151 CICS

June 1998

In this issue

update

CICS Update

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
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; you will need the user-id shown on your address label.

Editor
Robert Burgess

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.

Subscriptions and back-issues
A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £170.00 in the UK; $260.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £14.50 ($22.00) each including postage.

Printed in England.

CICS message log browser

We have written a CICS message log browser that works as a CICS transaction, giving us the ability to browse and search the different CICS log destinations on-line.

In the past, if we wanted to call CMAC (the IBM-supplied CICS messages and codes transaction) to get information about a DFH message or an abend code, we had to:

1 Leave the log browser.

2 Call CMAC.

3 Remember which message to enter, and view the explanation.

4 Exit CMAC.

5 Restart the log browser.

6 Search the point in the log that we came from.

This was not very user-friendly, and so I thought about having a direct link to CMAC from our log browser, with the possibility of passing the DFH message or abend code I was looking for.

Checking the CICS manuals for a 'user entry point' to DFHCMAC was unsuccessful. Some attempts with EXEC CICS START, LINK, and XCTL failed, so I asked IBM Support whether CMAC can be called from user programs. Their answer was "no way" – which gave me a challenge to make it work anyway!

Without boring you with testing details, I discovered that DFHCMAC requires a COMMAREA of 29,274 bytes, with some bits set. I did not find out about every byte in the COMMAREA, but I found out enough to make it possible to call DFHCMAC.

COMMAREA LAYOUT

The layout of the COMMAREA is as follows. Bytes 1 - 72 must be

blank, except for:

- Bytes 1 - 6, which must be either:
  - DFH message (component-id and message number, eg AC2206).
  - Right justified abend code with leading blanks.
  - All blank to display the DFHCMC01 input panel.
- Byte 7 – which must be C'Y'.

Bytes 73 to the end must be X'00', except for byte 76 where X'01', on the first call, makes DFHCMAC believe that you have come from the DFHCMC01 input panel.

This byte is also set by DFHCMAC on return to the calling transaction. The settings are:

- X'01' if returned from the DFHCMC01 input panel.
- X'02' if returning from the message explanation panel.

Because DFHCMAC is designed as a non-conversational program, it will issue EXEC CICS RETURN TRANSID with the calling transaction. Therefore, it is important to check byte 76 of the COMMAREA and the EIBAID field to decide whether to go back to DFHCMAC or to resume the calling transaction. Check the sample program CALLCMAC to see how it works.

Note: when it is first called, DFHCMAC expects the ENTER key to have been pressed. If called by a different function key, DFHCMAC will come up with the input panel DFHCMC01 and the message:

```
DFHME0501 AN INVALID OPTION HAS BEEN ENTERED.
```

You cannot remove this unless you have come from ENTER, or you make CICS believe that the ENTER key was hit, even if a different key was used.

SAMPLE PROGRAM CALLCMAC

```
CALLCMAC CSECT
CALLCMAC AMODE 31
```

```
CALLCMAC RMODE ANY
*
         DFHREGS                          REGISTER EQUATES
*
* CHECK IF WE ARE CALLED WITH COMMAREA
*
         XR    R6,R6
         EXEC  CICS ADDRESS COMMAREA(R6) RESP(CRESP)
*
         CLC   CRESP,DFHRESP(NORMAL)     ADDRESS OK?
         BNE   CALLCMAC_ALLOCATE           NO, ALLOCATE NEW
         C     R6,=X'FF000000'           COMMAREA GIVEN?
         BE    CALLCMAC_ALLOCATE           NO, ALLOCATE NEW
         LTR   R6,R6                     ALL ZERO
         BZ    CALLCMAC_ALLOCATE           YES, ALLOCATE NEW
         CLC   EIBCALEN,=H'29274'        RIGHT LENGTH?
         BNE   CALLCMAC_ALLOCATE           NO, ALLOCATE NEW
*
* NOW WE CAN BE SURE TO COME FROM DFHCMAC WITH A VALID COMMAREA
*
         CLI   EIBAID,DFHPF3             EXIT PRESSED?
         BNE   CALLCMAC_XCTL               NO, CONTINUE WITH DFHCMAC
*
*        CLI   75(R6),X'02'              EXIT FROM EXPLANATION MAP?
         CLI   75(R6),X'01'              EXIT FROM SELECTION MAP?
         BNE   CALLCMAC_XCTL               NO, CONTINUE WITH DFHCMAC
*
         B     CALLCMAC_EXIT             EXIT PROGRAM
*
* ALLOCATE STORAGE FOR COMMAREA, ALL HEX ZERO
*
CALLCMAC_ALLOCATE DS 0H
*
         EXEC  CICS GETMAIN LENGTH(29274) INITIMG(X'00') RESP(CRESP)   *
               SET(R6)
*
         CLC   CRESP,DFHRESP(NORMAL)     ALLOCATE OKAY?
         BNE   CALLCMAC_EXIT                NO, EXIT
*
* FILL FIRST 72 BYTES WITH BLANKS
*
         MVC   0(72,R6),BLANK            FIRST 72 BYTES BLANK
*
         MVI   6(R6),C'Y'                WHATEVER THIS IS FOR
*
*  MOVE MESSAGEID, ABEND CODE OR LEAVE BLANK
*
*        MVC   0(6,R6),=CL6'      '      BLANK FOR INPUT MAP
*        MVC   0(6,R6),=CL6'  ASRA'      OR RIGHT JUST. ABEND CODE
         MVC   0(6,R6),=CL6'AC2206'      OR COMPONENT/MESSAGE NUMBER
```

```
*
         MVI   75(R6),X'Ø1'              REQUIRED
         B     CALLCMAC_XCTL             CALL DFHCMAC


*
* TRANSFER CONTROL TO DFHCMAC
*
CALLCMAC_XCTL DS ØH
         EXEC  CICS XCTL PROGRAM('DFHCMAC')                              *
                COMMAREA(Ø(R6)) LENGTH(29274) RESP(CRESP)
*
* EXIT PROGRAM
*
CALLCMAC_EXIT DS ØH
         EXEC  CICS SEND CONTROL ERASE FREEKB RESP(CRESP)
         EXEC  CICS RETURN
         EJECT
*
         COPY  DFHAID
*
BLANK    DC    1ØØCL1' '
         LTORG
*
DFHEISTG DSECT
         DS    ØF
CRESP    DS    F
         END
```

The sample program is assembled with HLASM Release 2.0. Use your standard procedure for CICS Assembler programs. Link-edit parameters are AMODE 31 RMODE ANY.

Testing was done with CICS/ESA Version 4.1. For other CICS versions, it is probable that the COMMAREA structure will need to be reviewed.

To get back from samples to production usage, I changed our CICS log browser to do the following work:

- To call DFHCMAC:

```
if in log browse mode
  if enter key was pressed
       if cursor position is within log data area
          extract word under cursor
            if DFHxxxxxxx use as message
            all others take as abend codes
          save current log position
```

```
            build commarea for DFHCMAC
            XCTL to DFHCMAC
```

- On return from DFHCMAC:

```
if commarea passed
   if commarea length is 29274 bytes
      if EIBAID is PF3 and DFHCMC01 input panel was on screen
      (X'Ø1' at offset 75) restore old log position
      resume log browse
   otherwise return to DFHCMAC
```

This integrates the easy handling of the CICS message and abend code explanation into our log-browse transaction, saving keystrokes, time, and the need to remember abend codes or message numbers while jumping between transactions.

Stefan Raabe
Systems Programmer (Germany)

Date testing CICS applications – part 2

This month we complete the code for IY2K, which enables each user in the CICS region to establish his own private *'future' date.*

```
CHCKNEXT DS    ØH
         LA    R9,SHLINEL(9)       ADDRESS NEXT ROW
         BCTR  R8,RØ               ONE MORE PROCESSED
         B     CHCKLOOP
         DROP  R9
         DROP  R4
         DROP  R3
         EJECT
*-------------------------------------
* END AS A NORMAL PROGRAM
*-------------------------------------
         SPACE
RETURN   DS    ØH                      RETURN TO THE CALLER
         L     R13,UEPEPSA             ADDRESS OF EXIT SAVE AREA
         RETURN (14,12),RC=UERCNORM    RESTORE REGS AND RETURN
         SPACE
         END   IPPCPCFT
```

IPPCEIDI is the exit disabling program which stops both exits, and frees the piece of shared storage.

```
         TITLE 'IPPCEIDI - DISABLE XEIOUT AND XPCFTCH - IY2K'
         SPACE 2
IPPCEIDI AMODE 31
IPPCEIDI RMODE ANY
         SPACE 2
*-------------------------------------
*
* INVOKED BY THE EIDI TRANSACTION
*
* DISABLE BOTH EXIT PROGRAMS (IPPCEIOU AND IPPCPCFT)
*
*  - DISABLE IPPCEIOU EXIT STOP
*  - EXTRACT GLOBAL AREA ADDRESS
*  - FREE SHARED STORAGE
*  - DISABLE EXIT EXITALL
*
*-------------------------------------
         SPACE
         DFHREGS
         SPACE
RGA      EQU   9
RW       EQU   10
         EJECT
*--------------------------------------
*        CICS WORKING STORAGE - DYNAMIC USER STORAGE
*--------------------------------------
DFHEISTG DSECT
GALEN    DS    H
         SPACE 2
GASTORD  DSECT              (MAPPING OF CONT LEVEL DATA)
GAEYEC   DS    CL4
GASHAR   DS    CL4
         SPACE 2
YES      EQU   X'FF'
NO       EQU   X'00'
         SPACE 2
*--------------------------------------
*        CICS CODING
*--------------------------------------
         SPACE 2
IPPCEIDI DFHEIENT CODEREG=(R11),EIBREG=(R12),DATAREG=(R13)
*
         B     START
         DC    CL9'IPPCEIDI'
         DC    CL9'&SYSDATE'
         DC    CL9'&SYSTIME'
```

```
START    DS    ØH
*
         EXEC CICS ADDRESS EIB(R12)
*
         MVC   GALEN,=H'8'  LENGTH OF GLOBAL EXIT AREA
*
* STOP (DISABLE) THE IPPCEIOU EXIT PROGRAM (NOT YET DELETED)
*
         EXEC CICS DISABLE PROGRAM('IPPCEIOU') EXIT('XEIOUT')          C
               NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    DISAPCFT
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG1) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
* STOP (DISABLE) THE IPPCPCFT EXIT PROGRAM (NOT YET DELETED)
*
DISAPCFT DS    ØH
         EXEC CICS DISABLE PROGRAM('IPPCPCFT') EXIT('XPCFTCH')         C
               NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    EXTRGWA
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG2) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
* NOW OBTAIN THE GWA ADDRESS BECAUSE ...
*
EXTRGWA  DS    ØH
         EXEC CICS EXTRACT EXIT PROGRAM('IPPCEIOU')                    C
               GASET(RGA) GALENGTH(GALEN)                              C
               NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    FREESTOR
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG3) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
FREESTOR DS    ØH
         USING GASTORD,RGA
         L     RW,GASHAR
*
* WE MUST FREE THE PIECE OF SHARED STORAGE ...
*
```

```
         EXEC CICS FREEMAIN DATAPOINTER(RW) NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    STOPEIOU
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG4) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
* NOW WE CAN DELETE THE IPPCEIOU EXIT PROGRAM
*
STOPEIOU DS    ØH
         EXEC CICS DISABLE PROGRAM('IPPCEIOU') EXITALL                 C
               NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    STOPPCFT
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG5) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
STOPPCFT DS    ØH
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG6) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
*
* AND THE IPPCPCFT EXIT PROGRAM IS ALSO DELETED
*
         EXEC CICS DISABLE PROGRAM('IPPCPCFT') EXITALL                 C
               NOHANDLE
*
         CLC   EIBRESP,DFHRESP(NORMAL) RESPONSE NORMAL?
         BE    STOPPCOK
*
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG7) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
*
STOPPCOK DS    ØH
         EXEC CICS WRITE OPERATOR TEXT(ERRMSG8) TEXTLENGTH(OPMSGLEN)   C
               NOHANDLE
         B     RETURN
         EJECT
ERRMSG1  DC    CL6Ø'IPPCEIDI - IPPCEIOU - DISABLE STOP FAILED'
ERRMSG2  DC    CL6Ø'IPPCEIDI - IPPCPCFT - DISABLE STOP FAILED'
ERRMSG3  DC    CL6Ø'IPPCEIDI - IPPCEIOU - EXTRACT EXIT FAILED'
ERRMSG4  DC    CL6Ø'IPPCEIDI - IPPCEIOU - FREEMAIN FAILED'
ERRMSG5  DC    CL6Ø'IPPCEIDI - IPPCEIOU - DISABLE EXITALL FAILED'
ERRMSG6  DC    CL6Ø'IPPCEIDI - IPPCEIOU - EXIT DISABLED'
ERRMSG7  DC    CL6Ø'IPPCEIDI - IPPCPCFT - DISABLE EXITALL FAILED'
```

```
ERRMSG8  DC    CL6Ø'IPPCEIDI - IPPCPCFT - EXIT DISABLED'
OPMSGLEN DC    F'6Ø'               WRITE OPERATOR MESSAGE LENGTH
         EJECT
*-------------------------------------
*        RETURN TO CALLER ...
*-------------------------------------
RETURN   DS    ØH
         END   IPPCEIDI
```

