165

CICS update

August 1999

In this issue

CICS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: info@xephon.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions

Articles published in *CICS Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

CICS Update on-line

Code from *CICS Update* can be downloaded from our Web site at http://www.xephon.com/cicsupdate.html; you will need the user-id shown on your address label.

Editor
Robert Burgess

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

Auto-install for printers

Are you frustrated with unnecessary printer definitions because you don’t know which definition to delete? So were we – until we came up with this idea to define printers automatically as our clerks log on.

CICS REQUIREMENT

This solution will only work from CICS 4.1 upwards because of the CREATE command and the terminal auto-install program in the AOR. We run in an MVS 5.2 environment.

OUR PRINTER ENVIRONMENT

We have printers defined in our TOR for print screens (CSPK). We also have printer definitions in some of our AORs for CICS transactions that run on printers.

Our standard for printer names is as follows. The terminal ID is cut from the first four characters of the netname. The first three characters of the terminal ID is called a string. Each unique string has one printer defined to it. This printer, known as the default printer, always has ‘2’ in the fourth character. For example, terminals AF01, AF03, or AF08 all belong to printer AF02.

We use a terminal auto-install program to define all our terminals in the TOR. Also, in the AOR, we run the auto-install program to define a surrogate TCT entry when transaction routing executes. We have approximately 4,000 printers defined in our TCT. We are notified by the network team of each printer definition, but we are not always notified when a string of terminals are taken out of use. For special cases of print screen (CSPK) the printout needs to be printed elsewhere, rather than on the default printer. For this situation, we have a file (SPRN) that contains a list of override printers for each terminal ID.

OUR SOLUTION

From our auto-install program for terminals, we start our new transaction AUPR. Our auto-install program was built from the sample auto-install program DFHZATDX. The same transaction is activated from the TOR and the AOR. AUPR decides whether the default printer and/or the override printer need to be defined in this CICS for this terminal.

Printers for CSPK are needed only in the TOR. Virtual terminals (VIRT) from Netpass (session manager) don't have default printer definitions.

We execute a CICS ENQUEUE command on the printer name so that we don't have any conflicts if two terminals needing the same printer should log on at the same time.

An INQUIRE terminal command is executed to determine whether these printers were already defined in the CICS.

Netnames for the printers are read in from a VSAM file containing all the printers. This file is created anew by our VTAM team whenever changes occur in the VTAM definitions. The file contains the four characters of the terminal ID and the eight characters of the netname of all the printers. The file is called AUTOPRNT.

A CICS CREATE command is executed to define the printer. A message is written to the DCT queue CPLI.

This transaction will run in a separate TCLASS to prevent overloads in the morning when everyone signs on or in situations where the computer or CICS fails.

AN ALTERNATIVE SOLUTION

Before we implemented the start to AUPR in the terminal auto-install program, we had it running from our sign-on program. One program ran in the TOR and executed all the necessary creates for the TOR. It also had code that started another transaction for all the AORs that had transactions running on printers.

RESULTS

In our TOR, the maximum number of printers defined comes to 1,700. In any AOR the maximum number of printers defined comes to 500. We are now also free from the paperwork of defining printers.

SOURCE CODE

In our auto-install terminal program, the following code was added. Add the following data definitions:

```
DFHEISTG DSECT ,
USER_NETNAME  DS CL8
USER_MSG      DS CL12Ø
```

The following code is added in the auto-install section for the TOR:

```
         LH   R6,NETNAME_LENGTH  PICK UP NETNAME LENGTH
         LA   R7,8               SET LENGTH FOR COMPARE
         CR   R6,R7              NETNAME LONGER THAN 8 CHARS?
         BNH  NETNMOVE           ...NO, MOVE FIRST N CHARS
         MVC  USER_NETNAME(8),NETNAME
         B    STRTTRAN           START TRAN FOR PRINTER AUTOINSTALL
NETNMOVE BCTR R6,RØ              LENGTH-1 ==> R6
         MVC  USER_NETNAME,=CL8' '
         EX   R6,NETNMVC         SELECT NETNAME
         B    STRTTRAN           START TRAN FOR PRINTER AUTOINSTALL
*
NETNMVC  MVC  USER_NETNAME(Ø),NETNAME    EXECUTE MOVE
MSG1     DC CL12Ø'SYAUPR41:START TRAN FOR PRINTER AUTOINSTALL FAILED'
*
STRTTRAN EXEC CICS START TRANSID('AUPR')
                   FROM(USER_NETNAME)
                   LENGTH(L'USER_NETNAME)
                   NOHANDLE
         CLC  EIBRESP,DFHRESP(NORMAL)
         BE   RETURN
         MVC  USER_MSG,MSG1
         EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(USER_MSG) NOHANDLE
```

The following code is added in the section called function seven for install to shipped terminals:

```
         MVC   USER_NETNAME,=CL8' '
         L     R8,INSTALL_SHIPPED_NETNAME_PTR
         MVC   USER_NETNAME,Ø(R8)
         EXEC CICS START TRANSID('AUPR')
                   FROM(USER_NETNAME)
                   LENGTH(L'USER_NETNAME)
                   NOHANDLE
         CLC   EIBRESP,DFHRESP(NORMAL)
         BE    RETURN
         MVC   USER_MSG,MSG2
         EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(USER_MSG) NOHANDLE
         B     RETURN              EXIT PROGRAM
MSG2     DC CL12Ø'SYAUPR41:START TRAN FOR SHIPPED AUTOINSTALL FAILED'
```

This is the code for the AUPR transaction:

```
AUPRNTR: PROC(CA_PTR) OPTIONS(MAIN);
 DCL PLIXOPT CHAR(28) VAR STATIC EXTERNAL
                      INIT('ISASIZE(35ØØ) NOSTAE');

 /* TRAN: AUPR    PROGRAM: AUPRTOR                           */
 /* DESCRIPTION:  PROCESS FOR AUTOINSTALL FOR PRINTERS       */
 /*                    AS TERMINALS LOG ON TO CICS.          */
 /*—————————————————————————————————————————————————————————— */
 /*————————————————— */
 /* GET COMMON AREA  */
 /*————————————————— */
 DCL 1 GETNET,
       2 GETCOMM               CHAR(4),
       2 GETREM                CHAR(4),
 /*—————————————*/
 /* SPRN RECORD */
 /*—————————————*/
     SPRNPRNT                  CHAR(8Ø),
     SPRNTERM                  CHAR(4),
 /*——————————————*/
 /* WORKING AREA */
 /*——————————————*/
     (SUBSTR,ADDR)             BUILTIN,
     DEFPRINT(2)               CHAR(4),
     ACQ                       FIXED BIN(31),
     ATLEN                     FIXED BIN(15),
     ATTRB                     CHAR(39),
     AUTO8Ø                    CHAR(8Ø),
     CA_PTR                    PTR,
     YESSPRN                   FIXED BIN(15),
     KEYTERM                   CHAR(4),
     MSGHELLO                  CHAR(1ØØ),
     NETNM                     CHAR(8),
     RESPCODE                  FIXED BIN(31),
     SYSIDNM                   CHAR(4),
     1 TIME,
       2 FIL1                  CHAR(1),
       2 HH                    CHAR(2),
       2 FIL2                  CHAR(1) INIT(':'),
       2 MM                    CHAR(2),
       2 FIL3                  CHAR(1) INIT(':'),
       2 SS                    CHAR(2),
     TIMESTR   BASED(ADDR(TIME)) CHAR(9),
     1 TIMEPIC,
       2 NULL                  PIC '9',
       2 HH                    PIC '99',
       2 MM                    PIC '99',
       2 SS                    PIC '99',
     TIMESTR2  BASED(ADDR(TIMEPIC)) PIC '(7)9',
     NOVIRT                   FIXED BIN(15),
     DECLEND                  CHAR(11) INIT('END DECLARE');
  /*—————————————————*/
  /*  INITIALIZATION */
  /*—————————————————*/
   TIMESTR2 = EIBTIME;
   TIME.HH = TIMEPIC.HH;
   TIME.MM = TIMEPIC.MM;
   TIME.SS = TIMEPIC.SS;
   DEFPRINT(1) = '    ';
   DEFPRINT(2) = '    ';
  /*————————————————————————*/
  /*    RECEIVE COMMON AREA */
  /*————————————————————————*/
   EXEC CICS RETRIEVE INTO(GETNET);
  /*—————————————————————————————————————————*/
  /*    CHECK IF TERMINAL IS FROM NETPASS */
  /*—————————————————————————————————————————*/
   NOVIRT = 1;
   IF GETCOMM = 'VIRT' THEN NOVIRT = 2;
  /*———————————————————————————————————*/
  /*    BUILD DEFAULT PRINTER NAME  */
  /*———————————————————————————————————*/
   DEFPRINT(1) = SUBSTR(GETCOMM,1,3)||'2';

  /*——————————————————————————————————————————————*/
  /*    CHECK SPRN(FOR CSPK) FILE ONLY IN TOR */
  /*——————————————————————————————————————————————*/
   YESSPRN = 1;
   EXEC CICS ASSIGN SYSID(SYSIDNM);
   IF SUBSTR(SYSIDNM,4,1) ¬= 'A'
      THEN DO;
      EXEC CICS READ FILE('SYPVPRNT') INTO(SPRNPRNT) RIDFLD(GETCOMM)
                RESP(RESPCODE) NOHANDLE;
      SPRNTERM = SUBSTR(SPRNPRNT,5,4);
      IF DFHRESP(NORMAL) = RESPCODE THEN DO;
         YESSPRN = 2;
         DEFPRINT(2) = SPRNTERM;
         END;
      ELSE DO;
         IF DFHRESP(DISABLED)  = RESPCODE |
            DFHRESP(NOTOPEN)   = RESPCODE |
            DFHRESP(FILENOTFOUND) = RESPCODE
         THEN DO;
            MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                       ' SPRN FILE ERROR   ';
            EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
            END;
         IF DFHRESP(NOTFND)   ¬= RESPCODE
         THEN DO;
```

```
          MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
               ' SPRN UNKNOWN ERROR/ RESPCODE ='|| RESPCODE;
          EXEC CICS WRITEQ TD QUEUE('CSSL') FROM(MSGHELLO);
          END;
       END;
    END;
/*—————————————————————————————————————————*/
/*  DO THIS STEP 1 OR 2 TIMES DEPENDING   */
/*  ON WHETHER AN ENTRY WAS FOUND IN SPRN.*/
/*—————————————————————————————————————————*/
DO I = NOVIRT TO YESSPRN;
/*———————————————————————————— */
/*  CHECK IF PRINTER IS DEFINED  */
/*———————————————————————————— */
 KEYTERM = DEFPRINT(I);
 EXEC CICS ENQ RESOURCE(KEYTERM) LENGTH(4);
 EXEC CICS INQUIRE TERMINAL(KEYTERM) ACQSTATUS(ACQ)
        RESP(RESPCODE) NOHANDLE;
 IF DFHRESP(NORMAL) = RESPCODE THEN DO;
    MSGHELLO = 'AUPRTOR -'|| TIMESTR || ' REQUEST FROM '||
         GETCOMM || ' UNNECESSARY ' ||
         KEYTERM || ' ALREADY DEFINED';
    EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
    GO TO SKIPTURN;
    END;
 IF DFHRESP(TERMIDERR) ¬= RESPCODE THEN DO;
    MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
             ' RESPCODE = '|| RESPCODE || ' FOR PRINTER '
             || KEYTERM || ' FROM TERMINAL ' || GETCOMM;
    EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
    GO TO SKIPTURN;
    END;
 ELSE DO;
   /*———————————————————————————————————*/
   /*     GET NETNAME FROM VTAM FILE    */
   /*———————————————————————————————————*/
    NETNM = '        ';
    EXEC CICS READ FILE('AUTOPRNT') INTO(AUTO80) RIDFLD(KEYTERM)
           RESP(RESPCODE) NOHANDLE;
    IF DFHRESP(NOTOPEN) = RESPCODE |
       DFHRESP(FILENOTFOUND) = RESPCODE THEN DO;
       MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                 ' AUTOPRNT FILE NOT READABLE <————';
       EXEC CICS WRITEQ TD QUEUE('CSSL') FROM(MSGHELLO);
       MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                 ' AUTOPRNT FILE NOT READABLE <————';
       EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
       GO TO LEAVE;
       END;
    ELSE IF DFHRESP(DUPKEY) = RESPCODE |
       DFHRESP(NOTFND) = RESPCODE THEN DO;
       MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                    ' NETNAME NOT FOUND FOR PRINTER ' ||
                    KEYTERM  || ' FROM TERMINAL ' || GETCOMM;
         EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
         GO TO SKIPTURN;
         END;
      ELSE DO;
         NETNM = SUBSTR(AUTO8Ø,1,8);
         END;
     /*----------------------*/
     /*    CREATE PRINTER    */
     /*----------------------*/
      ATTRB = 'NETNAME('||NETNM||
              ') TYPETERM(LU3ØØØ)';
      ATLEN = 34;
      EXEC CICS CREATE TERMINAL(KEYTERM)
                ATTRIBUTES(ATTRB) ATTRLEN(ATLEN)
                RESP(RESPCODE) NOHANDLE;
      IF DFHRESP(NORMAL) = RESPCODE THEN DO;
         MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                     ' ' || KEYTERM ||
                    ' SUCCESSFULLY CREATED FOR ' || GETCOMM;
         EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
         END;
      ELSE DO;
         MSGHELLO = 'AUPRTOR -'|| TIMESTR ||
                    ' '||KEYTERM||
                    ' NOT CREATED,RESPCODE = '||
                    RESPCODE || ' FOR ' || GETCOMM;
         EXEC CICS WRITEQ TD QUEUE('CPLI') FROM(MSGHELLO);
         END;
      END;   /* END OF ELSE DO;  */
 SKIPTURN:
      EXEC CICS DEQ RESOURCE(KEYTERM) LENGTH(4);
   END;        /* END OF DO LOOP FOR 2 POSSIBLE PRINTERS */
 LEAVE:
   EXEC CICS RETURN;
  END AUPRNTR;
//*
//LKED.SYSLMOD DD DSN=CICS.LOADSYS,DISP=SHR
//LKED.SYSIN DD *
     NAME AUPRNTR(R)
/*
```

