August 1998

In this issue

3  Temporary storage queue exit
15  An MQSeries API-exit for CICS
29  An update to the screen viewing utility
35  Managing CICS printers – part 2
48  CICS news
Temporary storage queue exit

At my shop I needed to be able to move transactions from one CICS to another to do load balancing. This might seem like a straightforward task, except that I use semi-permanent temporary storage queues to share some data – not only across transactions but across application systems.

To allow me to do the transaction load shifting required, I decided to implement the temporary storage queue exits to dynamically function ship the temporary storage queue requests to a temporary storage owning region (without the application knowing about it), in this case my TOR.

This is performed in the following way. Firstly, you must write a PLT program to enable the exits XTSEREQ and XTSEREOQC and to associate your user program with them.

The following code is my implementation of this:

```plaintext
******************************************************************
IDENTIFICATION DIVISION.
******************************************************************
PROGRAM-ID.                DPKCS215.
AUTHOR.                    BRUCE BORCHARDT.
INSTALLATION.              KOHLS DEPARTMENT STORES.
DATE-WRITTEN.
DATE-COMPiled.

******************************************************************
* CICS PROGRAM - DPKCS215
* *
* PLT TRANSACTION TO ENABLE REMOTE TEMP STORAGE Q EXIT .*
******************************************************************

ENVIRONMENT DIVISION.
******************************************************************

DATA DIVISION.
******************************************************************

WORKING-STORAGE SECTION.

The XTSEREQ exit allows you to intercept temporary storage API requests before any action has been taken on the request. The XTSEREQC exit allows you to intercept the response after a temporary storage API request has completed.

This PLT transaction must run on all CICSs that you want to do the dynamic function shipping.

Secondly, you must modify the supplied temporary storage queue exits.

This sample is supplied in hardcopy only as DFH$XTSE in Appendix E of the CICS customization guide.

The following is the exit, modified for my use:

*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*--
SHARED_STORAGE   DSECT
SHARED_EYECATCHER DS CL16
SHARED_NAME      DS CL8
SHARED_SYSID     DS CL4
*

* THE FOLLOWING DEFINITIONS ARE FOR PROGRAM WORKING STORAGE.
*

DFHEISTG DSECT
RETCODE      DS XL4                    PROGRAM RETURN CODE
TR_ERROR_N   DS X                      ERROR NUMBER FOR TRACE ENTRY
RESP         DS X                      API RESPONSE
   EJECT ,

*********************************************************************
* PROGRAM REGISTER USAGE :
*    R0 - WORK REGISTER
*    R1 - POINTS TO DFHUEPAR PLIST ON ENTRY
*       WORK REGISTER
*    R2 - DFHUEPAR PARAMETER LIST
*    R3 - CODE BASE REGISTER
*    R4 - <UNUSED>
*    R5 - <UNUSED>
*    R6 - SUBROUTINE LINKAGE REGISTER
*    R7 - ADDRESS OF TS QUEUE NAME FROM COMMAND PLIST
*    R8 - COMMAND PARAMETER LIST UEPCLPS
*    R9 - ADDRESS OF TABLE_ENTRY IN TS_ROUTING_TABLE
*    R10- <UNUSED>
*    R11- EIB REGISTER
*    R12- WORK REGISTER
*    R13- DFHEISTG FOR API CALLS
*       KERNEL STACK FOR XPI CALLS
*    R14- WORK REGISTER
*    R15- WORK REGISTER

*********************************************************************
EJECT ,

*********************************************************************
* DPKCS22Ø - MAIN ROUTINE
*    THIS IS THE ENTRY POINT FOR THE EXIT PROGRAM. CONTROL IS PASSED
*    TO THE TS_REQUEST OR TS_REQUEST_COMPLETE ROUTINES DEPENDING
*    ON WHETHER THE EXIT WAS INVOKED AT THE XTSEREQ OR XTSEREQC EXIT
*    POINTS
*
*    REGISTERS:
*    R1 = UEPAR PLIST (SET ON ENTRY)
*       WORK REGISTER
*    R2 = UEPAR PLIST
*    R3 = PROGRAM BASE REGISTER (SET BY DFHEIENT)
*    R6 = LINKAGE REGISTER
*    R11= EIB REGISTER
*    R13= EISTG REGISTER (SET BY DFHEIENT)
* R15= WORK REGISTER
* USER EXIT RETURN CODE
 *
* LOGIC:
* DPKCS22Ø:
* EXEC INTERFACE ENTRY
* ADDRESS DFHUEPAR PLIST
* SET OK RETURN CODE
* ADDRESS THE EIB
* TRACE ENTRY
* SELECT EXITID
* WHEN(XTSEREQ) THEN CALL TS_REQUEST
* WHEN(XTSEREQC) THEN CALL TS_REQUEST_COMPLETE
* OTHERWISE CALL ERROR(INVALID_EXIT)
* END SELECT
* TRACE EXIT
* SET EXIT RETURN CODE
* RETURN
*********************************************************************
DPKCS22Ø DFHEIENT
DPKCS22Ø AMODE 31
DPKCS22Ø RMODE ANY
LR R2,R1                DFHUEPAR PLIST PROVIDED BY CALLER
    USING DFHUEPAR,R2     USE R2 TO ADDRESS UEPAR PLIST
 *
LA R15,UERCNORM        SET OK RESPONSE
ST R15,RETCODE           IN WORKING STORAGE
 *
EXEC CICS ADDRESS EIB(R11)
    USING DFHEIBLK,R11
* EXEC CICS WRITEQ TD QUEUE('CSSL') FROM(MESSAGE1) LENGTH(13)
 *
BAL R6,TRACE_ENTRY      TRACE PROGRAM ENTRY
 *
L  R1,UEPEXN           ADDRESS OF THE 1 BYTE EXIT ID
CLI Ø(R1),XTSEREQ       IS THIS XTSEREQ EXIT?
BE TS_REQUEST          ..YES BRANCH TO ROUTINE
CLI Ø(R1),XTSEREQC      IS THIS XTSEREQC EXIT?
BE TS_REQUEST_COMPLETE .. YES BRANCH TO ROUTINE
B  ERROR1              OTHERWISE BRANCH TO ERROR ROUTINE
 *
RETURN    DS ØH                  RETURN POINT
BAL R6,TRACE_EXIT       TRACE PROGRAM EXIT
 *
L R15,RETCODE         FETCH RETURN CODE
DFHEIRET RCREG=15     RETURN TO CICS
EJECT .
*====================================================================
* TS_REQUEST - INVOKED AT XTSEREQ EXIT POINT
DETERMINE THE TS QUEUE NAME AND SCAN THE TS_ROUTING_TABLE FOR
A MATCH. IF AN ENTRY EXISTS IN THE TABLE, THEN CHECK THE ACTION
FIELD AND CALL THE ROUTE_REQUEST OR LOCAL_REQUEST ROUTINES.