IPPCIY2K

Now CICS is ready to start changing dates, as requested by the user. The IY2K transaction, invoking the IPPCIY2K program, allows users to enter the date with which they want their transactions to run. Our IY2K solution is based on a terminal identifier, but other key indicators (eg user-id, transaction, etc) can be used with little change to the code. Using terminals allowed us to use Year 2000 testing on inbound LU6.2 transactions.

The IY2K transaction checks the input parameters and, when PF4 is pressed, the shared storage is accessed and updated with the requested date. When a date has been set, PF6 is provided to reset the terminal to the 'normal' system date. PF6 also clears the entry in the shared storage table.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. IPPCIY2K.
      *-------------------------------------
      *
      * INVOKED BY IY2K TRANSACTION
      *
      * MAP : IPPIY2K
      *
      * THIS PROGRAM IS THE USER INTERFACE TO THE SHARED STORAGE PIECE
      * ALLOWING THE USER TO SET A PRIVATE DATE FOR A GIVEN TERMINAL
      * - PF3 = QUIT PROGRAM (IDEM CLEAR PFKEY)
      * - PF4 = UPDATE THE SHARED STORAGE TABLE WITH THE PRIVATE DATE
      * - PF6 = RESET (CLEAR) THE SHARED STORAGE TABLE FOR TERMINAL
      *
      *-------------------------------------
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *-------------------------------------
      * GLOBAL AREA STUFF
      *-------------------------------------
       Ø1  GA-LEN                        PIC S9(4) COMP VALUE Ø.
```

```
 Ø1  GA-PTR                          USAGE IS POINTER.
 Ø1  GA-PTR-NUM REDEFINES GA-PTR  PIC S9(8) COMP.
*-------------------------------------
* DATE WORK FIELDS
*-------------------------------------
 Ø1  JULDATE                         PIC S9(7) COMP-3.
 Ø1  JULDAYS                         PIC S9(3) COMP-3.
 Ø1  TOTDAYS                         PIC S9(8) COMP-3.
 Ø1  DAYCOUNT                        PIC S9(8) COMP.
 Ø1  Y2KFDAYN                        PIC S9(2) COMP-3.
 Ø1  Y2KFMONN                        PIC S9(2) COMP-3.
 Ø1  Y2KFYEAN                        PIC S9(4) COMP-3.
 Ø1  Y2KFYEAR                        PIC S9(4) COMP-3.
 Ø1  ABSTIME                         PIC S9(15) COMP-3 VALUE Ø.
 Ø1  Y2KFABSN                        PIC S9(15) VALUE Ø
          SIGN LEADING SEPARATE.
 Ø1  Y2KFABSX REDEFINES Y2KFABSN  PIC X(16).
 Ø1  MONTHVAL PIC X(24) VALUE '312931Ø3131Ø3131Ø313Ø31'.
 Ø1  MONTHTAB REDEFINES MONTHVAL.
     Ø3 MONTH OCCURS 12 PIC 99.
*-------------------------------------
* GENERAL WORK FIELDS
*-------------------------------------
 Ø1  CNT                             PIC S9(4) COMP VALUE Ø.
 Ø1  CNTS                            PIC S9(4) COMP VALUE Ø.
 Ø1  CMD-RESP                        PIC S9(8) COMP VALUE Ø.
 Ø1  CMD-RESP2                       PIC S9(8) COMP VALUE Ø.
 Ø1  IY2K-WORKAREA.
     Ø3  TX-QUIT-TO-CICS.
         Ø5  FILLER                  PIC X(158) VALUE SPACES.
         Ø5  TEXT-WORK               PIC X(32) VALUE
             'IY2K PROGRAM ENDED.'.
*-------------------------------------
* OUR MAP LAYOUT
*-------------------------------------
 COPY IPPIY2K.
 COPY DFHAID.
 COPY DFHBMSCA.
*-------------------------------------
* OUR COMMAREA
*-------------------------------------
 Ø1  COMMAREA.
     Ø3  FILLER                      PIC X(8) VALUE 'IPPCIY2K'.
     Ø3  FILLER                      PIC X(8) VALUE 'COMMAREA'.
     Ø3  OLD-FTER                    PIC X(4) VALUE SPACES.
*-------------------------------------
* LINKAGE SECTION
*-------------------------------------
 LINKAGE SECTION.
 Ø1  DFHCOMMAREA PIC X(8Ø).
*-------------------------------------
```

```
* LAYOUT OF GLOBAL WORK AREA IN COBOL
*-------------------------------------
 Ø1  GASTOR.
     Ø3 GA-EYEC   PIC X(4).
     Ø3 SH-PTR   USAGE IS POINTER.
     Ø3 SH-PTR-NUM REDEFINES SH-PTR PIC S9(8) COMP.
*-------------------------------------
* LAYOUT OF SHARED STORAGE AREA IN COBOL
*-------------------------------------
 Ø1  SHSTOR.
     Ø3 SH-EYEC   PIC X(4).
     Ø3 SH-CNT    PIC S9(4) COMP.
     Ø3 SH-ROW OCCURS 5ØØ.
        Ø5 SH-FTER  PIC X(4).
        Ø5 SH-FDAT  PIC S9(7) COMP-3.
        Ø5 SH-FABS  PIC S9(15) COMP-3.
        Ø5 SH-FDAY  PIC 99.
        Ø5 SH-FMON  PIC 99.
        Ø5 SH-FYEA  PIC 9999.
*
 PROCEDURE DIVISION.
*
     MOVE SPACES TO Y2KMESSO
*
*    EIBCALEN = Ø IS AT FIRST INVOCATION
*
     IF EIBCALEN = Ø
      PERFORM STARTIT
     ELSE
      MOVE DFHCOMMAREA TO COMMAREA
      PERFORM REC-MAP
     END-IF

     PERFORM RET-TO-CICS

     .
*-------------------------------------
* ANALYZE KEY PRESSED
*-------------------------------------
*
 REC-MAP.
*
     EXEC CICS RECEIVE MAP('IPPIY2K') MAPSET('IPPIY2K')
          RESP(CMD-RESP)
     END-EXEC
*
     IF CMD-RESP = DFHRESP(NORMAL) OR
        CMD-RESP = DFHRESP(MAPFAIL)
      EVALUATE EIBAID
       WHEN DFHENTER
        PERFORM PROC-ENTER
```

```
        WHEN DFHPF6
         PERFORM PROC-RESET
        WHEN DFHPF4
         PERFORM PROC-UPDAT
        WHEN DFHCLEAR
         PERFORM ENDIT
        WHEN DFHPF3
         PERFORM ENDIT
        WHEN OTHER
         PERFORM PROC-OTHER
       END-EVALUATE
      ELSE
       PERFORM ENDIT
      END-IF
      .
*-------------------------------
* NOT SUPPORTED KEY PRESSED
*-------------------------------
*
 PROC-OTHER.
      MOVE 'YOU HIT A BAD KEY DUMMY' TO Y2KMESSO
      .
*-------------------------------
* PF6 = RESET ENTRY IN TABLE
*-------------------------------
*
 PROC-RESET.
*
      MOVE -1 TO Y2KFDAYL
      PERFORM SCAN-SHSTOR
      IF CNTS = Ø
       MOVE 'ENTRY NOT FOUND IN SHARED STORAGE' TO Y2KMESSO
      ELSE
       PERFORM VARYING CNT FROM CNTS BY 1 UNTIL CNT > SH-CNT
        MOVE SH-ROW(CNT + 1) TO SH-ROW(CNT)
       END-PERFORM
       SUBTRACT 1 FROM SH-CNT
       MOVE 'YEAR / DATE RESET ' TO Y2KMESSO
*
       EXEC CICS ASKTIME ABSTIME(ABSTIME)
       END-EXEC
*
       EXEC CICS FORMATTIME ABSTIME(ABSTIME)
                 TIME(Y2KTIMEO) TIMESEP(':')
                 DDMMYYYY(Y2KDATEO) DATESEP('/')
       END-EXEC
*
       MOVE Y2KDATEO(1:2) TO Y2KFDAYO
       MOVE Y2KDATEO(4:2) TO Y2KFMONO
       MOVE Y2KDATEO(7:4) TO Y2KFYEAO
       MOVE SPACES TO Y2KFJULO
```