MESSAGES

The following are samples of the DCT messages that are written.

Terminal JFI4 logs on:

```
DFHZC5966 I 99/Ø3/17 Ø6:26:35 CICSPROD INSTALL STARTED FOR
```

```
       TERMINAL   (  JFI4  ) SYSID (CPRD) (MODULE : DFHBSTZ ).
DFHZC6935 I 99/Ø3/17 Ø6:26:35 CICSPROD AUTOINSTALL FOR
       TERMINAL JFI4 WITH NETNAME JFI4P3EØ USING MODEL OR TEMPLATE
           TYPLU22  SUCCESSFUL.
```

As a result of JFI4 logging on, AUPRNTR started to run and installed the printer JFI2:

```
DFHZC5966 I 99/Ø3/17 Ø6:26:35 CICSPROD INSTALL STARTED FOR
       TERMINAL   (  JFI2  ) SYSID (CPRD) (MODULE : DFHBSTZ ).
???? ??? AUPR SØ3      99.Ø76 Ø6.26.35 CREATE TERMINAL(JFI2)...
       AUTINSTMODE PRINTERCOPY(NO) ALTPRINTCOPY(NO)...
       TERMPRIORITY(Ø) INSERVICE(YES) AT
AUPRNTR - Ø6:26:35 JFI2 SUCCESSFULLY CREATED FOR JFI4
```

JFI1 logs on:

```
DFHZC5966 I 99/Ø3/17 Ø6:27:36 CICSPROD INSTALL STARTED FOR
       TERMINAL   (  JFI1  ) SYSID (CPRD) (MODULE : DFHBSTZ ).
DFHZC6935 I 99/Ø3/17 Ø6:27:36 CICSPROD AUTOINSTALL FOR TERMINAL JFI1
           TYPLU22  SUCCESSFUL.
```

As a result of JFI1 logging on, AUPRNTR started to run but the printer JFI2 was already defined:

```
AUPRNTR - Ø6:27:36 REQUEST FROM JFI1 UNNECESSARY JFI2 ALREADY DEFINED
```

Sandra Ben Dov, Nadav Tarshish, and Vitaly Miliavsky
Israel National Insurance Institute (Israel)

Analysing DSNC abends

The following article is based on COBOL II, CICS/ESA 4.1, and DB2 Version 5. Although field offsets can change from release to release, the basic method should also work with releases other than those mentioned.

Under certain circumstances, DB2 abends a CICS task and a DSNC dump is taken. This article will help you to find out the reason for the abend and also find which SQL statement caused the error. You can find the abending program and statement even when you have some COBOL programs statically linked together.

FINDING THE CAUSE OF THE ABEND

To find the cause of the abend, the first task is to find the DB2 Reason Code. This code is contained in the structure CLOT. You can find a description of the CLOT in member DSNWCBDS in the DB2 sample library.

If you have the internal trace running, the CLOT is easy to find by searching the second 'AP 01C0' trace entry (the default trace ID, if this is not changed in the RCT). This contains the actual CLOT in the seventh parameter, as shown in Figure 1. Note that the trace shown in Figure 1 is truncated.

In the text form, which is not shown here, the seventh parameter starts with the eyecatcher DSN2LOT. The trace will always contain the seventh parameter when an exception has occurred. If you always want to have the DSN2LOT contained in the trace, you must ensure that you have defined trace level two for the FC component (for further information, see the *Changes to Database Management* section of the *CICS/ESA Release Guide*).

Now, in offset X'64', you will find a 4-byte hex value, which is the so-called 'Work Word One' or 'Reason Code'. Taking this message

```
AP 01C0 USER  EVENT - USER-EXIT-PROGRAM-ENTRY

TASK-00151 KE_NUM-005C TCB-009C5D90 RET-8DB44B92 TIME-17:37:44.1090718125 INTERVAL
  1-0000 C4C2F240 60405CC5 E7C35C
  2-0000 C9D5D8D8 80008424
  3-0000 C4C5C6C1 E4D3E340
  4-0000 0D7EA09C
  5-0000 0EBDFA30
  6-0000 0EB58090
  7-0000 C4E2D5F2 D3D6E340 000B0000 00020A90 0EBDFA30 0EB58090 E4C3C9C3 E2D6D7D9
    0020 40404040 40404040 C9D5D8D8 00000000 00000000 00000000 00000000 C3C8E2C5
    0040 C7C1F0F0 E9E3F1E4 C1F0F1F7 CFA26D32 29790000 00810004 80008424 00000000
    0060 00000000 00F30034 C9D5D8D8 80008424 0D8001F8 00000000 00000000 00000000
    0080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
    00A0 C4C5C6C1 E4D3E340 01C00000 00000000 00E4C3C9 C3E2D6D7 D9050484 00000000
    00C0 00000000 0000
```

Figure 1: Example of CLOT

number (in this case X'00F30034'), you can consult the *DB2 Messages and Codes* manual to find the corresponding explanation.

Another useful field is in offset X'5B', called CLOTCFLG. This one-byte length field contains further information and is needed when the Reason Code was set to X'00'. Further explanation of the CLOTCFLG can also be found in the *DB2 Messages and Codes* manual, in the DSNC section of the appendix *CICS Transaction Abend/Dump Code*.

If the internal trace is not running (or does not contain the seventh parameter for the DSN2LOT), you will have to go through the TIE chain for the failing task, until you find the TIE containing the eyecatcher DSN2LOT in offset X'6C'. Because the CLOT is now embedded in the TIE, the offsets for the Reason Code and for the CLOTCFLG are now X'D0' and X'C7' respectively.

FINDING THE FAILING SQL STATEMENT

Sometimes, the failing SQL statement is of interest. This can be found by examining the parameter list, which is passed to DB2. Because I couldn’t find any hints on how to do this, I wrote a test program, which I forced to a DSNC abend. Because I know which SQL statement will fail, I can backtrack the usage of the parameter list. This method seems to be working well.

```
01 SQL-PLIST3.                                              BLW=0000+1E8 0CL40
  05 SQL-PLIST-CON   PIC S9(9) COMP-4 VALUE +2621440.       BLW=0000+1E8,0000000 4C
  05 SQL-CALLTYPE    PIC S9(4) COMP-4 VALUE +50.            BLW=0000+1EC,0000004 2C
  05 SQL-PROG-NAME   PIC X(8)         VALUE ‘INQQUERY’.     BLW=0000+1EE,0000006 8C
  05 SQL-TIMESTAMP-1 PIC S9(9) COMP-4 VALUE +372896824.     BLW=0000+1F6,000000E 4C
  05 SQL-TIMESTAMP-2 PIC S9(9) COMP-4 VALUE +23578896.      BLW=0000+1FA,0000012 4C
  05 SQL-SECTION     PIC S9(4) COMP-4 VALUE +1.             BLW=0000+1FE,0000016 2C
  05 SQL-CODEPTR     PIC S9(9) COMP-4.                      BLW=0000+200,0000018 4C
  05 SQL-VPARMPTR    PIC S9(9) COMP-4 VALUE +0.             BLW=0000+204,000001C 4C
  05 SQL-APARMPTR    PIC S9(9) COMP-4 VALUE +0.             BLW=0000+208,0000020 4C
  05 SQL-STMT-NUM    PIC S9(4) COMP-4 VALUE +202.           BLW=0000+20C,0000024 2C
  05 SQL-STMT-TYPE   PIC S9(4) COMP-4 VALUE +3.             BLW=0000+20E,0000026 2C
```

Figure 2: SQL Precompiler trace

```
00000000   00B46EC4 C6C8C5C9 E4E24040 40404040  00000000 00000000 00000000 00000000
00000020   00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
00000040   00202E98 40404040 002000D0 80000000  0D7EA21C 00000000 0D800030 0D7EA21C
00000060   8009C798 8D560350 0009C798 0D6BB000  8DB452C4 0D6BB05C 0D79D080 00000000
00000080   0D6BAF30 0D7EA030 00081DAC 8009BCD0  0009CCD0 00058388 00058080 00000000
000000A0   00000000 00200050 00200054 00000000  00000000
```

Figure 3: First save area in offset X'54'

For each EXEC SQL, the SQL Precompiler generates a unique structure, which is the first one in the parameter list. It contains several fields, including the DBRM name (contained in the field SQL-PROG-NAME) and the statement number (contained in the field SQL-STMT-NUM) of the related EXEC SQL (see Figure 2).

To find which SQL statement caused the error, we have to go through the save areas until we reach the save area from our program. This is done by starting with the EXEC interface user structure, which contains the first save area at offset X'54'. Figure 3 shows the character display (with the eyecatcher >DFHEIUS truncated).

```
00000000 00201F64 7CF0F0F0 00003FE0 00000208 40000000 *U0000151...000........* 0D800000
0D800230 00203000 00108001 00201FC0 00000000 8DB44B92 *.....................k* 0D800020
0D8001F8 0D800448 0DB44832 0DB443D4 00000000 0DB443D4 *...u...8......M......M* 0D800040
0DB44410 0D800260 0DB44660 0DB44400 C3F2E3C7 E34EF4F8 *........-..-..C2TGT+48* 0D800060
00202E98 000C219C 0D800238 00000000 0000064D 00000000 *../....q.........(....* 0D800080
```

Figure 4: Offset X'58'

At offset X'58', we have the pointer to the save area of our COBOL program (in this case X'0D800030'). Because the save area is part of the so-called TGT, you will also see the eyecatcher C2TGT+48. This is shown in Figure 4, which now has its left side truncated. This is useful for debugging, because it contains the base locators used by COBOL to address the fields in the working storage and linkage section.

```
ØDB447C8 ØDB449BA ØDB449BA ØDB449C6   *..................H..........F*   ØD8ØØ1CØ
ØØØØØØØØ ØØØØØØØØ 8D8ØØ448 8D8ØØ328   *.......H......................*   ØD8ØØ1EØ
```

Figure 5: Content of R1

In the save area at offset X'18', we have the content of R1. This is at address 0D800048, so, R1 contains the value 0D8001F8. This is the pointer to the parameter list (see Figure 5, left part truncated again).

