *

***************************************************************

***************************************************************

***************************************************************

* DECLARES
***************************************************************
CAFCVT CSECT *
CAFCVT AMODE 31 *
CAFCVT RMODE ANY *
USING ",R10 R10 CURRENT BASE REGISTER
CAFCVTA DC A(X'80000000' +CAFCVT)
CONN DC CL12 'CONNECT' *
DISC DC CL12 'DISCONNECT' *
OPEN DC CL12 'OPEN' *
CLOSE DC CL12 'CLOSE' *
TRANSLAT DC CL12 'TRANSLATE' *
SQL DC CL12 'SQL' *
IFI DC CL12 'IFI' *
SYNC DC CL4 'SYNC' *
ABRT DC CL4 'ABRT' *
XI D DS CL4 *
XCAF DC CL4 'CAF ' *
CAFCODE DC F'-2000' SQLCODE FOR CAF ERRORS
EYEEAAAA DC CL4 'AAAA' SQLCODE FOR CAF ERRORS
JOBN DS CL8 JOBNAME TO DISPLAY IN MSG WTO
STPN DS CL16 STEP & PROC NAME
EXECPGM DS CL8 PROGRAM NAME
PLANOPEN DS CL8 PLAN NAME USED FOR OPEN THREAD
PLANPGM DS CL8 PLAN NAME FROM EXEC PGM=
PLANPKG DS OCL8 PLAN NAME FROM PACKAGE
PLANPKGH DS CL2 PLAN NAME FROM PACKAGE HEADER
PLANPKGC DC CL6 '000BPL' PLAN NAME FROM PACKAGE CONSTANT
PLANDBRM DS CL8 PLAN NAME FROM EXEC SQL PARMLIST
PLANTYPE DS CL1 PLAN TYPE USED FOR OPEN
PLANTPKG EQU C'K' PLAN TYPE IS PACKAGE
PLANTPGM EQU C'P' PLAN TYPE IS PGM
PLANDBR EQU C'D' PLAN TYPE IS DBRM
SSID DC CL4' ' *
RETRYC DS F RETRY COUNTER
PGMER DS F RMODE ENTRY POINT > 0 ANY
DSNDCAF DC CL1 'N' DSNDCAF PROCESS FROM DSNDB2 CAF
PGMCAFER DC CL3 '3A2' PGM AM(31) RM(ANY) & CAF AM(24)
PGMCAFAR DS OCL3 PGM CAF AMODE RMODE AMODE
PGMAM DC CL1 '? ' AMODE CALLING PROGRAM
PGMRM DC CL1 '? ' RMODE CALLING PROGRAM
ALIAM DC CL1 '? ' AMODE DSNALI PROGRAM
ALIRM DC CL1 '? ' RMODE DSNALI PROGRAM
HLIAM DC CL1 '? ' AMODE DSNHLI PROGRAM
WLIAM DC CL1 '? ' AMODE DSNWLI PROGRAM
ENVIR DC CL1 'B' ENVIRONMENT INDICATOR B I T
ENVBATCH EQU C'B' BATCH WITH EXEC PGM
ENVTSO EQU C'T' TSO WITH EXEC PGM=IKJEFT
ENVRXX EQU C'R' REXX WITH EXEC PGM=IRXJCL
STATUS DC CL1 '? ' STATUS
STATFIRS EQU C'?' FIRST TIME
STATCONN EQU C'C' CONNECTED WITH DB2
STATDISC EQU C'D'                DISCONNECTED WITH DB2
STATOPEN EQU C'O'                 OPEN THREAD CREATED
STATCLOK EQU C'E'                 CLOSE OK THREAD ENDED
STATCLKO EQU C'A'                  CLOSE KO THREAD ENDED
LASTEP  DC  CL1'?'                 LAST ENTRY POINT
LASTEPR EQU C'R'                  RETRY STATEMENT
LASTEPB EQU C'B'                 ROLLBACK AND RETURN TO MVS
LASTEPC EQU C'C'                COMMIT AND RETURN TO MVS
LASTEPD EQU C'D'                DISCONNECT & RECONNECT
LASTFUNC DC  CL12' ', *
MSGABND DS  CL4                  *
MSGXXXX DS  CL8                  *
EYEBBBB DC  CL4 'BBBB'           SQLCODE FOR CAF ERRORS
XETCODE DC  F'Ø'                RETURN CODE FROM CAF OR TO MVS
ASQLCA DC  F'Ø'                  *
EPSQL DC  F'Ø'                  *
IFI CMD DC  CL12' ', *
PARESAVE DC  F'Ø'                    ADDRESS TO PARAMETER LIST
EPDECP DS  A                  DSNHDECP ENTRY POINT
EPALI DS  A                  CAF ATTACH ENTRY POINT
EPHLI DS  A                  HLI SQL STMT ENTRY POINT
EPWLI DS  A                  WLI IFI CMD ENTRY POINT
RCM999  DC  F' -999'            *
RCMI  DC  F' -1'                *
RCØ  DC  F' Ø'                  *
RC1  DC  F' 1'                  *
RC2  DC  F' 2'                  *
RC4  DC  F' 4'                  *
RC8  DC  F' 8'                  *
RC12 DC  F' 12'                 *
RC32 DC  F' 32'                 *
RC100 DC  F' 100'               *
RC200 DC  F' 200'               *
RC204 DC  F' 204'               *
QUIESCE DC  XL3'000008'          TECB POSTCODE: STOP DB2 MODE=Q
POSTBIT EQU X'40'                TECB POSTBIT
REASED DS  CL9                   EDITED REASON CODE
REDIT DC  X'4021212121212121'  EDIT PATTERN TO SUPPRESS ZEROS
RLENG DC  X'404040404040404'  EDIT PATTERN TO SUPPRESS ZEROS
EJECT *
C1Ø2Ø1  DC  XL4'0OC1Ø2Ø1'        *
C1Ø2Ø2  DC  XL4'0OC1Ø2Ø2'        *
C1Ø2Ø3  DC  XL4'0OC1Ø2Ø3'        CLOSE WHEN NOT OPEN
C1Ø2Ø4  DC  XL4'0OC1Ø2Ø4'         *
C1Ø2Ø5  DC  XL4'0OC1Ø2Ø5'        CALL ATTACH CAN NOT TRANSLATE
C1Ø2Ø3  DC  XL4'0OC1Ø2Ø3'        CALL ATTACH GET RELEASE MISMATCH
C1Ø2Ø4  DC  XL4'0OC1Ø2Ø4'        CALL ATTACH READY FOR MORE INPUT
F3XXXX  DC  XL2'0OF3'             REASON CODE TO TRANSLATE
F3Ø002  DC  XL4'0OF3002'          DB2 SUBSYSTEM NOT UP