THE TS_ROUTING_TABLE IS MADE UP OF ENTRIES WITH THE FOLLOWING
STRUCTURE:

TABLE_ENTRY:

| ENTRY_NAME | NEW_NAME | QOR_SYSID | ACTION | *FILLER* |
| CHAR 8     | CHAR 8   | CHAR 4    | BIN 1   | CHAR 3   |

LAST ENTRY IS INDICATED BY SPECIAL TS_QUEUE NAME

REGISTERS:
R1 = WORK REGISTER
R7 = SET TO THE TS QUEUE NAME
R8 = COMMAND PARAMETER LIST (CLPS)
R9 = POINTS TO THE NEXT ENTRY IN THE TS_ROUTING_TABLE
R15= WORK REGISTER

LOGIC:
TS_REQUEST:
* IF CALLED RECURSIVELY THEN
  CALL ERROR(RECURSIVE_CALL1)
* ELSE
  IF THE COMMAND GROUP CODE IS NOT A TS REQUEST THEN
    CALL ERROR(INVALID_GROUP_CODE1)
  ELSE
    CLEAR THE UEPTQTOK
    ADDRESS THE COMMAND PLIST UEPCLPS
    FETCH TSQ_NAME
    FETCH START OF TS_ROUTING_TABLE
    CHECK_NEXT_ENTRY:
    GET THE NEXT TABLE ENTRY
    SELECT (ENTRY_NAME)
    WHEN (LAST_ENTRY) CALL ENTRY_NOT_FOUND
    WHEN (TSQ_NAME)
    SELECT (ENTRY_ACTION)
    WHEN (ROUTE) CALL ROUTE_REQUEST
    WHEN (LOCAL) CALL LOCAL_REQUEST
    OTHERWISE CALL ERROR(INVALID_TABLE_ACTION)
  END SELECT
  OTHERWISE
    GOTO CHECK_NEXT_ENTRY
  END SELECT
END IF
END IF
RETURN
TS_REQUEST DS ØH
* CHECK FOR POSSIBLE RECURSION
L R1,UEPRECUR ADDRESS OF RECURSIVE COUNT
LH R1,Ø(R1) FETCH COUNT
LTR R1,R1 HAS EXIT BEEN INVOKED RECURSIVELY?
BNZ ERROR2 YES BRANCH TO ERROR ROUTINE
*
* EXTRACT POINTER TO THE EID AND TS QUEUE NAME FROM CLPS
L R8,UEPCLPS FETCH ADDRESS OF COMMAND PLIST
USING TS_ADDR_LIST,R8 USE R8 TO ADDRESS CLPS
L R1,TS_ADDRØ ADDRESS THE EID..
L R7,TS_ADDR1 FETCH ADDRESS OF TS QUEUE
CLC =CL5'TCPIP',Ø(R7) CHECK FOR TCPIP QS
BZ RETURN YES, DO NOT ROUTE
CLC =CL2'DR',Ø(R7) CHECK FOR DRS QS
BZ RETURN YES, DO NOT ROUTE
DROP R8 DROP ADDRESSABILITY TO CLPS
*
* CHECK THAT THE COMMAND GROUP CODE CORRESPONDS TO A TS REQUEST
USING TS_EID,R1 ..WITH REGISTER 1
CLI TS_GROUP,TS_TEMPSTOR_GROUP IS THIS A TS REQUEST?
BNE ERROR3 ..NO BRANCH TO ERROR ROUTINE
DROP R1 DROP ADDRESSABILITY TO EID
*
* CLEAR THE TS REQUEST TOKEN
L R1,UEPTQTOK FETCH ADDRESS OF TOKEN
XC Ø(4,R1),Ø(R1) CLEAR TOKEN FOR XTSEREQC
B ENTRY_NOT_FOUND ..YES TAKE DEFAULT ROUTING ACTION
*====================================================================
* TS_REQUEST_COMPLETE - INVOKED AT XTSEREQC EXIT POINT
* FREE ANY SHARED STORAGE THAT WAS ACQUIRED DURING PREVIOUS
* INVOCATION AT XTSEREQ
*
* REGISTERS:
* R1 = WORK REGISTER
* R6 = LINKAGE REGISTER
* R8 = COMMAND PARAMETER LIST (CLPS)
*
* LOGIC:
* TS_REQUEST_COMPLETE:
* IF CALLED RECURSIVELY THEN
* CALL ERROR(RECURSIVE_CALL2)
* ELSE
* IF THE COMMAND GROUP CODE IS NOT A TS REQUEST THEN
* CALL ERROR(INVALID_GROUP_CODE2)
* ELSE
* IF UEPTQTOK->TOKEN ,= Ø THEN CALL FREEMAIN_SHARED_PLIST
* END IF
ENTRY_NOT_FOUND - NO ENTRY WAS FOUND IN THE TS_ROUTING_TABLE
NO ENTRY FOUND IN ROUTING TABLE FOR THIS TS QUEUE NAME. IN THE SAMPLE PROGRAM, ALL SUCH REQUESTS ARE ROUTED.

REGISTERS:
R1 = WORK REGISTER
R6 = LINK REGISTER
R8 = COMMAND PARAMETER LIST (CLPS)
R12= WORK REGISTER (SHARED_STORAGE)

LOGIC:
ENTRY_NOT_FOUND:
CALL GETMAIN_SHARED
COPY DEFAULT_SYSID INTO SHARED STORAGE
ADDRESS THE COMMAND PLIST
UPDATE ADDR7 TO POINT TO THE ADDRESS OF THE DEFAULT SYSID
SET THE SYSID EXISTENCE BIT IN THE EID
SET THE HI-ORDER BIT IN LAST ADDRESS IN CLPS
RETURN
ENTRY_NOT_FOUND DS ØH
BAL R6,GETMAIN_SHARED GETMAIN SHARED STORAGE
L R12,UEPTQTOK FETCH ADDRESS OF TOKEN
L R12,Ø(R12) FETCH SHARED STORAGE ADDRESS
USING SHARED_STORAGE,R12 ADDRESS USING R12
*
* UPDATE THE SYSID IN CLPS
* EXEC CICS WRITEQ TD QUEUE('CSSL') FROM(MESSAGE2) LENGTH(14)
MVC SHARED_SYSID,DEFAULT_SYSID COPY SYSID TO SHARED STORAG
L R8,UEPCLPS ADDRESS THE CLPS..
USING TS_ADDR_LIST,R8 ..WITH REGISTER 8
L R1,TS_ADDRØ ADDRESS THE EID..