At the address 0D8001F8, we have the pointer to the only parameter used (high order bit set), which points to the address 0D800448 (see Figure 6, left-side truncated).

```
ØØ28ØØØØ ØØ32C9D5  D8D8E4C5 D9E81639 F438Ø167 C91ØØØØ1 *.....INQQUERY..4...I..* ØD8ØØ44Ø
ØØØØØØØØ ØØCAØØØ3  ØØ28Ø8ØØ ØØ1EC9D5 D8D8E4C5 D9E81639 *.............INQQUERY.* ØD8ØØ46Ø
```

Figure 6: Pointer to address 0D800448

In offset X'06', we have the 8 bytes long DBRM name, in this case INQQUERY, and in offset X'24', the 2 byte statement number, X'00CA'. So, by browsing the DBRM INQQUERY and searching for X'00CA', we find the failing SQL statement, as shown in Figure 7.

```
DBRM............@........OPEN QUERY_CURSOR ....
CCDDØØØ2ØØØØØØØCØØØ1ØØØ1DDCD4DECDE6CEDEDD4ØØØØ44444444444444444444444444444444444
4294ØØØEØ1ØØØØØAØØØ4ØØØ26755Ø84598D349269ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
```

Figure 7: Finding the failing SQL statement

You will find the statement number in offset X'0E'. If you also want to know the host variables used by the program, the easiest way is to convert the statement number to decimal and then to search in the listing of the program.

Guido Rechsteiner
System Programmer
Swiss Securities Clearing Corporation (Switzerland)

Accessing CEDA from ISPF

INTRODUCTION

This article provides a facility for issuing any CEDA commands from an ISPF session.The mechanism that makes this possible comes from the IBM CICS Supportpac, CA1D, which can be downloaded free from the IBM CICS Web site (http://www.software.ibm.com/ts/cics/downloads/).

This provides two programs, one using the EXCI CALL interface and the other using the LINK interface, that can be called from REXX to execute programs in CICS via the EXCI. This facility uses the LINK interface.

DESCRIPTION

Commands are entered on an ISPF panel in exactly the same format as would be used if you were using CEDA on CICS. The command is passed in a COMMAREA to program SPGCEDA which then calls DFHEDAP. The outcome of the command is returned in the COMMAREA, then formatted and displayed on the panel.

An EXCI connection is needed in each CICS system in which the facility is to be used and the CICS SDFHEXCI load library is needed on the STEPLIB of the TSO logon PROC.

CICS CONNECTION AND SESSION DEFINITION

```
DEFINE CONNECTION(REXL) GROUP(TCTREXL)
DESCRIPTION(GENERIC PIPE FOR REXX EXCI)
       ACCESSMETHOD(IRC) PROTOCOL(EXCI) CONNTYPE(GENERIC)
       SINGLESESS(NO) DATASTREAM(USER) RECORDFORMAT(U) QUEUELIMIT(NO)
       MAXQTIME(NO) AUTOCONNECT(NO) INSERVICE(YES) ATTACHSEC(LOCAL)
       BINDSECURITY(NO) USEDFLTUSER(NO)
DEFINE SESSIONS(REXISESS) GROUP(TCTREXL)
DESCRIPTION(SESSIONS FOR SPECIFIC PIPE FOR REXX EXCI)
       CONNECTION(REXL) PROTOCOL(EXCI) MAXIMUM(Ø,Ø) RECEIVEPFX($)
       RECEIVECOUNT(1Ø) SENDSIZE(4Ø96) RECEIVESIZE(4Ø96)
       SESSPRIORITY(Ø) AUTOCONNECT(NO) BUILDCHAIN(YES) USERAREALEN(Ø)
       IOAREALEN(4Ø96,4Ø96) RELREQ(NO) DISCREQ(NO) NEPCLASS(Ø)
       RECOVOPTION(SYSDEFAULT)
```

SPGCEDA

```
*----------------------------------------------------------------------*
*                                                                      *
* CALL DFHEDAP (CEDA) PASSING A COMMAND RECEIVED IN THE FIRST 1Ø22     *
* BYTES OF THE COMMAREA. PASS ANY MESSAGES BACK TO THE CALLER IN THE   *
* BACK END OF THE COMMAREA. COMMAREA MUST BE 6ØØØ BYTES                *
*                                                                      *
* DFHEDAP EXPECTS A 5 FULLWORD COMMAREA :-                             *
*                                                                      *
*        WORD     POINTS TO                                            *
*         1       COMMAND BUFFER                                       *
*         2       COMMAND BUFFER LENGTH                                *
*         3       FLAG BYTE (MUST CONTAIN X'ØØ')                       *
*         4       RESPONSE BUFFER                                      *
*         5       RESPONSE BUFFER LENGTH                               *
*                                                                      *
         USING COMMAREA,R2
COMMAREA DSECT
COMMAND  DS   CL1Ø22
RESPONSE DS   CL4978
COMMALEN EQU  *-COMMAREA
*
         DFHREGS
*
         DFHEISTG
*
RESP     DS    F                  RESP FROM CICS COMMANDS
CEDAPARM DS    5F                 PARMS FOR DFHEDAP
*
SPGCEDA  DFHEIENT EIBREG=11,CODEREG=12,DATAREG=13
*
          CLC   EIBCALEN,ENOUGH    ENOUGH COMMAREA?
          BNE   THATS_ALL_FOLKS    NO - GET OUT NOW
          L     R2,DFHEICAP        GET COMMAREA ADDRESS
          LA    R1,COMMAND         GET COMMAND BUFFER ADDRESS
          ST    R1,CEDAPARM        PUT INTO PARM WORD 1
          LA    R1,COMMAND_LENGTH  GET COMMAND BUFFER LENGTH ADDRESS
          ST    R1,CEDAPARM+4      PUT INTO PARM WORD 2
          LA    R1,FLAG            GET FLAG ADDRESS
          ST    R1,CEDAPARM+8      PUT INTO PARM WORD 3
          LA    R1,RESPONSE        GET RESPONSE BUFFER ADDRESS
          ST    R1,CEDAPARM+12     PUT INTO PARM WORD 4
          LA    R1,RESPONSE_LENGTH GET RESPONSE BUFFER LENGTH ADDRESS
          ST    R1,CEDAPARM+16     PUT INTO PARM WORD 5
          EXEC CICS                                                     X
                LINK                                                    X
                PROGRAM('DFHEDAP')                                      X
                COMMAREA(CEDAPARM)                                      X
                LENGTH(COMMAREA_LENGTH)                                 X
                RESP(RESP)
```

```
THATS_ALL_FOLKS DS ØH
          EXEC CICS RETURN
*
ENOUGH   DC    AL2(COMMALEN)
COMMAND_LENGTH DC AL2(L'COMMAND)
COMMAREA_LENGTH DC AL2(2Ø)
RESPONSE_LENGTH DC AL2(L'RESPONSE)
*
FLAG     DC    X'ØØ'
*
         END
```

REXX ROUTINE CEDA

```
/********************************* REXX ******************************/
/*                                                                   */
/* Access CEDA from ISPF.                                            */
/*                                                                   */
/*********************************************************************/
nl = X2C("15")
program_name = "SPGCEDA"
address ispexec

/* Clear the normal paging keys so we can do our own  */
"VGET (ZPFØ7,ZPFØ8,ZPF19,ZPF2Ø)"
hpfØ7 = zpfØ7
hpfØ8 = zpfØ8
hpf19 = zpf19
hpf2Ø = zpf2Ø
zpfØ7 = ""
zpfØ8 = ""
zpf19 = ""
zpf2Ø = ""
"VPUT (ZPFØ7,ZPFØ8,ZPF19,ZPF2Ø)"

/* LIBDEF ISPLLIB to the library where CA1DLINK lives  */
"LIBDEF ISPLLIB DATASET",
  "ID('JSDCIC.SVG.CICSLOAD' 'SYSTAG.#1GAØ78.CICSLOAD')"

/* Here we go  */
"DISPLAY PANEL(CEDA)"
do while rc = Ø
  "VGET (RESPONSE) SHARED"

/* Clear the results area   */
  do i=1 to 14
    interpret "line"i" = ''"
  end
  select
```

```
/* ENTER pressed so we need to call SPGCEDA  */
    when response = "ENTER" then do
      message = ""
      start_at = 1

/* Prepare to call CA1DLINK  */
      cics = STRIP(cics)
      commarea = COPIES(" ",6ØØØ)
      commarea_len = LENGTH(commarea)
      parm = "PROGRAM("program_name") COMMAREA(COMMAREA)" ,
             "LENGTH("commarea_len") APPLID("cics")"
      commarea = OVERLAY(cedacmd,commarea)
      CALL CA1DLINK PARM

/* Check the response from CA1DLINK  */
      response = RIGHT(commarea,4978)
      length = C2D(SUBSTR(response,1,2))
      count = C2D(SUBSTR(response,3,2))
      severity = C2D(SUBSTR(response,5,2))
      sev1 = severity
      if severity > 4 then do
        rest = STRIP(SUBSTR(response,8))
        parse var rest message (NL) rest
      end

/* Get the lines of output from the buffer  */
      out_length = C2D(SUBSTR(response,length+1,2))
      out_count = C2D(SUBSTR(response,length+3,2))
      out_severity = C2D(SUBSTR(response,length+5,2))
      sev2 = out_severity
      rest = STRIP(SUBSTR(response,length+8))
      line. = ""
      lines = Ø
      parse var rest line (NL) rest
      do while rest ¬= ""
        lines = lines + 1
        line.lines = STRIP(line)
        parse var rest line (NL) rest
      end
      if line ¬= "" then do
        lines = lines + 1
        line.lines = STRIP(line)
      end
      call BUILD_SCREEN
    end

/* Page up  */
    when response = "UP" then do
      if start_at - 14 > Ø then start_at = start_at - 14
      call BUILD_SCREEN
    end

/* Page down  */
    when response = "DOWN" then do
      if start_at + 14 <= lines then start_at = start_at + 14
      call BUILD_SCREEN
    end
    otherwise nop
  end
  "DISPLAY PANEL(CEDA)"
end
"LIBDEF ISPLLIB"

/* Restore the normal paging keys  */
zpf07 = hpf07
zpf08 = hpf08
zpf19 = hpf19
zpf20 = hpf20
"VPUT (ZPF07,ZPF08,ZPF19,ZPF20)"
exit

/* Build the results screen  */
BUILD_SCREEN:
j = 1
do i=start_at to start_at + 13
  if line.i = "" then leave i
  interpret "line"j" = line.i"
  j = j + 1
end
message = "Stage 1 rc = "sev1", stage 2 rc = "sev2
return
```

PANEL CEDA

```
)Attr Default(%+_)
% TYPE(TEXT  ) INTENS(HIGH) COLOR(YELLOW)
+ TYPE(TEXT  ) INTENS(LOW ) COLOR(TURQUOISE)
_ TYPE(INPUT) INTENS(HIGH) CAPS(ON ) JUST(LEFT ) COLOR(GREEN)
! type(input) intens(high) caps(on ) just(left ) pad('.') COLOR(GREEN)
# type(output) intens(low ) caps(off) just(asis ) COLOR(green)
@ type(output) intens(low ) caps(off) just(asis ) COLOR(red)
)Body  Expand(//)
%-/-/- CEDA -/-/-
%Command ===>_zcmd
%
+CICS system  : !cics    +

+CEDA Command : _cedacmd

+-/-/-
```

```
#line1
#line2
#line3
#line4
#line5
#line6
#line7
#line8
#line9
#line1Ø
#line11
#line12
#line13
#line14
+-/-/-
@message
)Init
)Proc
&response = &Z
if (.RESP = END)
   &response = 'END'
if (.RESP = ENTER)
   &response = 'ENTER'
if (.PFKEY = PFØ7)
   &response = 'UP'
if (.PFKEY = PF19)
   &response = 'UP'
if (.PFKEY = PFØ8)
   &response = 'DOWN'
if (.PFKEY = PF2Ø)
   &response = 'DOWN'
VPUT (RESPONSE) SHARED
)End
```