```
      MOVE SPACES TO Y2KFABSO
*
     END-IF
     .
*-------------------------------
* PF4 = UPDATE ENTRY IN TABLE
*-------------------------------
*
 PROC-UPDAT.
     PERFORM PROC-CHECKINP
     IF Y2KMESSO = SPACES
      MOVE Ø TO JULDAYS
      PERFORM VARYING CNT FROM 1 BY 1 UNTIL CNT > Y2KFMONN
       IF CNT NOT = Y2KFMONN
        ADD MONTH(CNT) TO JULDAYS
       END-IF
      END-PERFORM
      ADD Y2KFDAYN TO JULDAYS
      IF Y2KFYEAN > 1999
       MOVE 1ØØØØØ TO JULDATE
      ELSE
       MOVE Ø TO JULDATE
      END-IF
      DIVIDE Y2KFYEAN BY 1ØØ GIVING CNTS REMAINDER Y2KFYEAR
*
* JULDATE IS THE PRIVATE DATE IN EIBDATE FORMAT
*
      COMPUTE JULDATE = JULDATE + Y2KFYEAR * 1ØØØ + JULDAYS
*
* THE ABSTIME DIFFERENCE IS COMPUTED - BASED ON THE DIFFERENCE IN
* DAYS BETWEEN THE "SYSTEM" DATE AND THE "PRIVATE" DATE REQUESTED
*
      SUBTRACT 19ØØ FROM Y2KFYEAN GIVING Y2KFYEAR
      DIVIDE Y2KFYEAR BY 4 GIVING CNT REMAINDER CNTS
      COMPUTE TOTDAYS = CNT * (365 * 3 + 366) + CNTS * 365
      IF CNTS = Ø
       SUBTRACT 1 FROM TOTDAYS
      END-IF
      ADD JULDAYS TO TOTDAYS
*
* NOW TOTDAYS CONTAINS THE NUMBER OF DAYS SINCE Ø1Ø119ØØ
* NOW WE REQUEST THE DAYCOUNT OF THE "SYSTEM" DATE
*
      EXEC CICS ASKTIME ABSTIME(ABSTIME)
      END-EXEC
      EXEC CICS FORMATTIME ABSTIME(ABSTIME)
                DAYCOUNT(DAYCOUNT)
      END-EXEC
      SUBTRACT DAYCOUNT FROM TOTDAYS
*
* NOW TOTDAYS IS THE DIFFERENCE IN DAYS BETWEEN "SYSTEM" AND
```

```
* "PRIVATE" DATE
*
      COMPUTE ABSTIME = TOTDAYS * 86400 * 1000
*
* ABSTIME IS THE DIFFERENCE IN "ABSTIME" UNITS
*
      IF ABSTIME = 0
       MOVE 'PLEASE DO NOT SPECIFY THE CURRENT DATE' TO Y2KMESSO
      ELSE
       MOVE JULDATE TO Y2KFJULO
       MOVE ABSTIME TO Y2KFABSN
       MOVE Y2KFABSX TO Y2KFABSO
*
       PERFORM SCAN-SHSTOR
       IF CNTS = 0
        ADD 1 TO SH-CNT
        MOVE SH-CNT TO CNTS
        IF CNTS > 500
         MOVE 'SHARED TABLE FULL' TO TEXT-WORK
         PERFORM DISPERR
        END-IF
       END-IF
       MOVE Y2KFDAYN TO SH-FDAY(CNTS)
       MOVE Y2KFMONN TO SH-FMON(CNTS)
       MOVE Y2KFYEAN TO SH-FYEA(CNTS)
       MOVE Y2KFTERO TO SH-FTER(CNTS)
       MOVE JULDATE  TO SH-FDAT(CNTS)
       MOVE ABSTIME  TO SH-FABS(CNTS)
       MOVE 'F6-RESET' TO Y2KPF06O
       MOVE 'FUTURE DATE IS SET' TO Y2KMESSO
      END-IF
     END-IF
     .
*-------------------------------------
* ENTER KEY ONLY CHECKS INPUT DATA ENTERED
*  IF TERMINAL WAS CHANGED, DATA IS RETRIEVED FROM SHARED STOR.
*-------------------------------------
*
 PROC-ENTER.
     IF Y2KFTERI NOT = OLD-FTER
      PERFORM SCAN-SHSTOR
      IF CNTS > 0
       MOVE SH-FDAY(CNTS) TO Y2KFDAYO
       MOVE SH-FMON(CNTS) TO Y2KFMONO
       MOVE SH-FYEA(CNTS) TO Y2KFYEAO
       MOVE SH-FDAT(CNTS) TO Y2KFJULO
       MOVE SH-FABS(CNTS) TO Y2KFABSN
       MOVE Y2KFABSX TO Y2KFABSO
       MOVE 'F6-RESET' TO Y2KPF06O
       MOVE -1 TO Y2KFDAYL
       MOVE 'FUTURE DATE WAS SET' TO Y2KMESSO
```

```
       ELSE
        EXEC CICS ASKTIME ABSTIME(ABSTIME)
        END-EXEC
*
        EXEC CICS FORMATTIME ABSTIME(ABSTIME)
                  TIME(Y2KTIMEO) TIMESEP(':')
                  DDMMYYYY(Y2KDATEO) DATESEP('/')
        END-EXEC
*
        MOVE 'DATE NOT SET FOR TERMINAL,USING CURRENT' TO Y2KMESSO
        MOVE SPACES TO Y2KPFØ6O
        MOVE -1 TO Y2KFDAYL
        MOVE Y2KDATEO(1:2) TO Y2KFDAYO
        MOVE Y2KDATEO(4:2) TO Y2KFMONO
        MOVE Y2KDATEO(7:4) TO Y2KFYEAO
        MOVE SPACES TO Y2KFJULO
        MOVE SPACES TO Y2KFABSO
       END-IF
      ELSE
       PERFORM PROC-CHECKINP
      END-IF
      .
*-------------------------------
* VALIDATES INPUT DATA ENTERED
*-------------------------------
*
 PROC-CHECKINP.
     MOVE SPACES TO Y2KMESSO
     IF Y2KFTERI = SPACES OR
        Y2KFTERI = LOW-VALUES
       MOVE 'TERMINAL MUST BE FILLED' TO Y2KMESSO
       MOVE -1 TO Y2KFTERL
     ELSE
      IF Y2KFDAYI = SPACES OR
         Y2KFDAYI = LOW-VALUES
        MOVE 'DAY MUST BE FILLED' TO Y2KMESSO
        MOVE -1 TO Y2KFDAYL
      ELSE
       IF Y2KFDAYI NOT NUMERIC
        MOVE 'DAY MUST BE NUMERIC' TO Y2KMESSO
        MOVE -1 TO Y2KFDAYL
       ELSE
        MOVE Y2KFDAYI TO Y2KFDAYN
        IF Y2KFMONI = SPACES OR
           Y2KFMONI = LOW-VALUES
          MOVE 'MONTH MUST BE FILLED' TO Y2KMESSO
          MOVE -1 TO Y2KFMONL
        ELSE
         IF Y2KFMONI NOT NUMERIC
          MOVE 'MONTH MUST BE NUMERIC' TO Y2KMESSO
          MOVE -1 TO Y2KFMONL
```

```
           ELSE
            MOVE Y2KFMONI TO Y2KFMONN
            IF Y2KFMONN < 1 OR
               Y2KFMONN > 12
             MOVE 'MONTH MUST BE IN RANGE 1-12' TO Y2KMESSO
             MOVE -1 TO Y2KFMONL
            ELSE
             IF Y2KFYEAI = SPACES OR
                Y2KFYEAI = LOW-VALUES
               MOVE 'YEAR MUST BE FILLED' TO Y2KMESSO
               MOVE -1 TO Y2KFYEAL
             ELSE
              IF Y2KFYEAI NOT NUMERIC
               MOVE 'YEAR MUST BE NUMERIC' TO Y2KMESSO
               MOVE -1 TO Y2KFYEAL
              ELSE
               MOVE Y2KFYEAI TO Y2KFYEAN
               IF Y2KFYEAN < 199Ø OR
                  Y2KFYEAN > 2Ø99
                MOVE 'YEAR MUST BE IN RANGE 199Ø-2Ø99' TO Y2KMESSO
                MOVE -1 TO Y2KFYEAL
               END-IF
              END-IF
             END-IF
            END-IF
           END-IF
          END-IF
         END-IF
        END-IF
       END-IF
       IF Y2KMESSO = SPACES
        DIVIDE Y2KFYEAN BY 4 GIVING CNT REMAINDER CNTS
        IF CNTS = Ø
         MOVE 29 TO MONTH(2)
        ELSE
         MOVE 28 TO MONTH(2)
        END-IF
        IF Y2KFDAYN < 1 OR
           Y2KFDAYN > MONTH(Y2KFMONN)
         MOVE 'DAY IS INVALID' TO Y2KMESSO
        END-IF
        MOVE -1 TO Y2KFDAYL
       END-IF
       .
*-------------------------------------
* INITIAL TRANSACTION START
*-------------------------------------
*
 STARTIT.
*
       EXEC CICS
```

```
           SEND CONTROL
           FREEKB
           ERASE
       END-EXEC
*
       MOVE LOW-VALUES TO IPPIY2KI
       MOVE -1 TO Y2KFDAYL
       MOVE 'IPPA SET APPLICATION DATE        ' TO Y2KTIT1O
       MOVE 'IPPCIY2K' TO Y2KPROGO
       MOVE 'IPPIY2K' TO Y2KMAPNO
       MOVE EIBTRMID TO Y2KTERMO Y2KFTERO
*
       EXEC CICS ASSIGN NETNAME(Y2KNETNO)
                        USERID(Y2KUSERO)
                        APPLID(Y2KAPPLO)
       END-EXEC
*
       EXEC CICS ASKTIME ABSTIME(ABSTIME)
       END-EXEC
*
       EXEC CICS FORMATTIME ABSTIME(ABSTIME)
                 TIME(Y2KTIMEO) TIMESEP(':')
                 DDMMYYYY(Y2KDATEO) DATESEP('/')
       END-EXEC
*
       MOVE 'F3-END' TO Y2KPFØ3O
       MOVE 'F4-UPD' TO Y2KPFØ4O
*
       PERFORM SCAN-SHSTOR
       IF CNTS > Ø
        MOVE SH-FDAY(CNTS) TO Y2KFDAYO
        MOVE SH-FMON(CNTS) TO Y2KFMONO
        MOVE SH-FYEA(CNTS) TO Y2KFYEAO
        MOVE SH-FDAT(CNTS) TO Y2KFJULO
        MOVE SH-FABS(CNTS) TO Y2KFABSN
        MOVE Y2KFABSX TO Y2KFABSO
        MOVE 'F6-RESET' TO Y2KPFØ6O
        MOVE 'FUTURE DATE WAS SET' TO Y2KMESSO
       ELSE
        MOVE 'DATE NOT YET SET FOR THIS TERMINAL' TO Y2KMESSO
        MOVE SPACES TO Y2KPFØ6O
        MOVE -1 TO Y2KFDAYL
        MOVE Y2KDATEO(1:2) TO Y2KFDAYO
        MOVE Y2KDATEO(4:2) TO Y2KFMONO
        MOVE Y2KDATEO(7:4) TO Y2KFYEAO
       END-IF
       .
*-------------------------------------
* SCANS THE SHARED STORAGE FOR THE REQUESTED TERMINAL
*-------------------------------------
*
```

```
 SCAN-SHSTOR.
*
      EXEC CICS EXTRACT EXIT PROGRAM('IPPCEIOU')
           GASET(GA-PTR) GALENGTH(GA-LEN)
           RESP(CMD-RESP)
      END-EXEC
*
      IF CMD-RESP NOT = DFHRESP(NORMAL)
       MOVE 'EXTRACT EXIT FAILED' TO TEXT-WORK
       PERFORM DISPERR
      ELSE
       IF GA-PTR-NUM = Ø
        MOVE 'NO GLOBAL EXIT AREA' TO TEXT-WORK
        PERFORM DISPERR
       ELSE
        SET ADDRESS OF GASTOR TO GA-PTR
        IF SH-PTR-NUM = Ø
         MOVE 'NO SHARED STORAGE ALLOCATED' TO TEXT-WORK
         PERFORM DISPERR
        ELSE
         SET ADDRESS OF SHSTOR TO SH-PTR
         MOVE Ø TO CNTS
         PERFORM VARYING CNT FROM 1 BY 1 UNTIL CNT > SH-CNT OR
                                               CNTS > Ø
          IF SH-FTER(CNT) = Y2KFTERO
           MOVE CNT TO CNTS
          END-IF
         END-PERFORM
        END-IF
       END-IF
      END-IF
     .
*-------------------------------------
* PSEUDO-CONVERSE, RETURN TO CICS
*-------------------------------------
*
 RET-TO-CICS.
*
     EXEC CICS ASKTIME ABSTIME(ABSTIME)
     END-EXEC
*
     EXEC CICS FORMATTIME ABSTIME(ABSTIME)
               TIME(Y2KTIMEO) TIMESEP(':')
               DDMMYYYY(Y2KDATEO) DATESEP('/')
     END-EXEC
*
     EXEC CICS SEND MAP('IPPIY2K') MAPSET('IPPIY2K') CURSOR
          FREEKB
     END-EXEC
     MOVE Y2KFTERO TO OLD-FTER
```