USING TS_EID,R1 ..WITH REGISTER 1
OI TS_BITS1,TS_SYSID_V INDICATE SYSID NOW PRESENT IN CLPS
DROP R1 DROP ADDRESSABILITY TO EID
LA R1,SHARED_SYSID FETCH ADDRESS OF THE NEW SYSID
ST R1,TS_ADDR7 STORE ADDRESS IN TS_ADDR7
OI TS_ADDR7,X'8Ø' INDICATE SYSID IS END OF PLIST
*
* CLEAR HI-ORDER BITS IN ARGS 1 TO 5
TM TS_ADDR1,X'8Ø'
BNE NOTFND1
NI TS_ADDR1,X'7F' INDICATE NOT LAST PARAMETER IN CLP
NOTFND1 DS ØH
TM TS_ADDR2,X'8Ø'
BNE NOTFND2
NI TS_ADDR2,X'7F' INDICATE NOT LAST PARAMETER IN CLP
NOTFND2 DS ØH
TM TS_ADDR3,X'8Ø'
BNE NOTFND3
NI TS_ADDR3,X'7F' INDICATE NOT LAST PARAMETER IN CLP
NOTFND3 DS ØH
TM TS_ADDR4,X'8Ø'
BNE NOTFND4
NI TS_ADDR4,X'7F' INDICATE NOT LAST PARAMETER IN CLP
NOTFND4 DS ØH
TM TS_ADDR5,X'8Ø'
BNE NOTFND5
NI TS_ADDR5,X'7F' INDICATE NOT LAST PARAMETER IN CLP
NOTFND5 DS ØH
B RETURN
DROP R8 DROP TS_ADDR_LIST
DROP R12 DROP SHARED_STORAGE
EJECT ,
*
*====================================================================
* GETMAIN_SHARED - OBTAIN SHARED STORAGE
* WE CANNOT USE TRANSACTION STORAGE TO PASS INFORMATION IN THE
* COMMAND PARAMETER LIST SINCE THIS IS VOLATILE AND WILL BE
* RELEASED WHEN THE EXIT PROGRAM RETURNS TO CICS.
* WE MUST OBTAIN SHARED STORAGE HERE, AND FREE IT AT THE
* TS REQUEST COMPLETE EXIT XTSEREQC

* REGISTERS:
* R0 = USED BY EXEC CICS CALL
* R1 = USED BY EXEC CICS CALL
* WORK REGISTER
* R6 = LINK REGISTER - RETURN ADDRESS
* R11= EIB REGISTER  (SET ON ENTRY)
* R12= WORK REGISTER
* R14= USED BY EXEC CICS CALL
* R15= USED BY EXEC CICS CALL

* LOGIC:
* GETMAIN_SHARED:
* EXEC CICS GETMAIN LENGTH(32) SET(UEPTQTOK) SHARED RESP(RESP)
* IF RESP ,= OK THEN
* CALL ERROR(GETMAIN_FAILED)
* ELSE
* ADDRESS SHARED STORAGE
* SET EYECATCHER 'XTSEREQ STORAGE'
* END IF
* RETURN

*====================================================================
GETMAIN_SHARED DS ØH
L   R12,UEPTQTOK     FETCH ADDRESS OF TOKEN
L   R12,Ø(R12)       FETCH SHARED STORAGE ANCHOR
LTR R12,R12          IS THE STORAGE ALREADY PRESENT
BNZR R6               ..YES RETURN
EXEC CICS GETMAIN LENGTH(32) SET(R12) SHARED -
                INITIMG(X'ØØ') RESP(RESP)
CLC   RESP,DFHRESP(NORMAL)   GETMAIN WORKED OK?
BNE    ERROR7            ..NO GOTO ERROR ROUTINE
L   R1,UEPTQTOK        FETCH ADDRESS OF TOKEN
ST   R12,Ø(R1)         SAVE ADDRESS OF STORAGE
 USING SHARED_STORAGE,R12
MVC   SHARED_EYECATCHER,EYE_CATCHER SET EYECATCHER
DROP  R12              DROP R12
BR    R6               RETURN TO CALLER
EJECT .

*====================================================================

* FREEMAIN_SHARED - FREE SHARED STORAGE
* FREE THE SHARED STORAGE ASSOCIATED WITH THIS COMMAND.

* REGISTERS:
* R0 = USED BY EXEC CICS CALL
* R1 = USED BY EXEC CICS CALL
* R6 = LINK REGISTER - RETURN ADDRESS
* R11= EIB REGISTER (SET ON ENTRY)
* R12= WORK REGISTER
* R14= USED BY EXEC CICS CALL
* R15= USED BY EXEC CICS CALL
*
* LOGIC:
* FREEMAIN_SHARED:
* ADDRESS SHARED STORAGE
* IF EYECATCHER ,= 'XTSEREQ STORAGE' THEN
  CALL ERROR(FREEMAIN_LOGIC_ERROR)
* ELSE
  EXEC CICS FREEMAIN DATAPointer(UEPTQTOK) RESP(RESP)
  IF RESP ,= OK THEN
    CALL ERROR(FREEMAIN_FAILED)
  END IF
* END IF
* RETURN
*====================================================================
FREEMAIN_SHARED DS ØH
L R12,UEPTQTOK FETCH TOKEN ADDRESS
L R12,Ø(R12) ADDRESS SHARED STORAGE ADDRESS
USING SHARED_STORAGE,R12 ..USING R12
CLC SHARED_EYECATCHER,EYE_CATCHER IS THIS OUR STORAGE?
BNE ERROR8 ..NO GOTO ERROR ROUTINE
DROP R12 DROP R12
EXEC CICS FREEMAIN DATAPointer(R12) RESP(RESP)
CLC RESP,DFHRESP(NORMAL) FREEMAIN WORKED OK?
BNE ERROR9 ..NO GOTO ERROR ROUTINE
L R12,UEPTQTOK FETCH TOKEN ADDRESS
XC Ø(4,R12),Ø(R12) CLEAR TOKEN ADDRESS
BR R6 RETURN TO CALLER
EJECT ,
*
*====================================================================
* TRACE ROUTINES
* ISSUE A TRACE XPI CALL
* *
* REGISTERS:
* R0 = USED BY XPI CALL
* R1 = DFHTRPT PLIST
* R6 = LINK REGISTER - RETURN ADDRESS
* R12= WORK REGISTER
* R13= EISTG REGISTER (SET BY DFHEIENT)
* KERNEL STACK ENTRY
* R14= USED BY XPI CALL
* R15= USED BY XPI CALL
*====================================================================
USING DFHTRPT_ARG,R1
TRACE_ENTRY DS ØH
  L    R1,UEPXSTOR     PREPARE FOR XPI CALL
  DFHTRPTX CLEAR,
       POINT_ID(TR_ENTRY)
  B    ISSUE_TRACE
TRACE_EXIT DS ØH
  L    R1,UEPXSTOR     PREPARE FOR XPI CALL
  DFHTRPTX CLEAR,
       POINT_ID(TR_EXIT)
  B    ISSUE_TRACE
TRACE_ERROR DS ØH
  L    R1,UEPXSTOR     PREPARE FOR XPI CALL
  DFHTRPTX CLEAR,
       POINT_ID(TR_ERROR),
       DATA1(TR_ERROR_N,1)
  BAL   R6,ISSUE_TRACE
  B    RETURN

*  *---------------------------------------------------------------------
*  ISSUE THE TRACE XPI CALL
*  *---------------------------------------------------------------------

ISSUE_TRACE DS ØH
  L    R8,UEPTRACE     ADDRESS OF TRACE FLAG
  TM    Ø(R8),UEPTRON  IS TRACE ON?