JCL CHECKER

```
/*————————————————————————————— REXX ——————————————————————————————— */
/*                                                                   */
/* Routine that will check JCL. Actually JES does the syntax         */
/* checking, we just make sure that datasets exist etc.              */
/* JCL to to be checked is read from DD JCL and a report is          */
/* produced at DD REPORT                                             */
/*                                                                   */
/* Sample JCL :-                                                     */
/*                                                                   */
/*  //JBSP27XX JOB .......................................           */
/*  //CICSPARM EXEC PGM=IKJEFTØ1,                                    */
/*  //         PARM='%JCLSCAN'                                       */
/*  //SYSEXEC  DD   DSN=<exec library>,DISP=SHR                      */
/*  //SYSTSPRT DD   SYSOUT=*                                         */
/*  //SYSTSIN  DD   DUMMY                                                 */
/*  //JCL      DD   DSN=<dataset>(member),DISP=SHR                        */
/*  //REPORT   DD   SYSOUT=*                                              */
/*                                                                        */

arg debug

if debug = "TRACE" then trace r

"PROFILE MSGID"
userid = USERID()

/*----------------------------------------------------------------------*/
/* Get the name of the dataset we're processing                          */
/*----------------------------------------------------------------------*/
rc = OUTTRAP("INFO.","*","CONCAT")
"LISTA STATUS"
rc = OUTTRAP("OFF")

do i=1 to info.Ø
  if POS("--DDNAME--DISP--",info.i) > Ø then iterate
  if POS("NULLFILE",info.i) > Ø then iterate
  if POS("TERMFILE",info.i) > Ø then iterate
  jcl_dsname = STRIP(info.i)
  j = i+1
  parse var info.j ddname .
  ddname = STRIP(ddname)
  if ddname = "JCL" then leave
  i = j
end

parse var jcl_dsname . "(" jcl_member ")" .

/*----------------------------------------------------------------------*/
/* Read the job                                                          */
/*----------------------------------------------------------------------*/
"EXECIO * DISKR JCL (STEM LINE. FINIS)"

/*----------------------------------------------------------------------*/
/* Find out if it's a JOB or an STC                                      */
/*----------------------------------------------------------------------*/

its_a_job = Ø
do i=1 to line.Ø
  if LEFT(line.i,3) = "//*" then iterate
  if POS(" JOB ",line.i) > Ø then its_a_job = 1
  leave
end

/*----------------------------------------------------------------------*/
/* If it's a job then modify its job card to make sure it's got         */
/* a TYPRUN=SCAN and MSGCLASS=X                                          */
/*-----------------------------------------------------------------------*/
if its_a_job then do i=1 to line.Ø
  select
    when LEFT(line.i,3) = "//*" then queue line.i
    when POS(" JOB ",line.i) > Ø then do
      parse var line.i "//" jobname ,
                       " JOB " . ,
                       "(" acct ")" ,
                       "'" pgmr "'" ,
                       rest .
      p = POS("MSGCLASS=",rest)
      if p > Ø then rest = OVERLAY("X",rest,p+9,1)
      if RIGHT(rest,1) ¬= "," then do
        line = "//"userid"XX JOB ("acct"),'"pgmr"'"rest","
        queue line
        line = "// TYPRUN=SCAN"
        queue line
      end
      else do
        line = "//"userid"XX JOB ("acct"),'"pgmr"'"rest
        queue line
        do j=i+1 to line.Ø
          parse var line.j "//" rest x .
          p = POS("MSGCLASS=",rest)
          if p > Ø then rest = OVERLAY("X",rest,p+9,1)
          if RIGHT(rest,1) ¬= "," then do
            line = "// "rest","
            queue line
            line = "// TYPRUN=SCAN"
            queue line
            i = j
            leave j
          end
          else do
            line = "// "rest
            queue line
          end
        end
      end
    end
    otherwise queue line.i
  end
end
/*-----------------------------------------------------------------------*/
/* If it's an STC then generate a job card and EXEC for it               */
/*-----------------------------------------------------------------------*/
else do
  queue "//"userid"XX JOB (ACCT#),'JCL CHECK',"
  queue "// CLASS=A,MSGCLASS=X,TYPRUN=SCAN"
  queue "//*"
  queue "//"jcl_member" EXEC "jcl_member
end
queue "GO"

/*-------------------------------------------------------------------*/
/* Submit the job and get its job-id                                 */
/*-------------------------------------------------------------------*/

x = OUTTRAP("LINE.","*","CONCAT")
address TSO "SUBMIT * END(GO)"
x = OUTTRAP("OFF")
do i = 1 to line.0
  if POS("IKJ56250I",line.i) > 0 then do
    parse var line.i . " JOB " jobid .
    leave i
  end
end

/*-------------------------------------------------------------------*/
/* Wait till the job finishes                                        */
/*-------------------------------------------------------------------*/
x = OUTTRAP("LINE.","*","CONCAT")
address TSO "STATUS "jobid
x = OUTTRAP("OFF")
parse var line.1 . (jobid) status
do while status ¬= "ON OUTPUT QUEUE"
  x = OUTTRAP("LINE.","*","CONCAT")
  address TSO "STATUS "jobid
  x = OUTTRAP("OFF")
  parse var line.1 . (jobid) status
end

/*-------------------------------------------------------------------*/
/* Get the job from the input queue                                  */
/*-------------------------------------------------------------------*/

address TSO "ALLOC F(LISTING) NEW SPACE(2 1) TRACKS"
x = LISTDSI("LISTING" "FILE")
x = OUTTRAP("LINE.","*","CONCAT")
address TSO "OUTPUT ("jobid") PRINT('"sysdsname"')"
x = OUTTRAP("OFF")
address TSO "EXECIO * DISKR LISTING (STEM LINE. FINIS)"
x = MSG("OFF")
address TSO "FREE F(LISTING)"
x = MSG("ON")

/*-------------------------------------------------------------------*/
/* See if we had a JCL error.                                        */
/*-------------------------------------------------------------------*/
```

```
jcl_err_stmt. = ""
jcl_errs = Ø
jcl_error = Ø
do i=1 to line.Ø
  if POS("IEFC452I",line.i) > Ø then do
    do j=i+1 to line.Ø
      if POS("STMT NO. MESSAGE",line.j) > Ø then do k=j+1 to line.Ø
        parse var line.k stmt_no msg_id msg
        if (msg_id = "IEFCØØ1I") | (msg_id = "IEFCØØ2I") then iterate
        jcl_errs = jcl_errs + 1
        jcl_err_stmt.jcl_errs = stmt_no
        j = k
      end
    end
    leave i
  end
end

/*———————————————————————————————————————————————————————————————————*/
/* Scan for datasets                                                 */
/*———————————————————————————————————————————————————————————————————*/
dsname. = ""
line_no. = ""
dsnames = Ø
do i=1 to line.Ø
  if ((POS("//",line.i) > Ø)   & ,
      (POS("DSN=",line.i) > Ø) & ,
      (POS("//*",line.i) = Ø)) | ,
     ((POS("XX",line.i) > Ø)   & ,
      (POS("DSN=",line.i) > Ø) & ,
      (POS("XX*",line.i) = Ø)) then do
    parse var line.i . "DSN=" dsname "," .
    parse var dsname dsname .
    if POS("&",dsname) = Ø then do
      dsnames = dsnames + 1
      dsname.dsnames = dsname
      line_no.dsnames = i
    end
    else do j=i+1 to line.Ø
      if POS("IEFC653I",line.j) > Ø then do
        parse var line.j . "DSN=" dsname "," .
        if dsname ¬= "" then do
          parse var dsname dsname .
          dsnames = dsnames + 1
          dsname.dsnames = dsname
          line_no.dsnames = j
          i = j
          leave j
        end
      end
    end
  end
end

/*-----------------------------------------------------------------------*/
/* Write the report                                                      */
/*-----------------------------------------------------------------------*/

line_count = 66
out = Ø
out_line. = ""
head_1 = "1CICS Team JCL Checker run on "DATE("E")" at "TIME("N")
head_2 = "ØChecking "jcl_dsname
head_3 = "Ø"CENTRE("ERROR MESSAGES",4Ø,"-")
head_3 = head_3" "CENTRE("JES LISTING",8Ø,"-")

no_jcl_yet = 1
jcl_to_do = Ø
final_return_code = Ø
do i=1 to line.Ø
  line = SUBSTR(line.i,2)
  if no_jcl_yet then do
    if POS("//",line) > Ø then do
      jcl_to_do = 1
      no_jcl_yet = Ø
    end
  end
  if POS("STMT NO. MESSAGE",line) > Ø then jcl_to_do = Ø
  msg = COPIES(" ",41)
  out_line = msg||line
  if jcl_to_do then do
    parse var line stmt_no .
    no_jcl_error = 1
    do j=1 to jcl_errs
      if jcl_err_stmt.j = stmt_no then do
        msg = "!E! JCL ERROR : SEE MESSAGES"
        out_line = OVERLAY(msg,out_line,2,4Ø)
        no_jcl_error = Ø
        final_return_code = 8
        leave j
      end
    end
    if no_jcl_error then do j=1 to dsnames
      if i = line_no.j then do
        msg = SYSDSN("'"dsname.j"'")
        if msg ¬= "OK" then do
          msg = "!E! "msg
          out_line = OVERLAY(msg,out_line,2,4Ø)
          final_return_code = 8
        end
```

```
          leave j
        end
      end
    end
    line_count = line_count + 1
    if line_count > 56 then do
      out = out + 1
      out_line.out = head_1
      out = out + 1
      out_line.out = head_2
      out = out + 1
      out_line.out = head_3
      out = out + 1
      out_line.out = " "
      line_count = 5
    end
    out = out + 1
    out_line.out = out_line
  end

  address TSO "EXECIO * DISKW REPORT (STEM OUT_LINE. FINIS)"
  exit final_return_code
```

Kevin Wailes
J Sainsbury (UK)

CEMT logger – an alternative design

In *A CEMT log for CICS 4.1, CICS Update*, October 1998, Russell Hunt described a system to record CEMT output to the CSMT transient data queue. This consisted of an XZCOUT user exit to capture TIOA output from CEMT commands in a GETMAINed work area, and a COBOL 'sweeper' program that periodically writes data from this work area to the CSMT destination.

I have found two problems with this design. Firstly, the user exit program consumes excessive resources, because:

- It executes an (XPI) GETMAIN/FREEMAIN for every invocation of the exit, including those invocations not related to the CEMT transaction.

- It formats the CEMT data before writing it to the buffer. This CPU-intensive processing is better (asynchronously) performed in the 'sweeper' program.

This excessive processing increases the pathlength for all VTAM-related terminal output.

The second problem is that the formatting algorithm for the CEMT data relies on hardcoded displacements of various fields in the output buffer. This makes the algorithm vulnerable to any PTF (or CICS release change) that affects the offset of CEMT output fields.

To overcome these problems, I have redesigned the system. The working storage for the user exit is now obtained once only on the first invocation of the 'sweeper' program (ZZZCEMT). The address of this storage is saved to the GWA of the user exit (ZZZZCOUT). The user exit cannot start processing until this pointer is set. Therefore there is only one GETMAIN per CICS run.

Also, the user exit does no formatting of the CEMT TIOA buffer, but just copies it 'as is' to the work buffer. The user-id is appended to this data as an additional refinement.

Formatting of the CEMT output is now performed by the 'sweeper' program. The formatting algorithm does not rely on hardcoded field offsets, but instead on the presence of the SBA character (X'11') at the start of each new line. This is used as the COBOL UNSTRING delimiter. This technique is unlikely to be affected by future CICS maintenance or release changes. Although COBOL UNSTRING is fairly CPU-intensive, the 'sweeper' transaction only runs every 'n' minutes as a low-priority non-terminal task. The output from the UNSTRING is not 100% neat and tidy, but is adequate for logging purposes.

The 'sweeper' transaction/program is kicked off at PLTPI time. The program then issues an ENABLE EXIT for the XZCOUT user exit, performs a GETMAIN for all working storage, and records the address of this storage in the exit's GWA. It then issues a (self)START after the agreed 'sweeper' interval and terminates. Every subsequent invocation checks the TIOA buffer for data to write to the CSMT TDQ.

Error conditions encountered by the user exit are recorded in the GWA and are written to the CSMT log by the ‘sweeper’ program. This is a better way to log user exit errors than by using WTO. WTOs always have the potential to flood the MVS console should a loop condition occur.