```
           MOVE LOW-VALUES TO IPPIY2KI
           EXEC CICS RETURN TRANSID(EIBTRNID)
                COMMAREA(COMMAREA)
                LENGTH(LENGTH OF COMMAREA)
           END-EXEC
           .
      *-------------------------------
      * STOP THE TRANSACTION
      *-------------------------------
      *
       ENDIT.
           EXEC CICS
               SEND CONTROL
               FREEKB
               ERASE
           END-EXEC
           PERFORM DISPERR
           .
       DISPERR.
           EXEC CICS
               SEND TEXT
               FROM(TX-QUIT-TO-CICS)
               LENGTH(LENGTH OF TX-QUIT-TO-CICS)
               FREEKB
               ERASE
           END-EXEC
           EXEC CICS RETURN
           END-EXEC
           .
```

IPPIY2K MAP

```
IPPIY2K  DFHMSD TYPE=&SYSPARM,MODE=INOUT,LANG=COBOL,                   C
               DATA=FIELD,TERM=3270,TIOAPFX=YES,STORAGE=AUTO,          C
               MAPATTS=(COLOR,HILIGHT),DSATTS=(COLOR,HILIGHT)
*
IPPIY2K  DFHMDI SIZE=(24,80),LINE=01,COLUMN=01,COLOR=BLUE
* HEADER LINE 1
Y2KPROG  DFHMDF POS=(01,01),LENGTH=08,ATTRB=(ASKIP)
Y2KTIT1  DFHMDF POS=(01,10),LENGTH=40,ATTRB=(ASKIP),COLOR=NEUTRAL
Y2KNETN  DFHMDF POS=(01,52),LENGTH=08,ATTRB=(ASKIP)
Y2KAPPL  DFHMDF POS=(01,61),LENGTH=08,ATTRB=(ASKIP)
Y2KDATE  DFHMDF POS=(01,70),LENGTH=10,ATTRB=(ASKIP)
* HEADER LINE 2
Y2KMAPN  DFHMDF POS=(02,01),LENGTH=08,ATTRB=(ASKIP)
Y2KTIT2  DFHMDF POS=(02,10),LENGTH=40,ATTRB=(ASKIP),COLOR=NEUTRAL
Y2KTERM  DFHMDF POS=(02,52),LENGTH=04,ATTRB=(ASKIP)
Y2KUSER  DFHMDF POS=(02,61),LENGTH=08,ATTRB=(ASKIP)
Y2KTIME  DFHMDF POS=(02,70),LENGTH=10,ATTRB=(ASKIP)
* THE DATA PORTION ...
```

```
         DFHMDF POS=(5,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                 C
               INITIAL='   TERMINAL ID  ===>',COLOR=PINK
*                      12345678Ø1234567890
Y2KFTER  DFHMDF POS=(5,22),LENGTH=4,ATTRB=(BRT,UNPROT,FSET,IC,NUM)
         DFHMDF POS=(5,27),LENGTH=1
*
         DFHMDF POS=(7,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                 C
               INITIAL='   FUTURE DAY   ===>',COLOR=PINK
*                      12345678Ø1234567890
Y2KFDAY  DFHMDF POS=(7,22),LENGTH=2,ATTRB=(BRT,UNPROT,FSET,IC,NUM)
         DFHMDF POS=(7,25),LENGTH=1
*
         DFHMDF POS=(9,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                 C
               INITIAL='   FUTURE MONTH ===>',COLOR=PINK
*                      12345678Ø1234567890
Y2KFMON  DFHMDF POS=(9,22),LENGTH=2,ATTRB=(BRT,UNPROT,FSET,NUM)
         DFHMDF POS=(9,25),LENGTH=1
*
         DFHMDF POS=(11,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                C
               INITIAL='   FUTURE YEAR  ===>',COLOR=PINK
*                      12345678Ø1234567890
Y2KFYEA  DFHMDF POS=(11,22),LENGTH=4,ATTRB=(BRT,UNPROT,FSET,NUM)
         DFHMDF POS=(11,27),LENGTH=1
*
         DFHMDF POS=(15,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                C
               INITIAL='   JULIAN DATE     :',COLOR=YELLOW
*                      12345678Ø1234567890
Y2KFJUL  DFHMDF POS=(15,22),LENGTH=7,ATTRB=(ASKIP)
         DFHMDF POS=(15,3Ø),LENGTH=1
*
         DFHMDF POS=(16,1),LENGTH=2Ø,ATTRB=(NORM,PROT),                C
               INITIAL='   ASKTIME DIF.    :',COLOR=YELLOW
*                      12345678Ø1234567890
Y2KFABS  DFHMDF POS=(16,22),LENGTH=16,ATTRB=(ASKIP,NUM)
         DFHMDF POS=(16,39),LENGTH=1
*
*
* MESSAGE LINE 21
Y2KMESS  DFHMDF POS=(21,Ø1),LENGTH=79,ATTRB=(ASKIP,BRT),INITIAL=' ',   C
               COLOR=RED
* COMMAND LINE 22
         DFHMDF POS=(22,Ø1),LENGTH=12,ATTRB=(ASKIP),                   C
               INITIAL='COMMAND ===>'
Y2KCOMM  DFHMDF POS=(22,14),LENGTH=4Ø,ATTRB=(BRT,UNPROT)
* Y2KPFS LINE 23
Y2KPFØ1  DFHMDF POS=(23,Ø1),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ3  DFHMDF POS=(23,14),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ5  DFHMDF POS=(23,27),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ7  DFHMDF POS=(23,4Ø),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ9  DFHMDF POS=(23,53),LENGTH=12,ATTRB=(ASKIP)
Y2KPF11  DFHMDF POS=(23,66),LENGTH=12,ATTRB=(ASKIP)
```