  BZ    NO_TRACE       NO - DO NOT ISSUE TRACE THEN
  LR    R12,R13       SAVE R13 ROUND XPI CALL
  L    R13,UEPSTACK
  DFHTRPTX CALL,
       IN,  
       FUNCTION(TRACE_PUT),
       POINT_ID(*),
       OUT,  
       RESPONSE(*),
       REASON(*)
  LR    R13,R12       RESTORE R13 (DFHEISTG)
NO_TRACE DS ØH
  BR    R6            RETURN TO CALLER
DROP  R1

*  *---------------------------------------------------------------------
*  ERRORN
*  ERROR HAS OCCURRED DURING PROCESSING
*  ISSUE A TRACE POINT AND RETURN TO THE CICS
*  *---------------------------------------------------------------------

ERROR1 DS ØH
  MVI   TR_ERROR_N,1
  B    TRACE_ERROR
ERROR2 DS ØH
  MVI   TR_ERROR_N,2
B     TRACE_ERROR
ERROR3 DS 0H
MVI TR_ERROR_N,3
B     TRACE_ERROR
ERROR4 DS 0H
MVI TR_ERROR_N,4
B     TRACE_ERROR
ERROR5 DS 0H
MVI TR_ERROR_N,5
B     TRACE_ERROR
ERROR6 DS 0H
MVI TR_ERROR_N,6
B     TRACE_ERROR
ERROR7 DS 0H
MVI TR_ERROR_N,7
B     TRACE_ERROR
ERROR8 DS 0H
MVI TR_ERROR_N,7
B     TRACE_ERROR
ERROR9 DS 0H
MVI TR_ERROR_N,7
B     TRACE_ERROR
EJECT ,
DROP R2                      DROP DFHUEPAR
DROP R11                     DROP EIB
LTORG ,
*****************************************************************************
* CONSTANTS
*****************************************************************************

DS 0D
EYE_CATCHER DC CL16'XTSEREQ STORAGE '
DEFAULT_SYSID DC CL4'CIP1'
LOCAL     EQU X'01'
ROUTE     EQU X'02'
MESSAGE1  DC CL13'ENTERING EXIT'
MESSAGE2  DC CL14'CHANGING SYSID'
*
* TRACE POINT IDS
TR_ENTRY   DC XL2'120'
TR_EXIT    DC XL2'121'
TR_ERROR   DC XL2'122'
*
END DPKCS22Ø

Bruce Borchardt
Senior Systems Programmer (USA) © Xephon 1998
An MQSeries API-exit for CICS

During the development of CICS applications that use MQSeries, our users and application programmers often asked me to check their application queues for messages. I was able to do this using the TSO interface that comes with MQSeries, but, in some cases the queues were empty, and it was impossible to determine whether:

- There were no messages to process.
- The messages had been processed already.
- MQSeries had lost the messages.

Although I have not encountered MQSeries losing messages, application staff have claimed the loss of messages on numerous occasions.

On other occasions, I was asked to investigate messages containing specific application data (e.g., order number) that may have been processed in the past, and if so, to find when these messages were processed.

I found answering these, and other questions, very difficult because MQSeries does not maintain a log of application calls to MQSeries and the data passed to, or received from, MQSeries. Dumping and searching the MQSeries system logs is not viable, not only because it is so time-consuming, but because it helps only in locating persistent messages.

To help our developers test their CICS programs, and to answer their questions, I needed to establish an exit point that would make it possible to log application calls, passed parameters, and application data to MQSeries, as well as completion and result codes.

In the MQSeries manuals, I found that CSQCAPX would serve as this exit – because it is invoked in CICS once before and once after an MQSeries call is executed. The IBM sample program, which can be found in the SCSQASMS library, was of no practical use for my purposes because it logs very little data, although it did show me how to use the exit.
I modified CSQCAPX to capture and record much more detailed information such as CICS region, date, MQ data areas information such as object names and handles, message-ids, correlation-id, and application data extracted from the messages themselves.

An overview of the logged data depending on the MQSeries API call is shown in Figure 1.

Object handle, in most cases message-id and correlation-id, will contain non-printable characters, so these are logged in character and dump format.

I did not pay much attention to MQINQ and MQSET calls, because they are not used in our applications at present.

For application data, the first 60 bytes are logged. These contain relevant information, such as order number and date, enabling application developers and users to identify the message quickly.

If the result code after an MQSeries call has a non-zero value, an additional log record containing the result code text will be written. This gives timely error determination because you are not required to dig for this text in the soft or hardcopy manuals.

Each record logged also contains processed date and time, CICS application-id, transaction name, and transaction number. Macro BGFILMSG, which I use in all my CICS programs to write log and statistic messages, is supplied with the program source. This fills up a predefined message text with data using placeholders (%). If not specified in the macro call, the length of the variable is used for substitution.

The log messages are written to a CICS transient data queue. Change the queue name to one of your favourite destinations, or use CKQQ or CKMQ, which come with MQSeries.

Figure 2 shows an example of the logging output from CSQCAPX. Explanations of the log messages are as follows:

1. (a) – Before opening an object, the object name is logged.
   (b) – After opening, codes and the returned object handle are logged.
<table>
<thead>
<tr>
<th>MQ-Call</th>
<th>Exitpoint</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MQOPEN</td>
<td>before</td>
<td>Object name out of object descriptor</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Object handle (assignment of subsequent calls to object name)</td>
</tr>
<tr>
<td>MQCLOSE</td>
<td>before</td>
<td>Object handle</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
</tr>
<tr>
<td>MQGET</td>
<td>before</td>
<td>Object handle</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC only if completion code = 0 (OK)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correlation-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Length of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message data (up to 60 bytes)</td>
</tr>
<tr>
<td>MQPUT/PUT1</td>
<td>before</td>
<td>Object handle if MQPUT, Object name if MQPUT1</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>Length of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message data (up to 60 bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Message-id</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correlation-id</td>
</tr>
<tr>
<td>MQINQ/MQSET</td>
<td>before</td>
<td>Object handle</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>CC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
</tr>
</tbody>
</table>

*Figure 1: MQ API – calls and logged data*
The common header of all log messages takes the form:

Message-id, date, time, CICS applid, transaction name, transaction number

Example:

CSQCAP0071 08/05/1998 15:20:08 SRVCICS 5998 00007130

Data part of log messages:

(1a) BEFORE MQOPEN XEPHON.MQPUT.QUEUE
(1b) AFTER MQOPEN CC: 0000 RC: 0000 HOBJ: 00000001
(2a) BEFORE MQOPEN XEPHON.MQGET.QUEUE
(2b) AFTER MQOPEN CC: 0000 RC: 0000 HOBJ: 00000002
(3a) BEFORE MQGET 00000002
(3b) AFTER MQGET CC: 0000 RC: 0000 HOBJ: 00000002
(3b) MSGID: CSQ MQS0 M n P
(3b) CORID: 000000000000000000000000000000000000000000000000000
(3b) LEN: 00000080
(3b) DATA: THIS DATA SHOULD BE READ BY MQGET
(4a) BEFORE MQGET 00000002
(4b) AFTER MQGET CC: 0002 RC: 2033 HOBJ: 00000002
(4b) RC IS: NO_MSG_AVAILABLE
(5a) BEFORE MQPUT 00000001
(5a) LEN: 00000080
(5a) DATA: JUST A TESTMESSAGE FOR MQPUT
(5b) AFTER MQPUT CC: 0000 RC: 0000 HOBJ: 00000001
(5b) MSGID: CSQ MQS0 MM
(5b) CORID: 000000000000000000000000000000000000000000000000000
(5b) LEN: 00000080
(5b) DATA: JUST A TESTMESSAGE FOR MQPUT
(6a) BEFORE MQPUT1 XEPHON.MQPUT1.QUEUE
(6a) LEN: 00000080
(6a) DATA: JUST A TESTMESSAGE FOR MQPUT
(6b) AFTER MQPUT1 CC: 0000 RC: 0000 HOBJ:
(6b) MSGID: CSQ MQS0 MM X
(6b) CORID: 000000000000000000000000000000000000000000000000000
(7) BEFORE MQCLOSE 00000001
(7) AFTER MQCLOSE CC: 0000 RC: 0000 HOBJ: 00000001
(7) BEFORE MQCLOSE 00000002
(7) AFTER MQCLOSE CC: 0000 RC: 0000 HOBJ: 00000002