ZZZCEMT

```
 CBL XOPTS(SP)
 CBL TRUNC(OPT) DATA(31)
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    ZZZCEMT.
*******************************************************************
*
*FUNCTION: WRITE CEMT RECORDS COLLECTED BY ZZZZCOUT EXIT
*          TO THE 'CSMT' TD QUEUE ('SWEEPER' PROGRAM).
*
*CALLS/LINKS : (NONE)
*
*DESCRIPTION:
*THIS PROGRAM COPIES CEMT OUTPUT RECORDED BY AN XZCOUT GLUE
*PROGRAM TO THE 'CSMT' DCT DESTINATION. THE RECORDS INCLUDE THE
*USER-ID. THIS MAKES IN EFFECT A CEMT LOGGING FACILITY.
*
*THE PROGRAM STARTS AT PLT TIME, ENABLES THE ZZZZCOUT EXIT
*PROGRAM, AND TERMINATES. JUST BEFORE TERMINATION,
*THE PROGRAM ISSUES A START WITH A SUITABLE
*INTERVAL - THE MONITORING INTERVAL - TO RE-ACTIVATE ITSELF.
*
* NOTE - THIS PROGRAM AND ITS ASSOCIATED TRANSACTION MUST RUN IN
* CICS KEY SINCE IT MUST ACCESS THE USER EXIT GWA.
*
*******************************************************************

 ENVIRONMENT DIVISION.
 DATA DIVISION.
*************************
 WORKING-STORAGE SECTION.
*************************
 Ø1  C-CONSTANTS.
     Ø2 C-EYECATCHER     VALUE '*START OF WORKING STORAGE*'
                                                PIC X(26).
     Ø2 C-PROGRAM-ID     VALUE 'ZZZCEMT'        PIC X(8).
     Ø2 C-REQID          VALUE 'ZZZREQ2'        PIC X(8).
     Ø2 C-EXIT-PROGRAM   VALUE 'ZZZZCOUT'       PIC X(8).
     Ø2 C-EXIT-GWALEN    VALUE +91              PIC S9(4) COMP.
     Ø2 C-XZCOUT         VALUE 'XZCOUT'         PIC X(8).
```

```
    Ø2 C-VERSION        VALUE 'Ø1.ØØ'          PIC X(5).
    Ø2 C-MSG-QUEUE      VALUE 'CSMT'           PIC X(4).
    Ø2 C-MONITOR-TRANS-ID VALUE 'ZZZ2'         PIC X(4).
    Ø2 C-DEFAULT-INTERVAL-MINS    VALUE +2     PIC S9(8) COMP.
    Ø2 C-UNDERSCORE     VALUE '_'              PIC X.
    Ø2 C-SBA-CHAR           VALUE X'11'        PIC X.
    Ø2 C-START-FIELD-CHAR   VALUE X'1D'        PIC X.
    Ø2 C-INSERT-CURSOR      VALUE X'13'        PIC X.
    Ø2 C-WORKAREA-SIZE  VALUE +65536           PIC S9(8) COMP.

Ø1 W-SWITCHES.
    Ø2 W-START-CODE           VALUE SPACES     PIC X(2).
       88 SW-START-NO-DATA    VALUE 'S '.
       88 SW-START-WITH-DATA  VALUE 'SD'.
       88 SW-TERMINAL-TASK    VALUE 'TD'.
    Ø2 W-PROCESS-SWITCH       VALUE 'N'        PIC X.
       88 SW-NOT-DONE         VALUE 'N'.
       88 SW-ALL-DONE         VALUE 'Y'.

Ø1  W-MSG.
    Ø2 W-MSG-PROGRAM                           PIC X(7).
    Ø2 FILLER            VALUE '-'             PIC X.
    Ø2 W-MSG-NO                                PIC X(2).
    Ø2 FILLER                                  PIC X.
    Ø2 W-MSG-TIME                              PIC X(8).
    Ø2 FILLER                                  PIC X.
    Ø2 W-MSG-TEXT                              PIC X(6Ø).

Ø1  W-TIOA-DATA                                PIC X(32768).

Ø1  W-WORK-FIELDS.
    Ø2 W-EXIT-GWALEN                           PIC S9(4) COMP.
    Ø2 W-WORK-LENGTH                           PIC S9(4) COMP.
    Ø2 W-UNSTRING-COUNT                        PIC S9(4) COMP.
    Ø2 W-WORKAREA-OFFSET                       PIC S9(8) COMP.
    Ø2 W-TDQ-BUFFER                            PIC X(8Ø).
    Ø2 W-REQID                                 PIC X(8).
    Ø2 W-PIC5                                  PIC 9(5).
    Ø2 W-GETMAIN-LENGTH                        PIC S9(8) COMP.
    Ø2 W-INTERVAL-BIN-MINS                     PIC S9(8) COMP.
    Ø2 W-INTERVAL-PIC-MINS                     PIC 9(2).
    Ø2 W-WHEN-COMPILED.
       Ø3 W-COMPILE-DATE                       PIC X(8).
       Ø3 W-COMPILE-TIME                       PIC X(8).
    Ø2 W-ABSTIME                               PIC S9(15) COMP.
    Ø2 W-RESP                                  PIC S9(8) COMP.
    Ø2 W-RESP2                                 PIC S9(8) COMP.
    Ø2 W-RESP-PIC                              PIC 99999.
    Ø2 W-RESP2-PIC                             PIC 99999.
    Ø2 W-WORK-PTR                              USAGE IS POINTER.
    Ø2 W-WORK-PTR-BIN REDEFINES W-WORK-PTR     PIC S9(8) COMP.
    Ø2 W-WORK-PTR-SAVE                         USAGE IS POINTER.
    Ø2 I                                       PIC S9(8) COMP.
    Ø2 J                                       PIC S9(8) COMP.
    Ø2 K                                       PIC S9(8) COMP.
    Ø2 W-CICS-STATUS                           PIC S9(8) COMP.
    Ø2 W-DDMMYYYY                              PIC X(1Ø).
    Ø2 W-TIME                                  PIC X(8).

 Ø1 W-CEMT-OUTPUT-TABLE.
    Ø2 W-CEMT-OUTPUT-LINE       OCCURS 1ØØ.
       Ø3 W-SBA-ORDERS                         PIC X(2).
       Ø3 W-CEMT-OUTPUT-DATA                   PIC X(79).

*****************************************************************
 LINKAGE SECTION.
*****************************************************************
 Ø1 DFHCOMMAREA                                PIC X(4Ø96).

* GWA DESCRIPTION MUST MATCH DSECT IN ZZZZCOUT
 Ø1 L-GWA.
   Ø2 L-GWA-DOUBLE-WORD                        PIC S9(15) COMP.
   Ø2 L-GWA-WORKAREA1-PTR                      USAGE IS POINTER.
   Ø2 L-GWA-WORKAREA2-PTR                      USAGE IS POINTER.
   Ø2 L-GWA-OFFSET                             PIC S9(8) COMP.
   Ø2 L-GWA-TIOALEN                            PIC S9(8) COMP.
   Ø2 L-GWA-TERMID                             PIC X(4).
   Ø2 L-GWA-TRANID                             PIC X(4).
   Ø2 L-GWA-USERID                             PIC X(8).
   Ø2 L-GWA-ERROR-NO                           PIC X.
      88 SW-NO-PROBLEMS          VALUE X'ØØ'.
      88 SW-NO-WORKAREA          VALUE X'Ø1'.
      88 SW-XPI-CALL-1-FAILED    VALUE X'Ø2'.
      88 SW-XPI-CALL-2-FAILED    VALUE X'Ø3'.
      88 SW-BUFFER-FULL          VALUE X'Ø4'.
   Ø2 L-GWA-ERROR-MESSAGE                      PIC X(49).
   Ø2 L-GWA-STATUS-FLAG                        PIC X.
      88 SW-WORKAREA1-ACTIVE     VALUE '1'.
      88 SW-WORKAREA2-ACTIVE     VALUE '2'.

 Ø1 L-WORKAREA.
   Ø2 L-WORKAREA-USERID                        PIC X(8).
   Ø2 L-WORKAREA-TERMID                        PIC X(4).
   Ø2 L-WORKAREA-TIOALEN                       PIC S9(5) COMP-3.
   Ø2 L-WORKAREA-TIOALEN-ALT REDEFINES L-WORKAREA-TIOALEN
                                               PIC X(3).
   Ø2 L-WORKAREA-TIOA-DATA                     PIC X(65526).

*****************************************************************
 PROCEDURE DIVISION.
```

```
******************************************************************

*******************
 ØØØØ-MAIN SECTION.
*******************
     PERFORM P-INITIALIZE.
     PERFORM P-PROCESS.
     PERFORM P-START-AGAIN.

 ØØØØ-CICS-RETURN.
     EXEC CICS RETURN NOHANDLE END-EXEC.
     GOBACK.

******************************************************************
 P-INITIALIZE.
******************************************************************
     MOVE SPACES        TO W-MSG.
     MOVE C-PROGRAM-ID  TO W-MSG-PROGRAM.
     MOVE WHEN-COMPILED TO W-WHEN-COMPILED.
     MOVE SPACES        TO W-TDQ-BUFFER.
     MOVE C-DEFAULT-INTERVAL-MINS TO W-INTERVAL-BIN-MINS.
     PERFORM P-GET-TIMESTAMP.
     PERFORM P-CHECK-CICS-STATUS.
     PERFORM P-CHECK-START-MODE.

******************************************************************
 P-CHECK-CICS-STATUS.
******************************************************************
     EXEC CICS INQUIRE SYSTEM
          CICSSTATUS(W-CICS-STATUS)
          RESP(W-RESP)
          RESP2(W-RESP2)
     END-EXEC.

     EVALUATE TRUE
        WHEN W-RESP NOT = DFHRESP(NORMAL)
           MOVE 'INQUIRE SYSTEM ERROR' TO W-MSG-TEXT
           MOVE 'Ø1' TO W-MSG-NO
           PERFORM P-HANDLE-ERROR
           GO TO ØØØØ-CICS-RETURN
        WHEN W-CICS-STATUS = DFHVALUE(ACTIVE)
           CONTINUE
        WHEN W-CICS-STATUS = DFHVALUE(STARTUP)
           PERFORM P-PLTPI-PROCESSING
           GO TO ØØØØ-CICS-RETURN
        WHEN OTHER
           MOVE 'CICS IS SHUTTING DOWN' TO W-MSG-TEXT
           MOVE 'Ø2' TO W-MSG-NO
           PERFORM P-HANDLE-ERROR
           GO TO ØØØØ-CICS-RETURN
     END-EVALUATE.

*****************************************************************
 P-PLTPI-PROCESSING.
*****************************************************************
     PERFORM P-GET-TIMESTAMP.

     MOVE WHEN-COMPILED TO W-WHEN-COMPILED.
     MOVE C-PROGRAM-ID  TO W-MSG-PROGRAM.
     MOVE 'ØØ'          TO W-MSG-NO.

     STRING
        'VERSION ' C-VERSION
        ' COMPILED ' W-COMPILE-DATE ' ' W-COMPILE-TIME
        DELIMITED BY SIZE
        INTO W-MSG-TEXT
     END-STRING.

     PERFORM P-WRITE-MSG.
     PERFORM P-ENABLE-EXIT-PROGRAM.
     PERFORM P-INITIALIZE-EXIT-WORKAREAS.
     PERFORM P-START-AGAIN.

*****************************************************************
 P-INITIALIZE-EXIT-WORKAREAS.
*****************************************************************
     PERFORM P-EXTRACT-EXIT.

     IF W-RESP NOT = DFHRESP(NORMAL) THEN
        MOVE 'Ø3' TO W-MSG-NO
        MOVE 'EXTRACT EXIT FAILED' TO W-MSG-TEXT
        PERFORM P-HANDLE-ERROR
        GO TO ØØØØ-CICS-RETURN
     END-IF.

     MOVE +Ø TO L-GWA-OFFSET.
     MOVE C-WORKAREA-SIZE TO W-GETMAIN-LENGTH.
     PERFORM P-GETMAIN.
     SET L-GWA-WORKAREA1-PTR TO W-WORK-PTR.
     SET ADDRESS OF L-WORKAREA TO W-WORK-PTR.
     MOVE LOW-VALUES TO L-WORKAREA.

     MOVE C-WORKAREA-SIZE TO W-GETMAIN-LENGTH.
     PERFORM P-GETMAIN.
     SET L-GWA-WORKAREA2-PTR TO W-WORK-PTR.
     SET ADDRESS OF L-WORKAREA TO W-WORK-PTR.
     MOVE LOW-VALUES TO L-WORKAREA.
     SET SW-WORKAREA1-ACTIVE TO TRUE.

*****************************************************************
```

```
 P-EXTRACT-EXIT.
*******************************************************************
     EXEC CICS EXTRACT EXIT
          PROGRAM(C-EXIT-PROGRAM)
          GALENGTH(W-EXIT-GWALEN)
          GASET(ADDRESS OF L-GWA)
          RESP(W-RESP)
          RESP2(W-RESP2)
     END-EXEC.

*******************************************************************
 P-GETMAIN.
*******************************************************************
     EXEC CICS GETMAIN
          FLENGTH(W-GETMAIN-LENGTH)
          SET(W-WORK-PTR)
          SHARED
          RESP(W-RESP)
          RESP2(W-RESP2)
     END-EXEC.

     IF W-RESP NOT = DFHRESP(NORMAL) THEN
        MOVE 'Ø4' TO W-MSG-NO
        MOVE 'GETMAIN FAILED' TO W-MSG-TEXT
        PERFORM P-HANDLE-ERROR
        GO TO ØØØØ-CICS-RETURN
     END-IF.

*******************************************************************
 P-ENABLE-EXIT-PROGRAM.
*******************************************************************
     EXEC CICS ENABLE
          PROGRAM(C-EXIT-PROGRAM)
          ENTRYNAME(C-EXIT-PROGRAM)
          EXIT(C-XZCOUT)
          GALENGTH(C-EXIT-GWALEN)
          START
          RESP(W-RESP)
          RESP2(W-RESP2)
     END-EXEC.

     IF W-RESP NOT = DFHRESP(NORMAL) THEN
        MOVE 'Ø5' TO W-MSG-NO
        MOVE 'ENABLE EXIT FAILED' TO W-MSG-TEXT
        PERFORM P-HANDLE-ERROR
        GO TO ØØØØ-CICS-RETURN
     END-IF.

*******************************************************************
 P-CHECK-START-MODE.
```

```
******************************************************************
     EXEC CICS ASSIGN
          STARTCODE(W-START-CODE)
          NOHANDLE
     END-EXEC.

     EVALUATE TRUE
        WHEN SW-START-NO-DATA
           CONTINUE
******* IF TERMINAL TASK, RESTART MONITORING IF IT HAS STOPPED
        WHEN SW-TERMINAL-TASK
           PERFORM P-INQUIRE-REQID
           IF EIBRESP NOT = DFHRESP(NORMAL) THEN
              PERFORM P-EXTRACT-EXIT
              IF EIBRESP NOT = DFHRESP(NORMAL) THEN
                 PERFORM P-ENABLE-EXIT-PROGRAM
                 PERFORM P-INITIALIZE-EXIT-WORKAREAS
              END-IF
              PERFORM P-START-AGAIN
              MOVE 'Ø6' TO W-MSG-NO
              MOVE 'MONITORING RESTARTED' TO W-MSG-TEXT
              PERFORM P-SEND-MSG
              GO TO ØØØØ-CICS-RETURN
           END-IF
        WHEN OTHER
           MOVE 'Ø7' TO W-MSG-NO
           MOVE 'INVALID INVOCATION MODE' TO W-MSG-TEXT
           PERFORM P-HANDLE-ERROR
           GO TO ØØØØ-CICS-RETURN
     END-EVALUATE.

******************************************************************
 P-SEND-MSG.
******************************************************************
     IF EIBTRMID(1:1) > SPACE THEN
        EXEC CICS SEND CONTROL
            ERASE
             NOHANDLE
        END-EXEC

        EXEC CICS SEND
             FROM(W-MSG)
             LENGTH(LENGTH OF W-MSG)
             NOHANDLE
        END-EXEC
     END-IF.
     MOVE SPACES TO W-MSG-TEXT.

******************************************************************
 P-PROCESS.
```

```
*****************************************************************
     PERFORM P-EXTRACT-EXIT.

     IF W-RESP NOT = DFHRESP(NORMAL) THEN
        MOVE 'Ø8' TO W-MSG-NO
        MOVE 'EXTRACT EXIT FAILED' TO W-MSG-TEXT
        PERFORM P-HANDLE-ERROR
        GO TO ØØØØ-CICS-RETURN
     END-IF.

     EVALUATE TRUE
        WHEN SW-WORKAREA1-ACTIVE
           SET W-WORK-PTR TO L-GWA-WORKAREA1-PTR
           SET SW-WORKAREA2-ACTIVE TO TRUE
        WHEN SW-WORKAREA2-ACTIVE
           SET W-WORK-PTR TO L-GWA-WORKAREA2-PTR
           SET SW-WORKAREA1-ACTIVE TO TRUE
        WHEN OTHER
           MOVE 'NO ACTIVE WORKAREA' TO W-MSG-TEXT
           MOVE 'Ø9' TO W-MSG-NO
           PERFORM P-HANDLE-ERROR
           GO TO ØØØØ-CICS-RETURN
     END-EVALUATE.

     MOVE +Ø TO L-GWA-OFFSET.
     SET ADDRESS OF L-WORKAREA TO W-WORK-PTR.
     SET W-WORK-PTR-SAVE       TO W-WORK-PTR.

    PERFORM P-CHECK-FOR-EXIT-ERROR.
**** EXIT IF NOTHING TO COPY
     IF L-WORKAREA-USERID = LOW-VALUES THEN
        PERFORM P-START-AGAIN
        GO TO ØØØØ-CICS-RETURN
     END-IF.

     SET SW-NOT-DONE TO TRUE.
     MOVE +Ø TO W-WORKAREA-OFFSET.
     PERFORM P-WRITE-CEMT-DATA UNTIL SW-ALL-DONE.
     SET ADDRESS OF L-WORKAREA TO W-WORK-PTR-SAVE.
     MOVE LOW-VALUES TO L-WORKAREA.

*****************************************************************
 P-CHECK-FOR-EXIT-ERROR.
*****************************************************************
     IF NOT SW-NO-PROBLEMS OR
        L-GWA-ERROR-MESSAGE NOT = LOW-VALUES THEN
        MOVE SPACES TO W-TDQ-BUFFER
        STRING
           'ZZZCEMT-99 ERROR IN ZZZZCOUT: '  DELIMITED BY SIZE
           L-GWA-ERROR-MESSAGE               DELIMITED BY '  '
           INTO W-TDQ-BUFFER
        END-STRING
        PERFORM P-WRITE-TDQ
        MOVE LOW-VALUES TO L-GWA-ERROR-MESSAGE
        SET SW-NO-PROBLEMS TO TRUE
     END-IF.

*******************************************************************
 P-WRITE-CEMT-DATA.
*******************************************************************
     IF L-WORKAREA-USERID = LOW-VALUES OR
        L-WORKAREA-TIOALEN-ALT = LOW-VALUES OR
        L-WORKAREA-TIOALEN IS NOT NUMERIC THEN
        SET SW-ALL-DONE TO TRUE
     ELSE
        MOVE L-WORKAREA-TIOALEN TO W-WORK-LENGTH
        STRING
           '** CEMT LOG '
           'USER='    L-WORKAREA-USERID ' '
           'TERMID='  L-WORKAREA-TERMID
           DELIMITED BY SIZE INTO W-TDQ-BUFFER
        END-STRING
        PERFORM P-WRITE-TDQ
        IF W-WORK-LENGTH < +1 THEN
           SET SW-ALL-DONE TO TRUE
        ELSE
           COMPUTE W-WORKAREA-OFFSET = W-WORKAREA-OFFSET   +
                             LENGTH OF L-WORKAREA-TIOALEN +
                             LENGTH OF L-WORKAREA-USERID  +
                             LENGTH OF L-WORKAREA-TERMID  +
                             W-WORK-LENGTH
           IF W-WORKAREA-OFFSET > C-WORKAREA-SIZE THEN
              SET SW-ALL-DONE TO TRUE
           ELSE
              PERFORM P-PARSE-CEMT-OUTPUT
              ADD W-WORKAREA-OFFSET TO W-WORK-PTR-BIN
              SET ADDRESS OF L-WORKAREA TO W-WORK-PTR
           END-IF
        END-IF
     END-IF.
```