```
* Y2KPFS LINE 24
Y2KPFØ2  DFHMDF POS=(24,Ø1),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ4  DFHMDF POS=(24,14),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ6  DFHMDF POS=(24,27),LENGTH=12,ATTRB=(ASKIP)
Y2KPFØ8  DFHMDF POS=(24,4Ø),LENGTH=12,ATTRB=(ASKIP)
Y2KPF1Ø  DFHMDF POS=(24,53),LENGTH=12,ATTRB=(ASKIP)
Y2KPF12  DFHMDF POS=(24,66),LENGTH=12,ATTRB=(ASKIP)
*
         DFHMSD TYPE=FINAL
         END
```

Now that all the programs are translated, compiled or assembled, and link-edited, we can start testing. Firstly, define all three transaction codes (EIEN, EIDI, and IY2K) in CEDA. These are just normal transactions. We use auto-install for all programs, and this works perfectly. If you didn't use the PLTPI for enabling the exits, issue the EIEN transaction to enable the GLUE programs (messages are written to the CICS message log). Hereafter, you can start using the IY2K transaction to set dates for your subsequent transactions.

Because we are using terminal auto-install in the CICS regions, we could also easily implement clean-up logic in the auto-install exit program. So, when a terminal is being cleaned up from the CICS system, we also clean up the entry in the shared storage if needed. (In fact this is exactly the same logic as the PF6 action in the IPPCIY2K program.) We feel that 500 entries in the shared storage table is more than enough for our testing needs.

NOTES

- As this is a useful tool in the test and development environment, we were not concerned about locking requirements against the shared table – some action might be needed in that area.
- We are running the IY2K solution in a CICS 4.1 environment. For the XPCFTCH code to work, the solution for APAR PN86734 (PTF UN93801) must be applied (otherwise the returned EIB address will be X'00000000').

Stan Adriaensen
Systems Engineer
Groupe Royale Belge/ IPPA (Belgium)

The CICS Log Manager

INTRODUCTION

The CICS Log Manager is a new CICS component, and was created and first shipped with CICS Transaction Server for OS/390 Release 1. It is a new CICS Domain, which replaces the journal control management programs of previous releases. Being restructured into a Domain and utilizing object-oriented programming techniques, the CICS Log Manager benefits from the improved reliability and serviceability that this brings.

CICS TRANSACTION SERVER

This article makes reference to CICS Transaction Server for OS/390 Releases 1 and 2. The CICS Transaction Server is a member of the OS/390 family of MVS-based software servers. IBM has integrated CICS with a set of other supporting software, offering a single product in their place.

The CICS component of CICS Transaction Server Release 1 has a release number of 0510. The CICS component of CICS Transaction Server Release 2 has a release number of 0520. However, there is no separately orderable product (such as 'CICS/ESA 5.1.0') – it is the CICS component of CICS Transaction Server Release 1.

JOURNAL CONTROL BEFORE CICS TRANSACTION SERVER

Prior to CICS Transaction Server Release 1, CICS journal control was handled by DFHJCP – the CICS Journal Control Program. This coordinated the various types of log record written by both CICS control programs and applications, buffering them into blocks of records in virtual storage buffers, and writing them to CICS journal datasets on DASD devices when the buffer space had filled or when a journal request that forced I/O had been issued.

DFHJCP used BSAM as the access method to perform I/O for its journalling requests to and from the datasets. BSAM needs to run

below the 16MB line, and so CICS journal buffers resided below the 16MB line in the CICS region. Being BSAM datasets, this meant CICS journals were sequential files.

In addition to DFHJCP, journal management was also handled by a number of other CICS service modules such as DFHJCO (for journal opens), DFHJCC (for closes), and DFHJCEOV (for journal advances).

CICS also had the capability of journalling to tape devices rather than to DASD.

CICS had a journal dedicated to system recovery, should an emergency restart of the CICS system be required. Two datasets were normally used – DFHJ01A and DFHJ01B. This system recovery journal was known as the CICS system log (syslog). If just one dataset was defined then only DFHJ01A was used for the system log. In the normal dual dataset case, CICS switched between the two when the current dataset had filled up. When a dataset had been filled, it could be archived before being reused. Dual datasets allowed archives of one to occur while CICS continued writing to the other.

Other journals defined to CICS were used to record user information such as audit trails, or to store data for purposes such as CICS file control forward recovery operations.

CICS created unique journal tasks during system initialization. There was one journal task for each journal defined to CICS. When an application issued a journalling request that led to I/O, the specific journal task for the journal to be written to was dispatched. The journal task waited for BSAM to complete the I/O operation; it did this by waiting for a physical ECB (PECB) to be posted by BSAM. There was one PECB per journal. The task that issued the journalling request (and any other tasks synchronized on the I/O) waited on logical ECBs (LECBs). The LECBs were pooled for use by any tasks that needed to wait on journalling. The journal task posted these LECBs, when its PECB had been posted by BSAM.

Having resumed all the synchronized tasks by posting their LECBs, the journal task suspended itself once more, in readiness for the next I/O operation.

The CICS journal control management of CICS/ESA, as described above, is established code that has evolved over the years to support CICS in the traditional MVS environment. CICS Transaction Server has been developed to exploit the parallel sysplex environment. This is a combination of software and hardware facilities designed to provide high-speed sharing of data and communications between separate MVS regions, bound together through the use of new coupling methods. Therefore, CICS Transaction Server requires a journalling mechanism that is able to fully exploit this new environment.

The CICS Log Manager Domain is built upon the use of the MVS System Logger. This is a component of OS/390 or MVS/ESA SP 5.2; it runs as a separate subsystem called IXGLOGR under MVS, and provides an API for other jobs to write, read, browse, and delete data to and from various logs.

The MVS System Logger manages log data in entities known as log streams – a log stream being a series of blocks of data. Each log stream is identified by its own (unique) log stream identifier, known as the log stream name. The format of these names can vary; an example is the form userid.applid.journalname, eg PRODCICS.CICS1A.DFHLOG. The CICS Log Manager implements various log streams for its own use, and others are available for user purposes.

The log stream name is the identifier used by the MVS System Logger to reference a particular log stream. From an application point of view, however, it is somewhat unwieldy. CICS applications can therefore refer to journal names when writing log data. These are up to eight characters in length, an example being DFHJ02. The situation is analogous to files and datasets, where, for example, an application could write to file FILEA, but under the covers CICS would call VSAM to write to dataset PRODCICS.CICS1A.FILEA.

The CICS system log is used to record all records required to provide dynamic transaction backout of a failing Unit Of Work (UOW) – for example, when a task abends having written to a recoverable VSAM file. In addition, the CICS system log is used to recover a CICS system to a committed state when performing an emergency restart. The CICS system log is therefore used to record ‘before-images’ of

changes to resources managed under CICS. Encapsulated by the CICS Log Manager, and therefore transparent to the rest of CICS, the CICS system log is physically implemented by two separate log streams:

- The primary log stream (also referred to as journalname DFHLOG) is used to hold data for most short-lived UOWs.
- The secondary log stream (also known as journalname DFHSHUNT) is used to hold data for those UOWs that are not short-lived, for example because they failed for some reason during part of CICS syncpoint processing, or because they represent long-running tasks that log and syncpoint infrequently.

CICS Log Manager is responsible for handling the movement and manipulation of UOW log data on these two CICS system log streams.

By exploiting the system log streams for dynamic transaction backout as well as for system recovery, CICS Transaction Server does not have to maintain separate 'in-core' dynamic logs for each inflight UOW. This contributes towards virtual storage constraint relief with CICS Transaction Server. Since the MVS System Logger provides the ability to both read and write to log streams, the CICS Log Manager is able to retrieve log data from the CICS system log directly and has no need for a dynamic log duplication of system log data. This is a big advantage over previous releases of CICS, where the dynamic log was required to provide the data for the backing out of a UOW, since closing the system log dataset for output, then opening it for input when backing out a UOW, would be completely impractical.

The CICS system log is intended for use for recovery purposes only and is not meant to be used for any other reason.

Another new object-oriented Domain in CICS Transaction Server (the Recovery Manager Domain) coordinates UOW and CICS system recovery. Recovery Manager invokes Log Manager to store and retrieve log data for commit and backout purposes.

In addition to the system log, CICS also provides support for what are known as general logs. Examples of these include user journals (for maintaining audit trails or company records), auto-journalling (for

CICS file control and terminal control), and forward recovery logging. Forward recovery logs are written to by CICS file control, and record 'after-images' of changes made to recoverable VSAM files for use with products such as CICSVR or its equivalent. CICS also provides a 'log-of-logs' (DFHLGLOG) which records information on log stream exception conditions, and also maintains tie-up records for forward recovery purposes.

Each CICS system running in a sysplex has its own unique CICS system log (and hence its own unique log streams for the system log). Conversely, forward recovery logs are shared across all the CICS systems within a sysplex. User journals may be shared between various CICS systems within the sysplex, or they can be stand-alone.

Log streams for use by CICS can be pre-defined to the MVS System Logger; alternatively, a dynamic definition can be created on first reference to a particular log stream, by using MVS log stream model definitions.

MVS SYSTEM LOGGER CONSIDERATIONS

The MVS System Logger exploits the Coupling Facility component of a sysplex. This may be either a 9674 stand-alone device or an LPAR-based Coupling Facility installed in an ES/9000 or 9672 machine. The Coupling Facility provides high-speed cacheing, list processing, and locking between each member of a sysplex. CICS Transaction Server Release 1 requires a Coupling Facility as a pre-requisite for its use. Note that with CICS Transaction Server Release 2 this is no longer a requirement, since it exploits new function added to the MVS System Logger in OS/390 Version 2 Release 4 that provides DASD-only logging support.

Those customers installing CICS Transaction Server Release 1, or those installing Release 2 who have a full sysplex environment and use a Coupling Facility, are required to define various Coupling Facility list structures for its use. These are designed for high-speed data storage and access between members of the sysplex. The MVS System Logger maps the log streams onto these list structures. Correct sizing of the structures used for CICS system log streams is an

important consideration. This is because the MVS System Logger has to manage the amount of structure space in use. When a predefined upper boundary value is reached (the HIGHOFFLOAD value as specified on the log stream definition), the MVS System Logger will move log data from the structure onto secondary storage (VSAM linear DASD datasets) until the predefined lower boundary value (LOWOFFLOAD) is achieved. This process is known as offloading. For those log streams that are not referred to from an on-line environment (eg user journals recording audit trails) this is not a problem. It would be a far more serious issue if this offloaded log data was required for dynamic transaction backout of a UOW. The performance of CICS backout processing would be unacceptable if any system log data had to be retrieved from DASD on a regular basis. For this reason, the sizing of the structures used for CICS system log streams is important. CICS should ideally have sufficient structure space to hold all inflight UOW log records within the Coupling Facility and prevent their spillage to secondary storage.

In addition to secondary storage, there is also a tertiary storage layer, which is when datasets are archived to DFHSM.

CICS SYSTEM LOG MANAGEMENT

As UOWs complete and tasks terminate on a CICS system, so the log data written to the CICS system log by those UOWs is no longer required for backout purposes. Prior to CICS Transaction Server, such redundant log data remained on the DFHJ01A and DFHJ01B datasets until subsequent writes to the system log caused it to wrap around and overlay the data.

Therefore, those long running transactions (typically conversational ones) that updated recoverable resources, then did nothing else for a considerable time, would have their backout data remain on the system log and be followed by log records for many other UOWs that ran after them. There was the danger that a log-wrap condition could occur, when DFHJ01A and DFHJ01B filled up and began overwriting backout data for those long-running UOWs that were still inflight. If CICS crashed and was emergency-restarted in such a situation, a DFHRU2802 condition would result if the user was unable to manually

back out the changes made by such UOWs.

To avoid this situation, and to prevent unnecessary CICS system log data for completed UOWs from remaining on the log, CICS Log Manager dynamically moves and deletes log records from the system log. At the time of an activity keypoint, the CICS Log Manager will determine the position of the oldest log data still required for any UOW of interest to CICS. It can then call the MVS System Logger to delete the CICS system log stream data up to this point. This process is known as trimming the tail of the system log. In addition, log records for those UOWs that are deemed long-running are moved from the primary system log stream to the secondary system log stream. This avoids the space used for the primary system log stream from being used for log data that will very probably not be referenced for some time. Failure to do this trimming would prevent adequate space for the primary system log stream from being available.

Note that the CICS Log Manager does not provide a mechanism for trimming log data from general log streams in this manner.

BRINGING IT ALL TOGETHER

Figure 1 brings together the various concepts described above. In the example shown, the sysplex consists of two MVS regions (MVS1 and MVS2). MVS1 is running three CICS systems (CICS1A, CICS1B, and CICS1C). MVS2 is running two CICS systems (CICS2A and CICS2B). Each MVS region is running an IXGLOGR subsystem to provide MVS System Logger services for that image. Both MVS regions are connected to a Coupling Facility; here, the log streams used by the various CICS systems are mapped to list structure storage. Finally, secondary storage for offloaded log data, and archived tertiary storage, is shown.

DFHJUP IMPLICATIONS AND ENHANCEMENTS

The DFHJUP utility has been provided with earlier releases of CICS to read and process CICS journal datasets. With CICS Transaction Server, journal datasets no longer exist – DFHJUP needs to be run against log streams instead.

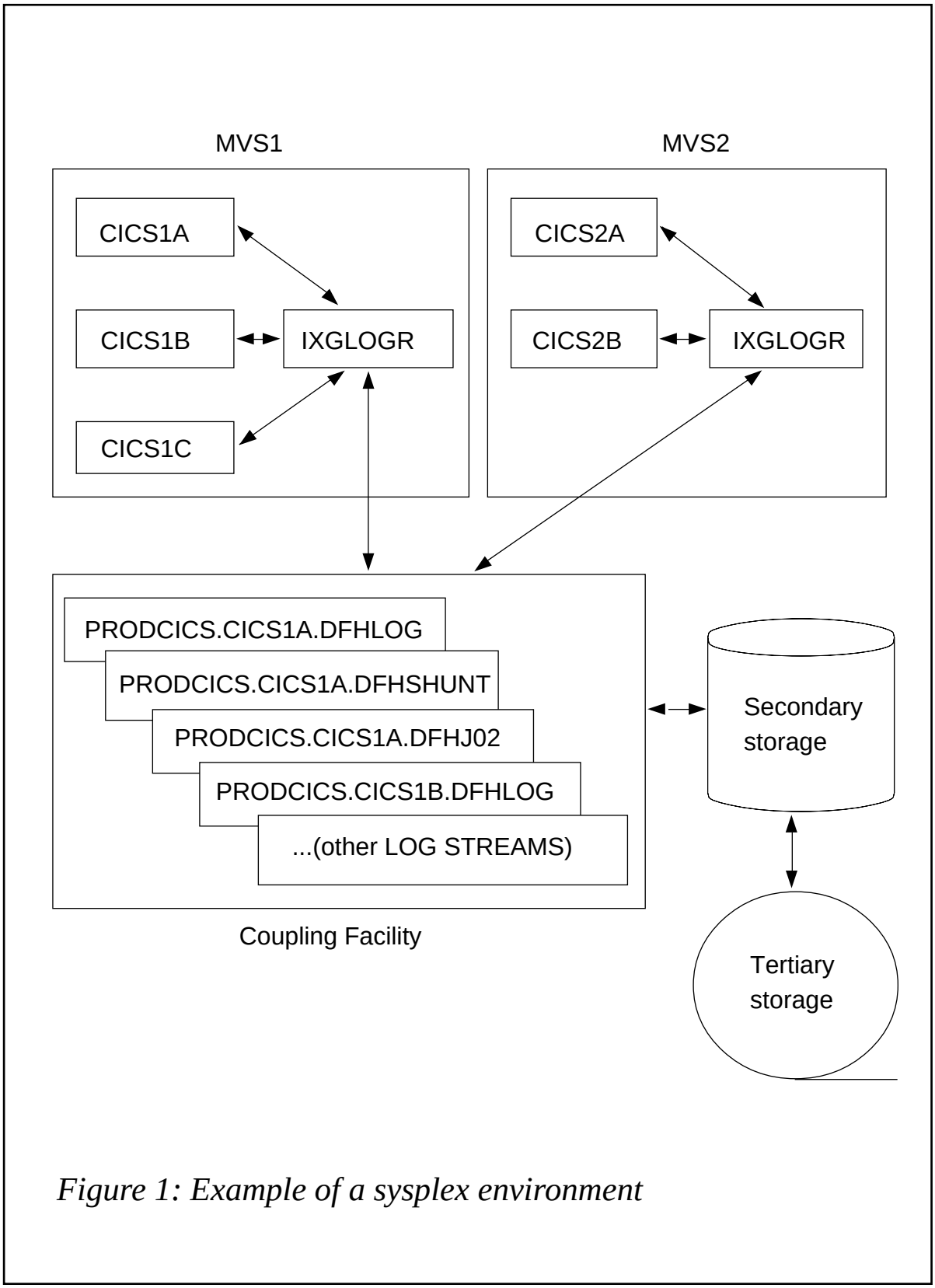

Figure 1: Example of a sysplex environment

In order to support this change, the JCL used to submit a DFHJUP job now supports a SUBSYS parameter on the input DD card. This allows the user to specify various options on the job, such as: 'DELETE' if the log stream data is to be deleted; 'COMPAT41' if it is to be returned in a CICS/ESA log record format; and 'TO=', which sets a limitation on how far through the log stream the job is to progress before ending.

Since DFHJUP has to read a log stream in order to copy or print it, this can be an unnecessary overhead if the job is being run with 'DELETE' specified – merely so that the log stream data may be deleted. To avoid this overhead, a job specifying EXEC PGM=IEFBR14 rather than DFHJUP can be submitted, with the necessary SUBSYS options. This allows the MVS System Logger to be called to delete some or all of the log stream data, without the additional effort of reading it first. With large volumes of log stream data this can be a beneficial saving.

While users may wish to use DFHJUP and/or IEFBR14 jobs to manage the size of their log streams for general logs, making use of the 'DELETE' option, care should be taken when performing similar management of the CICS system log streams. If CICS Log Manager is left to automatically handle the deletion of log data from the system log streams, then there should not be any need for manual deletion of data using a batch utility. If log data were deleted from the CICS system log that was still required for inflight UOWs, then dynamic transaction backout of a failing task or emergency restart of a failed CICS system would fail because incomplete recovery information was available from the CICS system log. For this reason, it is good policy never to manually delete log data from the CICS system log streams unless there is very good reason to do so, and unless the implications of doing so are fully understood.

An example of JCL and control statements follows. This is used to submit DFHJUP to print a section of general log stream data up to a limiting time value, then delete it up to that time value:

```
//DBAJUP JOB (accounting information),CLASS=A
//JUP    EXEC PGM=DFHJUP
//STEPLIB DD DSN=CTS110.CICS510.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1  DD DSN=PRODCICS.CICS1A.DFHJ02,
//     DCB=BLKSIZE=32760,
//     SUBSYS=(LOGR,DFHLG510,'TO=(1997/365,13:10)',DELETE
```

```
//SYSUT4  DD DUMMY
//SYSIN  DD *
OPTION PRINT
END
/*
```

In addition, DFHJUP has been enhanced to display more information at the start of each log block and record within the block that it formats out. Figure 2 shows an example of a DFHJUP print from a log stream. The log block is preceded by the MVS System Logger block identifier, the length of the block, and the GMT and local timestamps of when it was written. The records themselves are preceded by a new column which gives their offset in hexadecimal from the start of the block.

DFHLSCU – THE LOG STREAM SIZING UTILITY

DFHLSCU is a utility provided with CICS Transaction Server that is intended to help establish values for various parameters which need to be set when defining log stream structure sizes. The utility runs as a standard batch job, and analyses CICS/ESA journal datasets provided as input to it. From these CICS/ESA journals, DFHLSCU generates a report detailing the breakdown of the journal logging activity into time intervals, and showing the volume of log records written by each CICS component during those periods. Using this information, it provides recommendations on the definition of the equivalent log stream that is to be defined for CICS Transaction Server.

CICS LOG MANAGER SERVICE RECOMMENDATIONS

There have been a small number of APARs raised against the Log Manager Domain and associated utilities in CICS Transaction Server Release 1. It is recommended that customers apply the following PTFs to their systems in order to have these fixes available to them:

- APAR PQ05936/PTF UQ08279 (against the Log Manager Domain).
- APAR PQ04998/PTF UQ07483 (against the DFHJUP utility).
- APAR PQ01827/PTF UQ02090 (against the DFHLSCU utility).

Applying these PTFs will pre-require those other (earlier) fixes that

```
Block identifier - 0000000000001165

Length of block  -         00000172


GMT timestamp    - AFB1935203408E02   04/04/1998 13:24:11.163656

Local timestamp  - AFB1935203408E02   04/04/1998 13:24:11.163656


 000000   000000  6EC4C6C8 01400001  C9E8C3D3 D6D5C3C6   AFB19351 E24E0400  AFB19351 E24B0400  *>DFH. ..IYCLONCF..L.S+....L.S...*
 000020           00000000 00000003  00000000 00000C35   00000172                              *.....................           *
 000000   000034  00000069 00000010  AFB1880D D95E1C00   00000000 00000C35  01010000 0000049E  *.........H.R;..................*
 000020           00000001 00000000  00000C35 01010000   0000049E 001CE0D9  D4E4E600 40404040  *.......................RMUW.    *
 000040           40404040 40404040  C3C1E3C1 0000022C   0011E0C1 D7D9C4AF  B1880DB5 A0CC0201  *        CATA.......APRD..H......*
 000060           8000087A CID7D9C4  06                                                       *...:APRD.                          *
 000000   00009D  000000D5 00000010  AFB19365 E93B9000   00000000 00000003  00010000 00000034  *..N.......L.Z...................*
 000020           00000001 00000000  00000000 00000000   00000000 001CE0D9  D4E4E600 F8F7F3F6  *.......................RMUW.8736*
 000040           D7E8D2E2 F8F7F3F6  C3C5C3C9 0000034C   0054E0C1 D7E4E2AF  B1935D9B 6F680004  *PYKS8736CECI...<...APUS..L).?...*
 000060           20F8F7F3 F6D7E8D2  E2F8F7F3 F6000003   4CC3C5C3 C9C3C9C3  E2E4E3C5 D9404040  *.8736PYKS8736...<CECICICSUSER   *
 000080           1A11C7C2 C9C2D4C9  E8C14BD7 E8D2E2F8   F7F3F6B1 935D9B6F  68000100 00000000  *..GBIBMIYA.PYKS8736.L).?........*
 0000A0           00000080 000B7AC1  D7E4E201 80000000   00001000 00034CC3  C5C3C9F8 F7F3F600  *......:APUS...........<CECI8736.*
 0000C0           00000CC1 E6404000  000000E3 C5E2E340   C4C1E3C1 F1                           *...AW  ....TEST DATA1           *
```

Figure 2: Example of a DFHJUP print from a log stream

are also recommended to be applied to the Log Manager Domain and associated CICS utilities.

DASD ONLY LOGGING

OS/390 Version 2 Release 4 has provided a new function in the MVS System Logger to support DASD-only logging. This removes the pre-requisite of a Coupling Facility for CICS Transaction Server Release 2. DASD-only logging allows a migration path for users with single MVS regions, and those customers with non-parallel sysplex environments. Note that DASD-only logging can be used in a parallel sysplex, but sharing of log streams written to by different jobs is only supported within a particular MVS region.

The choice of Coupling Facility or DASD-only log streams is transparent at the application level. Log data is still written and read via the CICS Log Manager. The difference comes when the log data reaches the MVS System Logger. With DASD-only log streams, there are no structures defined with a Coupling Facility. The log data is written from an MVS System Logger Data Space onto the VSAM linear datasets which would be used as secondary storage in a Coupling Facility logging environment.

DASD-only log streams are useful in development environments, and for test CICS regions. They are also seen as a solution to the migration from CICS/ESA 4.1.0 to CICS Transaction Server Release 2 for those customers who are not intending to implement the parallel sysplex solution.

FURTHER READING

The following selection of manuals from IBM publications provides information on the implementation and use of log streams, CICS Log Manager, and MVS System Logger services in general:

- *OS/390 MVS Assembler Services Guide* (GC28-1862-02).
- *CICS Transaction Server for OS/390 Release Guide* (GC33-1570-01).

- *CICS Transaction Server for OS/390 CICS Operations and Utilities Guide* (SC33-1685-01).
- *CICS Transaction Server for OS/390 CICS System Definition Guide* (SC33-1682-01).

SUMMARY AND CONCLUSIONS

The CICS Log Manager component of CICS Transaction Server provides many benefits over the old journal control management of previous releases; these include the automatic on-line merging of log streams (eg for forward recovery across a sysplex), fast direct reading of log data, better log data management, the avoidance of log dataset switching and the wrap-around problem, and virtual storage constraint relief because of no in-core dynamic log storage.

I hope that this article has helped explain the background to MVS System Logger concepts and given a summary of the CICS Log Manager Domain, how it may be used (for both Coupling Facility and DASD-only log streams), and the various utilities that are provided with CICS to support it.

Readers wishing to discuss the material in this article further are welcome to contact me via e-mail, at andyw@vnet.ibm.com.

Andy Wright
CICS Change Team
IBM (UK)

Programmable CICS DB2 attachment switch

OUR PROBLEM

At a scheduled time every evening, we wanted to switch the DB2 attachment in CICS from our production DB2 to a 'mirror' DB2, which runs while the production DB2 is accessed by batch jobs and reorganization of DB2 tablespaces takes place.

The switching of the DB2 attachment can be carried out either from a terminal (using transaction DSNC with the RCT parameter suffix and DB2 subsystem name) or by calling the PLT programs. However, we wanted to build a way for the batch system to connect to CICS and switch the DB2 attachment, and then feed back the result to our batch system.

When we asked IBM, we were told that there is no callable interface to the DB2 attachment programs until at least CICS Version 5.

OUR SOLUTION

Because we could not stop and start the DB2 attachment with a different RCT (the DB2 subsystem name is included in the RCT), we decided to stop and start the DB2 attachment and change the load module of the RCT in the SDSNLOAD (which the RCT load modules are normally linked into).

To communicate between batch and CICS, we have written an EXCI program, called by batch, with a CICS server program intercepting the demand from the batch job, executing it, and returning a return code after execution.

This has worked well, the results of execution being written to the batch output and also to the master terminal.

THE PROGRAMS

BATCHCLI is the batch EXCI client calling CICS.

CICSSRV is the CICS server program.

The JCL for switching the RCT load module and then calling BATCHCLI follows:

```
//COPYRCT  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=X
//OUTMEM   DD DSN=DB2.ACTIVE.SDSNLOAD,DISP=SHR <- active DB2-Loadlib
//INMEM    DD DSN=save.of.RCT.loadlib,DISP=SHR <- save of original RCT
//SYSUT3   DD UNIT=SYSTSO,SPACE=(TRK,(1))
//SYSUT4   DD UNIT=SYSTSO,SPACE=(TRK,(1))
//SYSIN    DD *
COPYMOD    COPY OUTDD=OUTMEM,INDD=INMEM
           SELECT MEMBER=((DSN2CTmi,DSN2CTpr,R))
/*
//SWITCH   EXEC PGM=BATCHCLI,PARM='DBTCHNAMECICSAPPL        ',
//               COND=(0,LT)
//STEPLIB  DD DSN=CICS410.SDFHEXCI,DISP=SHR
//              DD DSN=your.batch.loadlib,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//*
```

The IEBCOPY step replaces DSN2CTpr with DSN2CTmi, this being the RCT pointing to our ‘mirror’ DB2. The step SWITCH calls program BATCHCLI with parameter D in the PARM-card. The BTCHNAME is a symbolic for activity logging, and the CICSAPPL is the application-id of the CICS system.

To switch back to the original RCT module, change the SELECT statement in the IEBCOPY step and re-execute the job:

```
     SELECT MEMBER=((DSN2CTpr,,R))
```

BATCHCLI

```
CBL XOPTS(EXCI,COBOL2)
IDENTIFICATION DIVISION.
PROGRAM-ID. BATCHCLI.
ENVIRONMENT DIVISION.
*=============================================================*
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT PRINTER ASSIGN TO SYSPRINT.
DATA DIVISION.
*=============================================================*
```

```
 FILE SECTION.
*=================================================================*
 FD  PRINTER BLOCK CONTAINS 128 CHARACTERS
    RECORDING MODE S
    LABEL RECORDS OMITTED.
 Ø1  OUTPUT-RECORD      PIC X(128).
 WORKING-STORAGE SECTION.
*=================================================================*
*   Declare call level, DPL, and EXEC level return code areas. *
*=================================================================*
 Ø1 EXCI-EXEC-RETURN-CODE.
   Ø2 EXEC-RESP        PIC 9(8) COMP.
      88  NO-ERROR       VALUE Ø.
   Ø2 EXEC-RESP2       PIC 9(8) COMP.
   Ø2 EXEC-ABCODE      PIC X(4).
   Ø2 EXEC-MSG-LEN     PIC 9(8) COMP.
   Ø2 EXEC-MSG-PTR     POINTER.
*=================================================================*
*   Declare areas to be used for outputting return codes.      *
*=================================================================*
 Ø1  OUTPUT-RETAREA.
   Ø5  FILLER           PIC X(2Ø)  VALUE SPACES.
   Ø5  O-RESP           PIC 9(8).
   Ø5  FILLER           PIC XX     VALUE SPACES.
   Ø5  O-RESP2          PIC 9(8).
   Ø5  FILLER           PIC XX     VALUE SPACES.
   Ø5  OEXCI-SUB-REASON1 PIC 9(8).
   Ø5  O-ABCODE-LINE    REDEFINES OEXCI-SUB-REASON1.
       1Ø  O-ABCODE     PIC X(4).
       1Ø  OPAD-ABCODE  PIC X(4).
   Ø5  FILLER           PIC X(8Ø) VALUE SPACES.
 Ø1  SUB                PIC S9(8) COMP.
 Ø1  OUT-REC.
   Ø5 OUT-REC-ELEM      PIC X OCCURS 128 TIMES.
*=================================================================*
*   card input                                                    *
*=================================================================*
 Ø1  CARTEIN.
   Ø5 CART-CODE          PIC X.
     88 DB2-CART         VALUE 'D'.
   Ø5 FILLER             PIC X.
   Ø5 CART-NOM           PIC X(8).
   Ø5 FILLER             PIC X.
   Ø5 CART-CICS          PIC X(8).
   Ø5 FILLER             PIC X.
   Ø5 CART-ZONE          PIC X(6Ø).
* CART-ZONE used for passage of other commands                   *
*=================================================================*
*   Initialize target information variables.                    *
*=================================================================*
```

```
 Ø1  TARGET-PROGRAM     PIC X(8)       VALUE 'CICSSRV '.
 Ø1  TARGET-SYSTEM      PIC X(8).
 Ø1  TARGET-TRANSID     PIC X(4)       VALUE 'EXCI'.
*================================================================*
*   Define Commarea struct.                                      *
*   COMM-CODE is the function code 'D' for DB2 switch            *
*   COMM-BID  is a symbolic batch-id for logging                 *
*   COMM-CNO  is the number of the card (imagine a stream of     *
*                  SYSIN cards                                   *
*   COMM-RC   is the return code from CICS                       *
*   COMM-ZONE is for passing other information to CICS (like     *
*             the name of a file to close or disable etc)        *
*================================================================*
 Ø1  COMMAREA.
   Ø5  COMM-CODE        PIC X.
   Ø5  COMM-BID         PIC X(8).
   Ø5  COMM-CNO         PIC 9(3).
   Ø5  COMM-RC          PIC X(2).
   Ø5  COMM-ZONE        PIC X(6Ø).
*================================================================*
*   Initialize commarea length and data length (in bytes).       *
*================================================================*
 Ø1  LINK-COM-LEN       PIC S9(4) COMP VALUE 74.
 Ø1  LINK-DAT-LEN       PIC S9(4) COMP VALUE 74.
*================================================================*
*   Initialize program specific variables and flags.             *
*================================================================*
 Ø1  SERVER-RETCODE     PIC S9(8) COMP VALUE ZERO.
 Ø1  SAVED-RESPONSE     PIC 9(8)  COMP VALUE ZERO.
 Ø1  PROGRAM-MESSAGES.
   Ø5  MSGØØ  PIC X(128) VALUE SPACES.
   Ø5  MSGØ1  PIC X(128) VALUE '*============== The very useful Com
-     'munication program from Batch to CICS ========*'.
   Ø5  MSGV   PIC X(128) VALUE '*
-     '                                                *'.
   Ø5  MSGØ6  PIC X(128) VALUE '*    Server Response:
-     '                                                *'.
   Ø5  MSGØ7.
    1Ø MSGØ7Z PIC X(12) VALUE '* Carte no. '.
    1Ø MSGØ7Y PIC 9(3).
    1Ø MSGØ7A PIC X(12) VALUE ' - Connect >'.
    1Ø MSGØ7X PIC X(1).
    1Ø MSGØ7W PIC X(4)  VALUE '< a '.
    1Ø MSGØ7B PIC X(8).
    1Ø MSGØ7C PIC X(19) VALUE ' with parameters = '.
    1Ø MSGØ7D PIC X(54).
    1Ø MSGØ7E PIC X(15) VALUE SPACES.
   Ø5  MSGØ8.
    1Ø MSGØ8A PIC X(33) VALUE '*    Return of CICS server  : RC '.
```

```
    1Ø MSGØ8B PIC X(2).
    1Ø MSGØ8E PIC X(93) VALUE '
-      '           *'.
   Ø5  MSG11  PIC X(128) VALUE '*   Problem with connection CICS.
-      ' Returncode is :                          *'.
   Ø5  MSG13  PIC X(128) VALUE '*   Message returning from CICS
-      '                     :                    *'.
   Ø5  MSG14  PIC X(128) VALUE '*    >>>> Execution stopped <<<<
-      '                                          *'.
   Ø5  MSG31  PIC X(128) VALUE '*=================== Program BATCH
-      'CLI finished. ================================*'.
*================================================================*
 LINKAGE SECTION.
   Ø1  EXEC-LEVEL-MSG.
     Ø5 EXEC-LEVEL-MSG-TEXT PIC X OCCURS 128 TIMES.
*
   Ø1  PARM-DATA.
     Ø5 PARM-STRING-LENGTH  PIC 9(4) COMP.
     Ø5 PARM-CODE           PIC X.
     Ø5 PARM-NOM            PIC X(8).
     Ø5 PARM-CICS           PIC X(8).
     Ø5 PARM-ZONE           PIC X(6Ø).

 PROCEDURE DIVISION USING PARM-DATA.
     OPEN OUTPUT PRINTER.
     MOVE SPACES TO TARGET-SYSTEM
     MOVE PARM-CICS TO TARGET-SYSTEM.
     PERFORM CALL-BY-PARM THRU CALL-BY-PARM-END.
     GO TO FINISH-PROGRAM.
*================================================================*
 CALL-BY-PARM.
*
     WRITE OUTPUT-RECORD FROM MSGØØ.
     WRITE OUTPUT-RECORD FROM MSGØ1.
     WRITE OUTPUT-RECORD FROM MSGV.
*  Set up the Commarea for transmission.
     MOVE PARM-CODE TO COMM-CODE.
     MOVE PARM-ZONE TO COMM-ZONE.
     MOVE 1         TO COMM-CNO.
     MOVE PARM-NOM  TO COMM-BID
     MOVE ZERO TO COMM-RC.
     MOVE TARGET-SYSTEM TO MSGØ7B.
     MOVE CART-CODE  TO MSGØ7X.
     MOVE Ø TO MSGØ7Y.
     MOVE COMM-ZONE TO MSGØ7D.
     WRITE OUTPUT-RECORD FROM MSGØ7.
*
*  Perform the Link Request;
*
```

```
     EXEC CICS LINK PROGRAM(TARGET-PROGRAM)
                    TRANSID(TARGET-TRANSID)
                    APPLID(TARGET-SYSTEM)
                    COMMAREA(COMMAREA)
                    LENGTH(LINK-COM-LEN)
                    DATALENGTH(LINK-DAT-LEN)
                    RETCODE(EXCI-EXEC-RETURN-CODE)
                    SYNCONRETURN
     END-EXEC.
*
*  Check on how well the request has performed. We may have
*  to abort further processing if either the link or the
*  server program failed in any way.
*
     IF NO-ERROR THEN
        MOVE  COMM-RC   TO MSGØ8B
        WRITE OUTPUT-RECORD FROM MSGØ8
        WRITE OUTPUT-RECORD FROM MSGV
        GO TO FINISH-PROGRAM
     ELSE
        PERFORM LINK-NOT-OK THRU LINK-NOT-OK-END
        GO TO FINISH-PROGRAM.
 CALL-BY-PARM-END.
     EXIT.
*================================================================*
*   In case of a failed LINK to CICS                             *
*================================================================*
 LINK-NOT-OK.
     WRITE OUTPUT-RECORD FROM MSG11.
     MOVE  EXEC-RESP TO O-RESP.
     MOVE  EXEC-RESP2 TO O-RESP2.
     MOVE  SPACES TO OPAD-ABCODE.
     MOVE  SPACES TO OPAD-ABCODE.
     MOVE  EXEC-ABCODE TO O-ABCODE.
     WRITE OUTPUT-RECORD FROM OUTPUT-RETAREA.
     IF EXEC-MSG-PTR IS NOT EQUAL TO NULLS THEN
        WRITE OUTPUT-RECORD FROM MSG13
        WRITE OUTPUT-RECORD FROM MSGV
        SET ADDRESS OF EXEC-LEVEL-MSG TO EXEC-MSG-PTR
        MOVE SPACES TO OUT-REC
        PERFORM TEST BEFORE
        VARYING SUB FROM 1 BY 1
        UNTIL SUB > EXEC-MSG-LEN
          MOVE EXEC-LEVEL-MSG-TEXT (SUB) TO OUT-REC-ELEM (SUB)
        END-PERFORM
        WRITE OUTPUT-RECORD FROM OUT-REC
        WRITE OUTPUT-RECORD FROM MSGV
     END-IF
     WRITE OUTPUT-RECORD FROM MSG14.
```

```
     MOVE EXEC-RESP TO SAVED-RESPONSE.
 LINK-NOT-OK-END.
     EXIT.
*
*  Exit to MVS.
*
 FINISH-PROGRAM.
     WRITE OUTPUT-RECORD FROM MSG31.
     CLOSE PRINTER.
     MOVE SAVED-RESPONSE TO RETURN-CODE.
     STOP RUN.
```

Note: To link-edit the BATCHCLI program, include the CICS410.SDFHEXCI library in the SYSLIB concatenation of the link-edit step.

CICSSRV

```
IDENTIFICATION DIVISION.
PROGRAM-ID. CICSSRV.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
FILE SECTION.