Figure 2: Logging output from CSQCAPX
2 The same as 1(a) and 1(b), except that a different object name is passed and a different object handle returned.

3 (a) – Before MQGET, the object handle is shown to identify the object name from the preceding MQOPEN command.
   (b) – After a successful MQGET, message-id, correlation-id, data length, and data are logged.

4 The same as 3, except that the MQGET was unsuccessful. The result code 2033 and the result code text NO_MSG_AVAILABLE is logged.

5 (a) – Before MQPUT, the object handle to identify the object name from the preceding MQOPEN, data length, and 60 bytes of data are logged.
   (b) – After a successful MQPUT, message-id and correlation-id are logged.

6 The same as 5(a) and 5(b), except for the MQPUT1 command, which is why the object name instead of the object handle is logged.

7 Close of queues, object handles are logged.

With this information directly at hand, I am able to answer all questions relating to the processing of MQSeries messages within the boundaries of CICS.

USAGE NOTES

The program is assembled with HLASM Release 2.0. Use your standard procedure for CICS Assembler programs and include the SCSQMACS library in the STEPLIB concatenation. The link-edit parameters are AMODE 31 RMODE ANY.

This exit has been tested and currently runs in a CICS/ESA Version 4.1 and MQSeries for MVS/ESA Version 1.1.4 environment. Recompilation should be enough for different CICS versions. In the case of a higher MQSeries version, the parameters in the MQ API calls and the used fields in the MQSeries data areas should be checked in
relation to length and current offset.

When CICS connects to the local queue manager, the exit, if in the RPL concatenation, is automatically enabled – unless there is a disabled program definition for CSQCAPX in the CICS system definition file. If you use CICS program auto-install, it is not enough to omit the program definition – the exit will be auto-installed and enabled if it is found in the RPL concatenation. If you are using the CICS storage protection feature, the exit must be defined with EXECKEY(CICS).

The exit itself can be switched on and off at any time during CICS execution using the IBM-supplied CKQC transaction (connection/modify/enable-disable exit). For example, you should disable the exit before issuing a CEMT SET NEWCOPY.

From my sample you should get a good idea and understanding of the benefits of such an exit, and how to implement and use CSQCAPX. You should be able to modify the exit to fit your particular environment and needs.

You should carefully consider the use of CSQCAPX and the amount of data logged – excessive logging will ultimately degrade the performance of the MQI. I strongly recommend that you read Chapter 15 in *MQSeries Application Programming Guide* before using CSQAPX.

If you have an excessive number of messages to process, it may be useful to reduce the number of log messages, or to concentrate on the most important calls. You may also consider only enabling the exit in error situations or for new or selective applications.

Thanks to Jon Herbert for his help in setting up this article.

*  
  MACRO
  BGFILMSG &MSG,&MSGTYPE,&F1,&L1,&F2,&L2,&F3,&L3,&F4,&L4,&F5,&L5,&F6,&L6 
  .*
  .* FILLS A MESSAGETEXT (SEE MSGXXX CONSTANT DEFINITIONS) WITH UP 
  .* TO 6 DATA FIELDS 
  .* 
  .* EXAMPLE:
  .*  
  .*  MSGØØ1 DC C'ØØ1TRANSACTION %... ABENDED WITH CODE %... '
LET'S ASSUME:

ABENDCODE IS CL8 '1234ASRA'
TXNAME IS CL4 'XPHN'
CSECT (PROGRAM) NAME IS XEPHON

BGFILEMSG MSG001,I,TXNAME,,ABENDCODE+4,4

I - INFORMATION
W - WARNING
E - ERROR
C - CRITICAL

WITH APPLID AND DATE/TIME FILLED COMPLETE MESSAGE WILL READ:

XEPHON001I DATE TIME APPLID TRANSACTION XPHN ABENDED WITH CODE ASRA

AIF ('&MSG' EQ '').NOMSG
AIF ('&MSGTYPE' EQ '').NOTYPE
* MOVE PROGRAM NAME
MVC BM_MPROGNAME(L'BMI_MPROGNAME),'=C'&SYSECT'
* MOVE MESSAGE NUMBER
MVC BM_MNUM(L'BMI_MNUM),&MSG
* CLEAR MESSAGE-TEXT
MVI BM_MTEXT,C'
MVC BM_MTEXT+1(L'BMI_MTEXT-1),BM_MTEXT
* MOVE MESSAGE-TEXT
MVC BM_MTEXT(L'&MSG-L'BMI_MNUM),&MSG+L'BMI_MNUM

* MOVE MESSAGE-IDENTIFIER (I,W,E,C)
AIF ('&MSGTYPE' EQ 'E').MSGERR
AIF ('&MSGTYPE' EQ 'I').MSGINF
AIF ('&MSGTYPE' EQ 'W').MSGWAR
AIF ('&MSGTYPE' EQ 'C').MSGCRI
AGO .BADTYPE

.MSGERR ANOP
MVI BM_MID,BM_ERROR
AGO .FIELD1

.MSGINF ANOP
MVI BM_MID,BM_INFORMATION
AGO .FIELD1
MSGCRI ANOP
MVI BM_MID,BM_CRITICAL