Editor's note: this article will be concluded next month.

David Roth
CICS Consultant (Germany)

Dealing with program abends – part 2

This month we conclude the program that gets control whenever CICS detects a program abend and displays the transaction ID under which the program was running, along with the abending program name.

DPWIZM – WIZARD MAIL INTERFACE PROGRAM

DPWIZM sends WIZMAIL messages to Wizard Mail using DPL, and was written to eliminate the 'ATNI' abend caused by DFHPEP. This abend occurred when the code below (formerly contained in DFHPEP) was executed, because DFHPEP executes under an abend state.

```
GBLC  &ABOV
         GBLC  &SYID
&ABOV    SETC  'Y'                SET TO 'N' IF YOU WANT 24-BIT USAGE.
&SYID    SETC  'DEVM'             SET TO SYSID OF WHERE WIZMAIL LIVES.
```

You should assemble and LNKEDT this program as you would any other Assembler CICS command-level program. There are two generation options, specified in the '&ABOV SETC 'Y'' and the '&SYID SETC ' '' shown above. You may wish to change these before assembly/LNKEDT:

- If you don't wish to run this program above the line (ie in 31-bit mode) or if you aren't using high-level Assembler (ie ASMA90), specify '&ABOV SETC N'.
- If you're running in an MRO/ISC environment and Wizard Mail is running in the TOR, you must specify the four-character SYSID of the CICS system where Wizard Mail is running by entering it in the '&SYID SETC ????' statement. Otherwise, set this option to blank. Note that the value specified in this option is not checked. If an incorrect SYSID is given, an error will occur at program execution time.

Note that:

- The assembly and LNKEDT of this program should end with a $RC of zero. If the $RC value is different, you should examine the assembly listing, determine the problem, fix it, and reassemble the program. Note that if you specify any value other than ‘Y’ or ‘N’ in the ‘&ABOV SETC ‘?’’ statement, you will receive a $RC value of four.

- Before you assemble/LNKEDT this program, you must assemble and catalogue the DPEIBC subroutine (see below), which is called by this program. The subroutine must be catalogued as an .OBJ MEMBERTYPE. The LIB.SUBLIB into which the subroutine is catalogued must be LIBDEFed when you assemble and catalogue this program.
- It doesn’t matter which LIB.SUBLIB this program is catalogued into, although it should be one of your own and not an IBM one. The LIB.SUBLIB into which you catalogue the program must be LIBDEFed in your CICS start-up JCL.
- This program requires no special PCT or PPT operands. An RDO sample of the PCT/PPT operands is shown below (note that you should replace the question marks with values of your choice):

```
PCT DEFINITION:
TRANSACTION    : DPWI
GROUP          : ????????
PROGRAM      ==> DPWIZM
TWASIZE      ==> ØØØØØ
PROFILE      ==> DFHCICST
PARTITIONSET ==>
STATUS       ==> ENABLED
PRIMEDSIZE   ==> ØØØØØ
REMOTE ATTRIBUTES
DYNAMIC      ==> NO
REMOTESYSTEM ==>
REMOTENAME   ==>
TRPROF       ==>
LOCALQ       ==>
SCHEDULING
PRIORITY     ==> ???
TCLASS       ==> NO
ALIASES
ALIAS        ==>
TASKREQ      ==>
XTRANID      ==>
RECOVERY
DTIMOUT      ==> NO
INDOUBT      ==> BACKOUT
RESTART      ==> NO
```

```
    SPURGE       ==> ??? (RECOMMEND YES)
    TPURGE       ==> ??? (RECOMMEND YES)
    DUMP         ==> YES
    TRACE        ==> YES
    SECURITY
    EXTSEC       ==> NO
    TRANSEC      ==> Ø1
    RSL          ==> ??? (RECOMMEND PUBLIC)
    RSLC         ==> NO

    PPT DEFINITION:
    PROGRAM        : DPWIZM
    GROUP          : ??
    LANGUAGE     ==> ASSEMBLER
    RELOAD       ==> NO
    RESIDENT     ==> NO
    RSL          ==> ??? (RECOMMEND PUBLIC)
    STATUS       ==> ENABLED
    REMOTE ATTRIBUTES
    REMOTESYSTEM ==>
    REMOTENAME   ==>
    TRANSID      ==>
    EXECUTIONSET ==> FULLAPI
```