WORKING-STORAGE SECTION.

 Ø1  CARTE-IN.
*    ************************************************************
*    **  Commarea coming from client, either program           **
*    **  CICS with a LINK or the batch client BATCHCLI         **
*    **  Description see in program BATCHCLI                   **
*    ************************************************************
     Ø5  COMMCODE PIC X(1).
     Ø5  COMMBID  PIC X(8).
     Ø5  COMMCNO  PIC 9(3).
     Ø5  COMMRC   PIC X(2).
     Ø5  COMMZONE PIC X(6Ø).

 Ø1  FLAGS.
     Ø2  SWTCHDB2          PIC X  VALUE 'D'.
     Ø2  IX                PIC 9(3).
     Ø2  DSN2EXT1          PIC X(8) VALUE 'DSN2EXT1'.
     Ø2  DSNCCMD           PIC X(8) VALUE 'DSNCCMD '.
     Ø2  STRT-LNG          PIC S9(4) COMP.
     Ø2  PCLOCK            PIC S9(8) COMP.
     Ø2  DCLOCK            PIC X(8).
```

```
     Ø2  TCLOCK            PIC X(8).
     Ø2  RESP              PIC S9(8) COMP.
     Ø2  RESP2             PIC S9(8) COMP.
     Ø2  CONSTAT           PIC S9(8) COMP.
     Ø2  ERR-CODE          PIC 9(2).
*— ØØ = Okay
*— 2Ø = commarea length false
*— 25 = Code card unknown
*— 31 = stop DB2 impossible
*— 97 = not authorized
*— 98 = invalid request
*— 99 = error
 Ø1  WS-FIC-CTRL           PIC X(1Ø).
 Ø1  WS-LG-COMM            PIC 9(Ø4).

*— commarea program length
 77  WLG-COMM              PIC S9(4) COMP VALUE 74.
   Ø1  LIGNE-CSMT.
     Ø2  FILLER               PIC X(8)  VALUE 'CICSSRV '.
     Ø2  FILLER               PIC X(3)  VALUE ' - '.
     Ø2  LC-HEURE           PIC X(12).
     Ø2  FILLER               PIC X(3)  VALUE ' - '.
     Ø2  LC-TRID            PIC X(4).
     Ø2  FILLER               PIC X(3)  VALUE ' - '.
     Ø2  LC-TERM            PIC X(4).
     Ø2  FILLER               PIC X(3)  VALUE ' - '.
     Ø2  LC-BID             PIC X(8).
     Ø2  FILLER               PIC X(3)  VALUE ' - '.
     Ø2  LC-TXT             PIC X(5Ø).

 Ø1  MESSAGES.
     Ø2  MSG2                 PIC X(22) VALUE 'Switch attachment
-     'DB2.'.
     Ø2  MSG9.
     Ø5 MSG9A                PIC X(13) VALUE 'Error card : '.
     Ø5 MSG9B                PIC 9(3).
     Ø5 MSG9C                PIC X(5) VALUE '- RC='.
     Ø5 MSG9D                PIC 9(2).
     Ø5 MSG9E                PIC X(1Ø) VALUE '- EIBRESP='.
     Ø5 MSG9F                PIC X(8).
     Ø5 MSG9G                PIC X(1) VALUE '/'.
     Ø5 MSG9H                PIC X(8).

 LINKAGE SECTION.
 Ø1  DFHCOMMAREA               PIC X(74).

 PROCEDURE DIVISION.
*——Initialization
     MOVE Ø                      TO ERR-CODE.
     MOVE Ø                      TO COMMRC.
```

```
      EXEC CICS ASKTIME ABSTIME(PCLOCK) END-EXEC.
      EXEC CICS FORMATTIME ABSTIME  (PCLOCK)
               DATESEP('/') DDMMYY(DCLOCK)
                TIMESEP TIME(TCLOCK)
               END-EXEC.
      MOVE TCLOCK     TO            LC-HEURE.
      MOVE COMMBID    TO            LC-BID.
      MOVE EIBTRNID   TO            LC-TRID.
      MOVE EIBTRMID   TO            LC-TERM.
      MOVE SPACES     TO            LC-TXT.

*--test commarea length
      EVALUATE EIBCALEN
          WHEN Ø
*--for testing purposes in CICS with START TRANSACTION
               EXEC CICS IGNORE CONDITION
                                 ENDDATA
               END-EXEC
               EXEC CICS RECEIVE
                         INTO   (CARTE-IN)
                         LENGTH (WLG-COMM)
               END-EXEC
          WHEN 74
*--call with a Commarea
               MOVE DFHCOMMAREA      TO CARTE-IN
          WHEN OTHER
               MOVE 2Ø               TO ERR-CODE
               PERFORM ERR-MSG
                  THRU ERR-MSG-END
               PERFORM FIN-PGM
                  THRU FIN-PGM-END
      END-EVALUATE.

      INSPECT CARTE-IN
              CONVERTING
              'abcdefghijklmnopqrstuvwxyz'
              TO
              'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.

      EVALUATE COMMCODE
          WHEN SWTCHDB2
             PERFORM SWTCH-DB2
                THRU SWTCH-DB2-END
          WHEN OTHER
             MOVE 25               TO ERR-CODE
             PERFORM ERR-MSG
                THRU ERR-MSG-END
      END-EVALUATE.

      PERFORM FIN-PGM
```

```
        THRU FIN-PGM-END.

*****************************************************************
*    ************************************************************
*    **       Switch attachment DB2                              **
*    **       prerequisite: replace module DSN2CTxx in           **
*    **       library SDSNLOAD with one pointing to target DB2   **
*    ************************************************************
 SWTCH-DB2.
     MOVE MSG2             TO LC-TXT.
     PERFORM WRITE-MSG THRU WRITE-MSG-END.
*    ** Ask state of DB2 attachment ******************************
     EXEC CICS INQUIRE EXITPROGRAM(DSN2EXT1)
          ENTRYNAME(DSNCCMD) CONNECTST(CONSTAT)
          RESP(RESP) END-EXEC.
     IF RESP = DFHRESP(PGMIDERR) THEN
*    ** PGMIDERR will say : attachment not active  ***************
*    ** Restart attachment  ***************************************
        EXEC CICS LINK PROGRAM('DSN2COM0') END-EXEC
      ELSE
*    ** Stop attachment *******************************************
        EXEC CICS LINK PROGRAM('DSN2COM2')
              RESP(RESP) END-EXEC
        IF RESP = DFHRESP(NORMAL) THEN
*    ** Looping some times until attachment is stopped ***********
*    ** or you decide there is a hanging thread ******************
           PERFORM INQ-DB2 THRU INQ-DB2-END
                   VARYING IX FROM 1 BY 1
                    UNTIL RESP = DFHRESP(PGMIDERR) OR
                   IX > 10
           IF RESP = DFHRESP(PGMIDERR) THEN
              EXEC CICS LINK PROGRAM('DSN2COM0') END-EXEC
           ELSE
             MOVE 31               TO ERR-CODE
             PERFORM ERR-MSG
                 THRU ERR-MSG-END
        ELSE
           IF RESP = DFHRESP(NOTAUTH) THEN
             MOVE 97               TO ERR-CODE
             PERFORM ERR-MSG
                THRU ERR-MSG-END.
 SWTCH-DB2-END.
*****************************************************************
 INQ-DB2.
          EXEC CICS INQUIRE EXITPROGRAM(DSN2EXT1)
                ENTRYNAME(DSNCCMD) CONNECTST(CONSTAT)
                RESP(RESP) END-EXEC.
 INQ-DB2-END.
*****************************************************************
 WRITE-MSG.
```

```
     EXEC CICS WRITEQ TD QUEUE('CSMT') FROM(LIGNE-CSMT)
         LENGTH(100) END-EXEC.
 WRITE-MSG-END.
*****************************************************************
 FIN-PGM.
*-- return transaction
     IF EIBCALEN > 0 THEN
          MOVE ERR-CODE   TO COMMRC
          MOVE CART-IN TO DFHCOMMAREA.
     EXEC CICS RETURN
     END-EXEC.
 FIN-PGM-END.
     EXIT.
*****************************************************************
 ERR-MSG.
     MOVE COMMCNO                  TO MSG9B.
     MOVE ERR-CODE                 TO MSG9D.
     MOVE RESP                     TO MSG9F.
     MOVE RESP2                    TO MSG9H.
     MOVE MSG9                     TO LC-TXT.
     EXEC CICS WRITEQ TD
               QUEUE  ('CSMT')
               FROM   (LIGNE-CSMT)
               LENGTH (100)
     END-EXEC.
 ERR-MSG-END.
     EXIT.
```

Note: the translation option ‘SP’ must be specified. For the definition of transaction EXCI and the generic EXCI connections see CICS documentation or *EXCI – what’s it all about?, CICS Update*, Issue 145, December 1997.

PRE-REQUISITES

The programmable CICS DB2 attachment switch requires CICS 4.1 to use the EXCI interface and INQUIRE for the DB2 attachment. There should be no problems with DB2 Version 3 and any other MVS version, although you will have to look at the attachment program names.

Elke Thamm
System Programmer
Le Foyer Assurances (Luxembourg)

SofTouch Systems has announced Region Generator for CICS, available for MVS and OS/390 operating systems, which enables fast and accurate creation of new CICS regions using user-defined standards and templates.

ReGen4CICS is designed to maintain continuity when building new regions. It uses structured techniques to assist in generating the new regions, automatically updating CSD files and installing resource definitions. Complete audit trails provide the benefit of workflow tracking.

For further information contact:
SofTouch Systems, 1300 South Meridian Avenue, Suite 600, Oklahoma City, OK 73108-1751, USA.
Tel: (405) 947 8080.

* * *

Netscape Communications has added custom extensions to the suite of pre-built extensions already available for CICS systems in its Extension Builder toolkit for Netscape Application Server.

Features in Version 2.1 include wizards for developing custom extensions, support for high volumes of concurrent users, and access to system services of the Application Server such as cacheing, multi-threading, multi-processing, connection pooling, connection management, and streaming. Also, custom extensions can be re-used and ported across different Application Server business applications.

For further information contact:
Netscape Communications, 501 East Middlefield Road, Mountain View, CA 94043, USA.
Tel: (650) 254 1900.

* * *

Unify has announced Version 4.0 of its Vision development environment, which will support CICS, MQSeries, and HTML in System/390 environments. Vision Version 4.0 is designed to integrate old and new applications within a single hybrid environment. A follow-on release, Version 4.b, will include support for specific applications, such as SAP R/3. Both releases support native DB2 connectivity, Java, and existing 3270 applications.

For further information contact:
Unify, 3927 Lennane Drive, Sacramento, CA 95834-1922, USA.
Tel: (800) 455 2405.
Unify, Prestige House, 23-26 High Street, Egham, Surrey, TW20 9DU, UK.
Tel: (01784) 484000.

* * *