AGO .FIELD1

MSGWAR ANOP
MVI BM_MID,BM_WARNING
AGO .FIELD1

FIELD1 ANOP
AIF ('&F1' EQ '').END

* SAVE REGISTERS USED BY TRT IN CASE THEY ARE USED
ST R1,BM_MSG_SAVER1
ST R2,BM_MSG_SAVER2

* INSERT FIRST FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L1' EQ '').LENGTH1
MVC Ø(&L1,R1),&F1 MOVE FIELD TO MESSAGE
AGO .FIELD2

LENGTH1 ANOP
MVC Ø(L'&F1,R1),&F1 MOVE FIELD TO MESSAGE
AGO .FIELD2

FIELD2 ANOP
AIF ('&F2' EQ '').ENDOFFIELDS

* INSERT SECOND FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L2' EQ '').LENGTH2
MVC Ø(&L2,R1),&F2 MOVE FIELD TO MESSAGE
AGO .FIELD3

LENGTH2 ANOP
MVC Ø(L'&F2,R1),&F2 MOVE FIELD TO MESSAGE
AGO .FIELD3

FIELD3 ANOP
AIF ('&F3' EQ '').ENDOFFIELDS

* INSERT THIRD FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L3' EQ '').LENGTH3
MVC Ø(&L3,R1),&F3 MOVE FIELD TO MESSAGE
AGO .FIELD4

LENGTH3 ANOP
MVC Ø(L'&F3,R1),&F3 MOVE FIELD TO MESSAGE
AGO .FIELD4

FIELD4 ANOP
AIF ('&F4' EQ '').ENDOFFIELDS

* INSERT FOURTH FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L4' EQ '').LENGTH4
MVC Ø(&L4,R1),&F4 MOVE FIELD TO MESSAGE
AGO .FIELD5

LENGTH4 ANOP
MVC Ø(L'&F4,R1),&F4 MOVE FIELD TO MESSAGE
AGO .FIELD5

FIELD5 ANOP
AIF ('&F5' EQ '').ENDOFFIELDS
* INSERT FIFTH FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L5' EQ '').LENGTH5
MVC Ø(&L5,R1),&F5 MOVE FIELD TO MESSAGE
AGO .FIELD6

.LENGTH5 ANOP
MVC Ø(L'&F5,R1),&F5 MOVE FIELD TO MESSAGE
AGO .FIELD6

.FIELD6 ANOP
AIF ('&F6' EQ '').ENDOFFIELDS
* INSERT SIXTH FIELD
TRT BM_MTEXT,BM_TRTAB
AIF ('&L6' EQ '').LENGTH6
MVC Ø(&L6,R1),&F6 MOVE FIELD TO MESSAGE
AGO .ENDOFFIELDS

.LENGTH6 ANOP
MVC Ø(L'&F6,R1),&F6 MOVE FIELD TO MESSAGE
AGO .ENDOFFIELDS

* .ENDOFFIELDS ANOP
* RESTORE REGISTERS
  L R1,BM_MSG_SAVER1
  L R2,BM_MSG_SAVER2
.END ANOP
MEXIT

* MACRO ERROR EXITS
.NOMSG MNOTE 12,' *** ERROR *** MESSAGEFIELD NOT SPECIFIED'
MEXIT
.NOTYPE MNOTE 12,' *** ERROR *** MESSAGETYPE NOT SPECIFIED'
MEXIT
.BADTYPE MNOTE 12,' *** ERROR *** WRONG MESSAGETYPE SPECIFIED'
MEXIT
MEND
MEXIT
MEND
EJECT

* TITLE 'CICS - MQSERIES API CROSSING EXIT'
PUNCH ' MODE AMODE(31),RMODE(ANY)'
* R8   BASE REGISTER
* R9   BASE REGISTER
* R10  MQXP
* R11  DFHEIBLK
* R12  BASE
* R13  DFHEISTG
* R14
* R15
*
  DFHREGS      REGISTER EQUATES
*MQXP          CMQXPA LIST=NO  EXIT PARM BLOCK STRUCTURE
*
* TRANSACTION STORAGE
*
DFHEISTG  DSECT
*
   DS  0F
BM_MSG    DS  0CL128  AREA FOR CICS LOG MESSAGES
*
BM_MHEADER DS  0CL10  MESSAGE-ID
BM_MPROGNAME DS  CL6   - PROGRAMNAME
BM_MNUM    DS  CL3    - MESSAGENUMBER
BM_MID     DS  CL1    - MESSAGEIDENTIFIER
BM_INFORMATION EQU C'I'   - I INFORMATION
BM_WARNING EQU C'W'   - W WARNING
BM_ERROR   EQU C'E'   - E ERROR
BM_CRITICAL EQU C'C'  - C CRITICAL
   DS  CL1
BM_MDATE   DS  CL10   DATE DD/MM/YYYY
   DS  CL1
BM_MTIME   DS  CL8    TIME HH:MM:SS
   DS  CL1
BM_MAPPLID DS  CL8    APPLICATION ID
   DS  CL1
BM_MTEXT   DS  CL88   88 BYTES MSG-TEXT
*
* WORK FIELDS....
   DS  0D
WRKDWORD   DS  D      WORK DOUBLE WORD
ABSTIME    DS  PL8    ASKTIME COMMAND
WORKFLD1   DS  CL8    USED FOR DATA CONVERSION
LENSAVE    DS  F      SAVE LENGTH OF DATA
BM_MSG_SAVER1 DS  F    SAVEAREA FOR BGFILMSG MACRO
BM_MSG_SAVER2 DS  F    SAVEAREA FOR BGFILMSG MACRO
COMPTR     DS  F      SAVE COMMAREA ADDRESS
*
TASKNUM    DS  CL8    CICS TASK NUMBER
OPCODE     DS  CL8    MQ OPERATION CODE FOR LOG
OBJECTNAME DS  CL48   MQ OBJECTNAME FOR LOG
BUFFER     DS  CL60   SAVE DATA FROM PUT/GET
CC C DS CL4 COMPLETION CODE FOR LOG
RC C DS CL4 RESULT CODE FOR LOG
WORKI DS CL8 SOME CONVERSION
HOBJ DS CL8 OBJECT HANDLE FOR LOG
DS CL1 SCRATCH
*
AFTERCALL DS CL1 FLAG TO REMEMBER EXITREASON
BEFORECALL DS CL1 FLAG TO REMEMBER EXITREASON
MQPUT DS CL1 FLAG FOR MQPUT / MQPUT1
LENGTH DS CL8 LENGTH OF GET/PUT DATA
MSGID DS CL24 MSGID
MSGIDDUMP DS CL48 DUMP OF MSGID
DS CL1 SCRATCH
CORID DS CL24 CORID
CORIDDUMP DS CL48 DUMP OF CORID
DS CL1 SCRATCH
*
EJECT
*******************************************************************************
* CODE START
*******************************************************************************
CSQCAPX DFHEIENT CODEREG=(R8,R9,R12),DATAREG=(R13)
B MAIN DO NOT EXECUTE EYE CATCHER
DC C'CICS / MQ API EXIT CSQCAPX '
DC C'DATE AND TIME ASSEMBLED : '
DC C'&SYSDATE',C','
DC C'&SYSTIME '
*******************************************************************************
MAIN DS ØH
*
* IF YOU LIKE TO EXCLUDE TRANSACTIONS FROM LOGGING, CODE SHOULD
* BE HERE...