DPWIZM

```
DPWI     TITLE 'DPWIZM - 1.Ø - WIZARD MAIL INTERFACE PROGRAM.'
         AIF   ('&ABOV' EQ 'Y').ABOVØØ
         AIF   ('&ABOV' EQ 'N').ABOVØØ
         MNOTE 4,'VALUE FOR ''ABOV'' NOT ''Y'' OR ''N'', FORCED TO ''Y'X
               '.'
&ABOV    SETC  'Y'
.ABOVØØ  ANOP
DFHEISTG DSECT
RESP     DS    F                    RESPONSE CODE. (FROM EXEC CICS COMMA
RETLEN   DS    H                    RETRIEVE LENGTH.
APPLID   DS    CL8                  APPLID.
STRTCD   DS    CL2                  START CODE.
*
         DFHREGS ,                  USE CICS REGISTER EQUATES.
*
         USING WIZCOMM,R9
WIZCOMM  DSECT                 WIZARD MAIL DSECT.
         DS   ØCL15ØØ
WIZERRM  DS   CL6Ø                  WIZMAIL MESSAGE HEADER.
WIZSUBJ  DS   CL25                  WIZMAIL MESSAGE SUBJECT.
WIZCONF  DS   C                     WIZMAIL CONFIRMATION CODE.
WIZDIS1  DS   CL7Ø                  WIZMAIL DISTRIBUTION LINE ONE (1).
WIZDIS2  DS   CL7Ø                  WIZMAIL DISTRIBUTION LINE TWO (2).
WIZDIS3  DS   CL7Ø                  WIZMAIL DISTRIBUTION LINE THREE (3).
```

```
WIZDIS4  DS   CL7Ø                 WIZMAIL DISTRIBUTION LINE FOUR (4).
WIZMSG1  DS   CL7Ø                 WIZMAIL MESSAGE LINE ONE (1).
WIZMSG2  DS   CL7Ø                 WIZMAIL MESSAGE LINE TWO (2).
WIZMSG3  DS   CL7Ø                 WIZMAIL MESSAGE LINE THREE (3).
WIZMSG4  DS   CL7Ø                 WIZMAIL MESSAGE LINE FOUR (4).
WIZMSG5  DS   CL7Ø                 WIZMAIL MESSAGE LINE FIVE (5).
WIZMSG6  DS   CL7Ø                 WIZMAIL MESSAGE LINE SIX (6).
WIZMSG7  DS   CL7Ø                 WIZMAIL MESSAGE LINE SEVEN (7).
WIZMSG8  DS   CL7Ø                 WIZMAIL MESSAGE LINE EIGHT (8).
WIZMSG9  DS   CL7Ø                 WIZMAIL MESSAGE LINE NINE (9).
WIZMSGA  DS   CL7Ø                 WIZMAIL MESSAGE LINE TEN (1Ø).
WIZMSGB  DS   CL7Ø                 WIZMAIL MESSAGE LINE ELEVEN (11).
WIZMSGC  DS   CL7Ø                 WIZMAIL MESSAGE LINE TWELVE (12).
WIZMSGD  DS   CL7Ø                 WIZMAIL MESSAGE LINE THIRTEEN (13).
WIZMSGE  DS   CL7Ø                 WIZMAIL MESSAGE LINE FOURTEEN (14).
WIZMSGF  DS   CL7Ø                 WIZMAIL MESSAGE LINE FIFTEEN (15).
WIZOPID  DS   CL8                  WIZMAIL ORIGINATING OPERATOR ID.
         DS   CL76                 NOT USED.
*
DPWIZM   DFHEIENT CODEREG=(RA),DATAREG=(R6),EIBREG=(R4)
         AIF   ('&ABOV' NE 'Y').ABOVØ1
DPWIZM   AMODE 31
DPWIZM   RMODE ANY
.ABOVØ1  ANOP
         MVC   WRTAREA,WRTAREA-1   CLEAR WRTAREA.
         MVC   RETLEN,=H'15ØØ'     SET RETRIEVE LENGTH.
         EXEC  CICS ASSIGN STARTCODE(STRTCD) APPLID(APPLID)            X
               RESP(RESP).         GET SOME INFORMATION.
         CLC   RESP,DFHRESP(NORMAL) WAS THERE AN ERROR.
         BE    START               NO-BRANCH TO START.
         MVC   WRTAREA(3Ø),=C'DPWIZM-ASSIGN ERROR.           '
         BAL   RB,ERROR3           PERFORM ERROR3 ROUTINE.
         B     RETURN              BRANCH TO RETURN.
*
START    EQU   *
         CLC   =C'S ',STRTCD       WERE WE STARTED WITHOUT DATA.
         BNE   START3              NO-BRANCH TO START3.
         MVC   WRTAREA(3Ø),=C'DPWIZM-START W/NO DATA ERROR '
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         B     RETURN              BRANCH TO RETURN.
*
START3   EQU   *
         CLC   =C'SD',STRTCD       WERE WE STARTED WITH DATA.
         BNE   LINKED              NO-BRANCH TO LINKED.
         EXEC  CICS RETRIEVE SET(R9) LENGTH(RETLEN).
         CLC   RESP,DFHRESP(NORMAL) WAS THERE AN ERROR.
         BE    RETRVE              NO-BRANCH TO RETRVE.
         MVC   WRTAREA(3Ø),=C'DPWIZM-RETRIEVE ERROR.         '
         BAL   RB,ERROR3           PERFORM ERROR3 ROUTINE.
         B     RETURN              BRANCH TO RETURN.
```

```
*
RETRVE   EQU   *
         CLC   =H'15ØØ',RETLEN     WAS RETRIEVE LENGTH 15ØØ.
         BE    RETRVE2             YES-BRANCH TO RETRVE2.
         MVC   WRTAREA(3Ø),=C'DPWIZM-RETRIEVE LENGTH ERROR. '
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         B     RETURN              BRANCH TO RETURN.
*
RETRVE2  EQU   *
*        CLC   =C'CICSPRD2',APPLID ARE WE RUNNING IN CICSPRD2.
*        BNE   RETRVE3             NO-BRANCH TO RETRVE3.
*        EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)         X
*              LENGTH(RETLEN) RESP(RESP). GO SEND MESSAGE.
*        B     RETRVE7             BRANCH TO RETRVE7.
*
*ETRVE3  EQU   *
         CLC   =C'    ',SYSID      IS SYSID BLANK.
         BE    RETRVE4             YES-BRANCH TO RETRVE4.
         EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)         X
               LENGTH(RETLEN) SYSID(SYSID) DATALENGTH(RETLEN)         X
               RESP(RESP).         GO SEND MESSAGE.
         B     RETRVE7             BRANCH TO RETRVE7.
*
RETRVE4  EQU   *
         EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)         X
               LENGTH(RETLEN)                DATALENGTH(RETLEN)       X
               RESP(RESP).         GO SEND MESSAGE.
*
RETRVE7  EQU   *
         CLC   RESP,DFHRESP(NORMAL) ANY ERROR.
         BE    RETRVE9             NO-BRANCH TO RETRVE9.
         BAL   RB,ERROR            PERFORM ERROR ROUTINE.
         B     RETURN              BRANCH TO RETURN.
*
RETRVE9  EQU   *
         CLC   WIZERRM(35),=C'                                   ' WAS
         BE    RETURN              NO-BRANCH TO RETURN.
         MVC   WRTAREA(25),=C'DPWIZM-WIZMAIL SEND ERROR' MVE ERROR MESS
         ST    RB,SVRB             SVE CONTENTS OF REG 11.
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         MVC   WRTAREA(L'WIZERRM),WIZERRM MVE WIZMAIL ERROR MESSAGE TO
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         L     RB,SVRB             RESTORE CONTENTS OF REG 11.
         B     RETURN              BRANCH TO RETURN.
*
LINKED   EQU   *
         CLC   RETLEN,EIBCALEN     WAS EIBCALEN LENGTH 15ØØ.
         BNE   RETURN              NO-BRANCH TO RETURN.
         EXEC  CICS ADDRESS COMMAREA(R9).
*        CLC   =C'CICSPRD2',APPLID ARE WE RUNNING IN CICSPRD2.
*        BNE   LINKED3             NO-BRANCH TO LINKED3.
*        EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)          X
*              LENGTH(RETLEN) RESP(RESP). GO SEND MESSAGE.
*        B     LINKED7             BRANCH TO LINKED7.
*
*INKED3  EQU   *
         CLC   =C'    ',SYSID      IS SYSID BLANK.
         BE    LINKED4             YES-BRANCH TO LINKED4.
         EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)          X
               LENGTH(RETLEN) SYSID(SYSID) DATALENGTH(RETLEN)          X
               RESP(RESP).         GO SEND MESSAGE.
         B     LINKED7             BRANCH TO LINKED7.
*
LINKED4  EQU   *
         EXEC  CICS LINK PROGRAM('WMPAIM1') COMMAREA(WIZCOMM)          X
               LENGTH(RETLEN)                DATALENGTH(RETLEN)        X
               RESP(RESP).         GO SEND MESSAGE.
*
LINKED7  EQU   *
         CLC   RESP,DFHRESP(NORMAL) ANY ERROR.
         BE    LINKED9             NO-BRANCH TO LINKED9.
         BAL   RB,ERROR            PERFORM ERROR ROUTINE.
         B     RETURN              BRANCH TO RETURN.
*
LINKED9  EQU   *
         CLC   WIZERRM(35),=C'                                  ' WAS
         BE    RETURN              NO-BRANCH TO RETURN.
         ST    RB,SVRB             SVE CONTENTS OF REG 11.
         MVC   WRTAREA(25),=C'DPWIZM-WIZMAIL SEND ERROR' MVE ERROR MESS
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         MVC   WRTAREA(L'WIZERRM),WIZERRM MVE WIZMAIL ERROR MESSAGE TO
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         L     RB,SVRB             RESTORE CONTENTS OF REG 11.
*
RETURN   EQU   *
         EXEC  CICS RETURN.        RETURN TO CICS.
*
ERROR    EQU   *              ERROR ROUTINE.
         MVC   WRTAREA(37),=C'DPWIZM-LINK ERROR TO WMPAIM1 OCCURRED'
*
ERROR3   EQU   *              ERROR ROUTINE.
         MVC   EIBFN1(8),EIBFN     MVE EIBFN/EIBRCODE TO EIBFN1.
         MVC   EIBFN2(8),EIBRESP   MVE EIBRESP/EIBRESP2 TO EIBFN2.
         LA    RD,SAVEAREA         LOAD ADDRESS OF SAVEAREA TO REG 13.
         CALL  DPEIBC,(EIBFN1,EIBOUT1,EIBFN2,EIBOUT2)
         ST    RB,SVRB             SVE CONTENTS OF REG 11.
         BAL   RB,WTOC             PERFORM WTOC ROUTINE.
         MVC   WRTAREA(15),=C'EIBFN/EIBRCODE='
         MVC   WRTAREA+15(L'EIBOUT1),EIBOUT1
         BAL   RB,WTOC              PERFORM WTOC ROUTINE.
         MVC   WRTAREA(17),=C'EIBRESP/EIBRESP2='
         MVC   WRTAREA+17(L'EIBOUT2),EIBOUT2
         BAL   RB,WTOC              PERFORM WTOC ROUTINE.
         L     RB,SVRB              RESTORE CONTENTS OF REG 11.
         BR    RB                   RETURN TO CALLER.
*
WTOC     EQU   *               WRITE TO CONSOLE ROUTINE.
         EXEC  CICS WRITE OPERATOR TEXT(WRTAREA)                      X
               TEXTLENGTH(L'WRTAREA). WRITE MESSAGE TO SYSTEM CONSOLE.
         MVC   WRTAREA,WRTAREA-1    CLEAR WRTAREA.
         BR    RB                   RETURN TO CALLER.
*
         DC    C' '                 DON'T MOVE/REMOVE THIS STATEMENT.
WRTAREA  DC    CL65' '              WRITE AREA.
*
SYSID    DC    CL4'&SYID'           SYSID SVE AREA WHERE WIZARD MAIL LIV
*
EIBFN1   DS    XL8                  INPUT FIELD FOR CALL TO DPEIBC.
EIBOUT1  DS    CL17                 OUTPUT FIELD FOR CALL TO DPEIBC.
EIBFN2   DS    XL8                  INPUT FIELD FOR CALL TO DPEIBC.
EIBOUT2  DS    CL17                 OUTPUT FIELD FOR CALL TO DPEIBC.
*
         DS    ØD
SVRB     DS    F                    SVE AREA FOR REG 11.
SAVEAREA DS    18F                  SVE AREA FOR CALL TO DPEIBC.
*
         LTORG
*
         END   DPWIZM
```

DPEIBC

The DPEIBC subroutine converts the EIBFN and EIBRCODE in the EXEC interface block from hexadecimal to EBCDIC for CICS programs.