F30011 DC XL4'00F30011' DB2 SUBSYSTEM NOT UP
F30012 DC XL4'00F30012' DB2 SUBSYSTEM NOT UP
F30025 DC XL4'00F30025' DB2 IS STOPPING (REASCODE)
F30034 DC XL4'00F30034' REASON CODE TO TRANSLATE
F30040 DC XL4'00F30040'
F30049 DC XL4'00F30049'
F30055 DC XL4'00F30055'
ORG *.239 *
HEXTAB DS 239C TRANSLATE TABLE
DC ' ' *
DC '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
TECB DS F TERMINATION ECB
SECB DS F START ECB
RIBPTR DS F RELEASE INFO BLOCK
RETCODE DS F RETURN CODE FROM CAF
REASCODE DS F REASON CODE FROM CAF
EI BPTR DS F EI BPTR CODE FROM CAF
SRDURA DC CL10' SRDURA(CD)' SRUDRA
DC '0' *
DC CL4' ZZZZ' *
EJECT *
ACONNDB2 DC A(CONNDB2) ADDRESS CONNDB2
ADISCB2 DC A(DISCDB2) ADDRESS DISCB2
AOPENDB2 DC A(OPENDB2) ADDRESS OPENDB2
ACLOSDB2 DC A(CLOSDB2) ADDRESS CLOSDB2
ACHEKCAF DC A(CHEKCAF) ADDRESS CHEKCAF
WAITIME DS 0D *
WAITH DS XL2 *
WAITM DS XL2 *
WAITS DS XL2 *
WAITT DS XL1 *
WAITH DS XL1 *
DW DS D *
SAVESNAP DS 5F *
WTOLOG WTO ' ZCAFLOG , ROUTCDE=(11) , MCSFLAG=( HRDCPY ) , MF=L '
CONSOLE WTO ' CONSOLE+4 , ROUTCDE=(2) , MCSFLAG=( HRDCPY ) , MF=L '
OR G CONSOLE+4
CONSSSID DS CL4 *
DS CL1 *
CONSSSPL DS CL4 *
DS CL1 *
Editor’s note: this article will be concluded next month.

Alain Piraux
System Engineer (Belgium)
BMC has announced new SmartDBA data management tools for DB2 UDB. SmartDBA Performance Management v2.5 provides event management, diagnostics, visualization, administration, space management, and tuning. SQL-BackTrack v3.0 gives users access to backup and recovery functions through the SmartDBA Web-console. The software also lets DBAs manage DB2 UDB, Oracle, and SQL Server databases together from a common SmartDBA Console.

Performance Management optimizes performance and availability of DB2 databases through an integrated set of expert DBA tools, all managed from a centralized Web console. v2.5 gets enhanced monitoring capabilities via integrated common alerts which enable DBAs to monitor, tune, and manage space within DB2 UDB databases.

The combined package enables DBAs to more easily resolve performance problems regardless of the cause, whether poor space utilization, poorly written SQL statements, or inefficient database configuration settings. Version 3.0 of the SQL-BackTrack automated backup and recovery tool is now integrated with SmartDBA console.

For further information contact: BMC, 2101 CityWest Blvd, Houston, TX 77042, USA. Tel: (713) 918 8800. URL: http://www.bmc.com/solutions/database.

---

Informatica has announced new load performance software for IBM DB2 Universal Database Enterprise Server Edition software via its PowerCenter 6.2 data integration product. Through parallelism technology, it can apparently insert data into multi-node DB2 databases ten to 20 times faster than previous releases, while adapting to changing database cluster configurations.

The tests were done on p690 servers against a variety of DB2 node configurations. The company successfully completed interoperability testing with IBM’s TotalStorage Enterprise Storage Server Model 800.

For further information contact: Informatica, 2100 Seaport Boulevard, Redwood City, CA 94063, USA. Tel: (650) 385 5000. URL: http://www.informatica.com/Products/data+integration/powercenter/default.htm.

---

IBM has announced DB2 Query Management Facility Version 8, which exploits new capabilities of DB2 V8 and has new data visualization, solution building, Web-enablement, and solution sharing capabilities.

New is DB2 QMFV8 is support for DB2 UDB V8 functionality, including DB2 Cube Views, long names, Unicode, and enhancements to SQL, drag-and-drop building of OLAP analytics, SQL queries, pivot tables, and other business analysis and reports, and visual data ‘appliances’.

For further information contact your local IBM representative. URL: http://www.ibm.com/qmf.