*
* CLC EIBTRNID,=CL4'TRAN'
* BE ENDPREG
*
* SOME COMMON WORK
*
MVI BM_MSG,C' ' CLEAR MESSAGE AREA
MVC BM_MSG+1(L'BM_MSG-1),BM_MSG
UNPK TASKNUM(8),EIBTASKN UNPACK TASK NUMBER
MVZ TASKNUM+7(1),TASKNUM+6 MAKE DISPLAYABLE
*
EXEC CICS ASSIGN APPLID(BM_MAPPLID) NOHANDLE
EXEC CICS ASKTIME ABSTIME(ABSTIME)
EXEC CICS FORMATTIME ABSTIME(ABSTIME) *
    DDMYYY(BM_MDATE) DATESEP('/') *
    TIME(BM_MTIME) TIMESEP(':') NOHANDLE
*
* CHECK THAT A COMMAREA HAS BEEN PASSED
*
CLC EIBCALEN.=F'Ø' CHECK COMMAREA LENGTH

BH GOOD_LENGTH >Ø, THAT'S OKAY

* NO COMMAREA PASSED TO PROGRAM, WRITE LOG MESSAGE AND RETURN

* BGFILEMSG MSGØØ1,C,EIBTRNID,,TASKNUM
BAL R6,CSOCAPX_WRITEMSG WRITE LOG MESSAGE
B ENDPROG EXIT PROGRAM

* ESTABLISH ADDRESSABILITY

GOOD_LENGTH DS ØH
L R1Ø,DFHEICAP LOAD ADDRESS OF COMMAREA
ST R1Ø,COMPTR SAVE COMMAREA POINTER
USING MQXP_COPYPLIST,R1Ø ADDRESS THE PARAMETER LIST
L R1Ø,MQXP_PXPB ADDRESS OF XPB FROM PARMLIST
USING MQXP,R1Ø AND ADDRESS IT

* SET SOME FLAGS
MVI OBJECTNAME,C' ' CLEAR OBJECTNAME
MVC OBJECTNAME(L'OBJECTNAME-1),OBJECTNAME
MVI AFTERCALL,FALSE SET FLAG
MVI BEFORECALL,FALSE SET FLAG

* CHECK IF WE ARE BEFORE MQ CALL
LA RØ,MQXR_BEFORE LOAD
C RØ,MQXP_EXITREASON IS IT BEFORE?
BNE TSTAFTER NO .. TRY AFTER
MVI BEFORECALL,TRUE SET BEFORE TRUE
B PROCESS_MQCALL CONTINUE WITH MQ CALL

* CHECK IF WE ARE AFTER MQ CALL
TSTAFTER DS ØH
LA RØ,MQXR_AFTER LOAD
C RØ,MQXP_EXITREASON IS IT AFTER?
BNE XR_UNKWN NO .. GO TO ERROR
MVI AFTERCALL,TRUE SET AFTER TRUE
B PROCESS_MQCALL CONTINUE WITH MQ CALL

* NEITHER BEFORE NOR AFTER CALL, WRITE LOG MESSAGE AND EXIT

XR_UNKWN DS ØH
L RØ,MQXP_EXITREASON LOAD INVOCATION REASON
CVD RØ,WRKDWORD CONVERT TO PACKED DECIMAL
UNPK WORK1(8),WRKDWORD+4(4) CONVERT TO ZONED DECIMAL
MVZ WORK1+7(1),WORK1+6 MAKE IT DISPLAYABLE
BGFILEMSG MSGØØ5,C,EIBTRNID,,TASKNUM,,WORK1 PREP MESSAGE
BAL R6,CSOCAPX_WRITEMSG WRITE LOG MESSAGE
B ENDPROG EXIT PROGRAM

********************************************************************
* HERE WE PROCESS THE MQ CALL
********************************************************************

PROCESS_MQCALL DS ØH PROCESS MQ CALL
* MQOPEN PROCESSING *

ISOPEN DS ØH
LA RØ,MQXC_MQOPEN LOAD
C RØ,MQXP_EXITCOMMAND IS IT MQOPEN
BNE ISCLOSE NO .. TRY MQCLOSE
MVC_OPCODE,OP_OPEN SET CHARACTER OPCODE
CLI BEFORECALL,TRUE ARE WE BEFORE MQOPEN?
BNE ISOPEN_AFTER NO, DO AFTER-CALL PROCESSING

* BEFORE MQOPEN *

LA R2,8 OFFSET TO OBJDESCR IN PARMS
BAL R6,GETOBJECTNAME GET OBJECTNAME
BGFIOMSG MSGØØ2,I,EIBTRNID,,TASKNUM,,OPCODE,,OBJECTNAME
BAL R6,CSQCAPX_WRITEMSG WRITE LOG MESSAGE
B ENDPROG EXIT PROGRAM

* AFTER MQOPEN *

ISOPEN_AFTER DS ØH
BAL R6,GETRESULTCODES GET COMPCODE AND REASON
LA R2,16 OFFSET TO HOBJ IN PARMS
BAL R6,GETOBJECTHANDLE GET OBJECT HANDLE FROM OPEN
BGFIOMSG MSGØØ3,I,EIBTRNID,,TASKNUM,,OPCODE,,CCC,,RCC,,HOBJ
BAL R6,CSQCAPX_WRITEMSG WRITE LOG MESSAGE
BAL R7,GETCHARACTERRC RC IN CHARACTER IF NEEDED
B ENDPROG EXIT PROGRAM
EJECT

* MQCLOSE PROCESSING *

ISCLOSE DS ØH
LA RØ,MQXC_MQCLOSE LOAD
C RØ,MQXP_EXITCOMMAND IS IT MQCLOSE?
BNE ISGET NO .. TRY MQGET
MVC_OPCODE,OP_CLOSE SET CHARACTER OPCODE
LA R2,8 OFFSET TO HOBJ IN PARMS
BAL R6,GETOBJECTHANDLE GET OBJECT HANDLE
CLI BEFORECALL,TRUE BEFORE MQCLOSE?
BNE ISCLOSE_AFTER NO, AFTER-CLOSE PROCESSING

* BEFORE MQCLOSE *

BGFIOMSG MSGØØ2,I,EIBTRNID,,TASKNUM,,OPCODE,,HOBJ
BAL R6,CSQCAPX_WRITEMSG WRITE LOG MESSAGE
B ENDPROG EXIT PROGRAM

* AFTER MQCLOSE *

ISCLOSE_AFTER DS ØH
BAL R6,GETRESULTCODES GET COMPCODE AND REASON
BGFILMSG MSGØØ3,I,EIBTRNID,,TASKNUM,,OPCODE,,CCC,,RCC,,HOBJ
BAL R6,CSCCAPX_WRITEMSG WRITE LOG MESSAGE
BAL R7,GETCHARACTERRC RC IN CHARACTER IF NEEDED
B ENDPROG EXIT PROGRAM

* MQGET PROCESSING *

ISGET DS ØH
LA RØ,MQXC_MQGET LOAD
C RØ,MQXP_EXITCOMMAND IS IT MQGET?
BNE ISPUT NO .. TRY MQPUT
MVC OPCODE,OP_GET SET CHARACTER OPCODE
LA R2,8 OFFSET TO HOBJ IN PARMS
BAL R6,GETOBJECTHANDLE GET OBJECT HANDLE
CLI BEFORECALL,TRUE BEFORE MQGET?