Two parameters must be passed, consisting of one field each, while a third and fourth parameter, each consisting of two fields, are optional. The first parameter must always be EIBFN.

The second parameter must be seventeen bytes, and is where the conversion of the EIBFN and EIBRCODE will be placed. The layout of the output will be as follows:

Field	Position	Value	Length
EIBFN	Ø1-Ø4		Ø4
	Ø5-Ø5	-	Ø1
EIBRCODE	Ø6-17		12

The optional third and fourth parameters consist of two fields each, as follows:

- The third parameter must contain two 4-byte hex fields consisting of the response (ie EIBRESP) field and the response two (ie EIBRESP2) field.
- The fourth parameter must contain two 8-byte fields plus an additional byte, for a total length of seventeen bytes, and is where the result of the conversion of the EIBRESP and EIBRESP2 fields will be placed. A hyphen is placed between each field.

Note that:

- The contents of the fields you move into the first parameter don't matter, as long as they're eight hex bytes long. This means that you could move two 4-byte hex fields. In other words, you can use this subroutine to convert any two 4-byte hex fields. But remember that the hyphen will then be placed incorrectly.
- The contents of the fields you move into the third parameter don't matter either, as long as they're also eight hex bytes long.
- The correct number of parameters in this subroutine is either two or four. If an incorrect number of parameters are passed, the first four bytes of the first parameter are returned with X'FFFFFFFF'.

Calling sequences

The calling sequences follow.

For COBOL:

```
-  CALL 'DPEIBC' USING EIBFN, OUTPUT.
```

or

```
-  CALL 'DPEIBC' USING EIBFN, OUTPUT, PARAM3, PARAM4.
```

For ALC:

```
-   LA    13,SAVEAREA (13 CAN ALSO BE R13 OR RD).
    CALL  DPEIBC,(EIBFN,OUTPUT)
```

or

```
    CALL  DPEIBC,(EIBFN,OUTPUT,PARAM3,PARAM4)
```

```
               .
               .  (MAINLINE PART OF PROGRAM).
               .
SAVEAREA DC    18F'Ø'
```

For RPGII:

```
     -   CALL  'DPEIBC'
               PARM          EIBFN
               PARM          OUTPUT
```

or

```
               PARM          PARAM3
               PARM          PARAM4
```

An eighteen-word save area must be passed through register 13 by the user (STD COBOL LINKAGE).

DPEIBC

```
DPEI     TITLE 'DPEIBC - 1.Ø - CONVERTS EIBFN AND EIBRCODE CODES FROM HX
               EX TO EBCDIC.'
*
DPEIBC   CSECT Ø
DPEIBC   AMODE 31
DPEIBC   RMODE ANY
         ENTRY DPEIBC
         BALR  15,Ø                 LOAD TEMPORARY BASE.
         USING *,15                 INFORM ASSEMBLER.
         SAVE  (14,12)
         DROP  15                   DROP TEMPORARY BASE.
         BALR  8,Ø                  LOAD BASE REG.
         USING *,8                  INFORM ASSEMBLER.
         ST    13,SAVEAREA+4        STORE CALLERS RETURN ADDRESS.
         LA    13,SAVEAREA          LOAD CALLERS REGISTERS.
         B     EIBBEG               BRANCH TO EIBBEG.
*
         DC    C'DPEIBC STARTS HERE. ' INSERT EYE CATCHER.
*
EIBBEG   EQU   *
         LM    3,4,Ø(1)             LOAD ADDRESS OF EIB CODES AND OUTPUT
         STM   3,4,SVR3R4           SVE REGS 3 AND 4.
         MVI   NUMPRM,X'ØØ'         SET NUMBER OF PARAMETERS TO ZERO.
         SR    6,6                  SET PARAMETER COUNT TO ZERO.
*
EIBARG   EQU   *
         TM    Ø(1),X'8Ø'           ARE WE DONE.
         BO    EIBLST               YES-BRANCH TO EIBLST.
         LA    6,4(6)              INCREMENT REG 6 BY ONE (1).
         LA    1,4(1)              INCREMENT REG 1 TO NEXT PARAMETER.
         B     EIBARG              BRANCH TO EIBARG.
*
EIBLST   EQU   *
         SR    1,6                 RESTORE REG 1.
         SRL   6,2                 DIVIDE REG 6 BY 2.
         LA    6,1(6)              BUMP BY ONE FOR FIRST TIME.
         STC   6,NUMPRM            SAVE NUMBER OF PARAMETERS PASSED.
         CLI   NUMPRM,X'Ø2'        WAS ONE (1) PARAMETER PASSED.
         BL    EIBERR              YES-BRANCH TO EIBERR. (ERROR).
         CLI   NUMPRM,X'Ø4'        WAS MORE THAN FOUR (4) PARAMETERS PA
         BH    EIBERR              YES-BRANCH TO EIBERR. (ERROR).
         CLI   NUMPRM,X'Ø2'        WERE TWO (2) PARAMETERS PASSED.
         BE    EIBNXT              YES-BRANCH TO EIBNXT.
         LM    5,6,8(1)            LOAD ADDRESSES OF THIRD AND FOURTH P
         STM   5,6,SVR5R6          SVE REG 5 AND 6.
         LA    9,8                 LOAD COUNT OF BYTES TO CHANGE.
         SLR   1,1                 CLEAR REG 1.
         SLR   2,2                 CLEAR REG 2.
*
EIBLOP1  EQU   *
         IC    1,Ø(Ø,5)            LOAD BYTE FROM RESP FIELD.
         LR    2,1                 PUT IT IN REG 2 ALSO.
         SRL   1,4                 MVE 1ST 4 BITS TO LAST 4 BITS.
         IC    1,TRAN(1)           GET CHARACTER FOR HEX DIGIT.
         STC   1,Ø(Ø,6)            STORE IN OUTPUT ERROR CODES.
         LA    6,1(6)              INCREMENT OUTPUT ADDRESS BY ONE (1).
         N     2,=XL4'ØF'          GET RIGHT NIBBLE ALONE.
         IC    2,TRAN(2)           GET CHARACTER FOR HEX DIGIT.
         STC   2,Ø(Ø,6)            STORE IN OUTPUT ERROR CODES.
         LA    6,1(6)              INCREMENT OUTPUT ADDRESS BY ONE (1).
         LA    5,1(5)              INCREMENT EIB CODES ADDRESS BY ONE (
         C     9,=F'5'             ARE WE AT THE END OF RESP FIELD.
         BNE   EIBEXT              NO-BRANCH TO EIBEXT.
         MVI   Ø(6),C'-'           MVE DASH (-).
         LA    6,1(6)              INCREMENT OUTPUT ADDRESS BY ONE (1).
*
EIBEXT   EQU   *
         BCT   9,EIBLOP1           BRANCH TO EIBLOP1 UNTIL REG 9 ZERO.
*
EIBNXT   EQU   *
         LA    9,8                 LOAD COUNT OF BYTES TO CHANGE.
         SLR   1,1                 CLEAR REG 1.
         SLR   2,2                 CLEAR REG 2.
*
EIBLOP3  EQU   *
         IC    1,Ø(Ø,3)            LOAD BYTE FROM EIB CODES.
         LR    2,1                 PUT IT IN REG 2 ALSO.
         SRL   1,4                 MVE 1ST 4 BITS TO LAST 4 BITS.
         IC    1,TRAN(1)           GET CHARACTER FOR HEX DIGIT.
         STC   1,Ø(Ø,4)            STORE IN OUTPUT ERROR CODES.
         LA    4,1(4)              INCREMENT OUTPUT ADDRESS BY ONE (1).
         N     2,=XL4'ØF'          GET RIGHT NIBBLE ALONE.
         IC    2,TRAN(2)           GET CHARACTER FOR HEX DIGIT.
         STC   2,Ø(Ø,4)            STORE IN OUTPUT ERROR CODES.
         LA    4,1(4)              INCREMENT OUTPUT ADDRESS BY ONE (1).
         LA    3,1(3)              INCREMENT EIB CODES ADDRESS BY ONE (
         C     9,=F'7'             ARE WE AT THE END OF EIBFN.
         BNE   EIBEXT3             NO-BRANCH TO EIBEXT3.
         MVI   Ø(4),C'-'           MVE DASH (-).
         LA    4,1(4)              INCREMENT OUTPUT ADDRESS BY ONE (1).
*
EIBEXT3  EQU   *
         BCT   9,EIBLOP3           BRANCH TO EIBLOP3 UNTIL REG 9 ZERO.
*
EIBRTN   EQU   *
         SR    15,15               CLEAR REG 15.
         L     13,SAVEAREA+4       LOAD RETURN ADDRESS TO REG 13.
         RETURN (14,12),RC=(15)    RETURN TO CALLER.
*
EIBERR   EQU   *
         MVC   Ø(4,3),=X'FFFFFFFF' INDICATE PARAMETER ERROR.
         B     EIBRTN              BRANCH TO EIBRTN.
*
         DC    C'DPEIBC STORAGE HERE. ' INSERT EYE CATCHER.
*
NUMPRM   DS    X
TRAN     DC    C'Ø123456789ABCDEF'
SVR3R4   DS    2F
SVR5R6   DS    2F
SAVEAREA DC    18F'Ø'
*
         END
```

Robert Botsis
Senior Systems Programmer (USA)

Why not share your expertise and earn money at the same time? *CICS Update* is looking for JCL, macros, program code, etc, that experienced CICS users have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

CICS news

Iona Technologies has announced Orbix for CICS, enabling integration of CICS-based mainframe applications with the rest of the enterprise using standards-based CORBA technology.

Orbix for CICS runs inside the CICS transaction monitor and provides full application integration facilities with enterprise server systems on the mainframe, Unix, Windows, and Java platforms. It enables CICS applications to act as both CORBA servers and clients, enabling CICS to participate as a true Enterprise Application Server peer. Features include full ORB support within the CICS environment; COBOL, PL/I, and C++ language bindings; and native support for CICS Transaction Services 1.3.

For further information contact:
Iona Technologies, 60 Aberdeen Avenue, Cambridge, MA 02138, USA.
Tel: (617) 949 9000.
URL: http://www.iona.com.

* * *

Sterling Software has announced a new CICS/VSE Interface extension to its VM:Webgateway Web-to-host software, providing Web access to VSE data and applications through a GUI. VSE developers can carry out Web enhancements using CICS/VSE COBOL and other CICS command-level programming languages.

With the new interface, application developers can write CGI scripts using either CICS/VSE COBOL or REXX. The product includes a VSE tutorial demonstrating how to create CGI scripts and how to Web-enable VSE applications. CGI scripts from the sample application can be copied and applied to real VSE applications to bring them to the Web.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB.
Tel: (0181) 867 8000.
URL: http://www.sterling.com.

* * *

CICS users can benefit from Tivoli's e-business management software for OS/390, allowing use of System/390 as the management server with service level improvement and business process view capabilities.

New products include Tivoli Manager for OS/390, Tivoli Service Desk for OS/390 Version 1.2, enhancements to Tivoli NetView for OS/390 and Tivoli Global Enterprise Manager, and an OS/390 version of Tivoli Enterprise. New direct support for OS/390 applications includes CICS, VTAM, and MQ, and an application toolkit for custom-built applications.

For further information contact:
Tivoli Systems, 9442 Capitol of Texas Highway North, Arboretum Plaza One, Austin, TX 78759, USA.
Tel: (512) 436 8000.
URL: http://www.tivoli.com.

xephon