BNE ISGET_AFTER NO, AFTER-MQGET PROCESSING

* BEFORE MQGET *

BGFILMSG MSGØØ2,I,EIBTRNID,,TASKNUM,,OPCODE,,HOBJ
BAL R6,CSCCAPX_WRITEMSG WRITE LOG MESSAGE
B ENDPROG EXIT PROGRAM

* AFTER MQGET *

ISGET_AFTER DS ØH
BAL R6,GETRESULTCODES GET COMPCODE AND REASON
BGFILMSG MSGØØ3,I,EIBTRNID,,TASKNUM,,OPCODE,,CCC,,RCC,,HOBJ
BAL R6,CSCCAPX_WRITEMSG WRITE LOG MESSAGE
BAL R7,GETCHARACTERRC RC IN CHARACTER IF NEEDED

* CHECK IF DATA WAS RECEIVED *
CLC CCC,=C’ØØØØ’ COMPLETIONCODE Ø?
BNE ENDPROG NO, EXIT PROGRAM

* COLLECT DATA *
LA R2,12 OFFSET TO MSGDESCRIPTOR
BAL R6,GETMSGIDCORID GET MSGID, CORID AFTER MQGET
LA R2,28 OFFSET TO DATALENGTH
BAL R6,GETDATALENGTH GET DATALENGTH AFTER MQGET
LA R2,24 OFFSET TO BUFFER
BAL R6,GETDATA GET DATA AFTER MQGET CALL

* WRITE LOG MESSAGES

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

Stefan Raabe
Systems Programmer
Braun AG (Germany) © Xephon 1998
An update to the screen viewing utility

We have found Richard Keane’s article, *A screen viewing utility*, published in *CICS Update*, Issue 105, August 1994, extremely useful for our Help Desk facility. A modification was described in *CICS Update*, Issue 120, November 1995, to enable this utility to be CICS 4.1 compatible and to PEEK by user-id rather than by terminal-id.

We have now added a further change to allow the automatic screen update every two seconds. We also tried a one-second delay, but this caused a problem on the terminal we tried to PEEK. In the original program the person doing the PEEKing had to press ENTER to retrieve the next PEEKed display.

Here’s what happens – the Help Desk person logs on to a CICS terminal, clears the screen, and types PEEK USER-ID followed by the ENTER key. The next screen displayed will be the user’s terminal.

In response to the PEEK command, program ITPEEK gets control and retrieves the user-id text from the originator’s terminal. It then attempts to find the terminal-id for that user-id. If the terminal-id cannot be located, an error message is sent back to the originator’s terminal and control returns to CICS, and that is the end of the story.

However, if the terminal-id is located, the program starts TRANSID LOOK, passing a COMMAREA containing the user’s terminal-id, and returns to CICS. The RECEIVE in ITPEEK initially takes the data from the originator’s terminal – note this for later! The transaction-id LOOK is started on the user’s terminal.

The program ITLOOK gets control next. Its first job is to check whether there is a COMMAREA attached to this terminal; if so, it needs to be saved for later. If no COMMAREA is attached, there is nothing to be saved. Next, the cursor position is saved and a RETRIEVE command performed to obtain the COMMAREA sent by ITPEEK. The terminal’s buffer is read and saved in a buffer, together with other information, and a START SHOW transaction is done with this buffer set as the COMMAREA.

Now there are two possible exits. If no COMMAREA was detected at
the start of ITLOOK, then there was no transaction waiting to run on this terminal, so a simple return to CICS is done. Otherwise, the transaction that was to run must be set up again by doing a return to it, specifying the COMMAREA found at the start of ITLOOK, and then returning to CICS to wait for the terminal operator to do what he was going to do before we jumped in!

Transaction SHOW now gets control back on the originator’s terminal and program ITSHOW retrieves the COMMAREA, which contains the data from the user’s terminal’s buffer. A SEND command is used to display this on the originator’s terminal and a two-second delay is performed.

The next thing to do is to check whether the originator has pressed PF3 – to end the loop – and, if so, simply exit to CICS. If PF3 has not been pressed then a RETURN IMMEDIATE command is executed using the INPUTMSG option. In the INPUTMSG option is the text PEEK USER-ID and the transaction-id specified is PEEKed. Once again the PEEK-LOOK-SHOW loop has been activated. The receive in ITPEEK now gets its data, not from the terminal, but from the INPUTMSG option of the return command. When the operation is complete, simply press PF3!

You should ensure that the two terminals have the same number of lines and columns, or, at least, that the originator’s is the larger.

ITLOOK

***********************************************************************
* PROGRAM : ITLOOK                                                  *
* TRANSID : LOOK                                                   *
***********************************************************************
*ASM XOPTS(NOPROLOG)
DFHEISTG DSECT ,
  DFHEISTG
LOOK CSECT ,
  DFHEIENT CODEREG=(3,8)
  MVC LCOMAREA(2),=H'0'
      INITIALIZE STORAGE AREAS
* CHECK TO SEE IF A COMMAREA IS PRESET FOR THIS TERMINAL. IF SO WE *
* MUST SAVE IT FOR THE NEXT TRANSACTION - WAITING TO RUN ON THIS TERM *
  LH  5,EIBCALEN   COMMAREA LENGTH
  LTR  5,5         LENGTH ZERO ?
  BZ  NOCOMARE     YES, NO NEED TO SAVE
STH 5,LCOMAREA  NO, SAVE LENGTH
L 4,DFHEICAP   ADDRESS OF COMMAREA
BCTR 5,Ø    LESS ONE FOR EXECUTE
CH 5,LENCOMSA  CHECK LENGTH NOT TOO LARGE
BH NOSHOW      YES, TOO LONG, EXIT
EX 5,MOVINCOM MOVE INTO SAVEAREA

* RETRIEVE THE TERMINAL ID ON WHICH 'PEEK' WAS STARTED *

NOCOMARE EXEC  CICS RETRIEVE ,
    INTO(PARMS) ,    GET INVOKER'S TERM-ID
    LENGTH(LPARMS)
MVC PEEKTERM(4),TERMID  INVOKER'S TERM-ID
MVC LOOKTERM(4),EIBTRMID TARGET TERM-ID
MVC CURSOR(2),EIBCPOSN  CURSOR POS
MVC LRBUFFER(2),SLRBUFER SET LENGTH OF RECEIVE BUFFER
EXEC CICS RECEIVE ,    READ CONTENTS OF SCREEN
    INTO(RBUFFER) X
    LENGTH(LRBUFFER) X
    BUFFER X
    ASIS X
    LEAVEKB X
LH 4,LRBUFFER LENGTH OF DATA READ
AH 4,=H'10'  PLUS 10 BYTES FOR TRANS 'SHOW'
STH 4,LPASDATA SAVE LENGTH

* IF THE TRANSACTION WHICH PREVIOUSLY RAN ON THIS TERMINAL WAS PSEUDO-
* CONVERSATIONAL IT WILL HAVE RETURNED TO CICS USING EXEC CICS
  * RETURN TRANS-ID('XXXX'). *
* WHEN THE USER ENTERED DATA TRANS-ID XXXX WOULD HAVE BEEN STARTED BY *
* CICS. BECAUSE WE ARE RUNNING IN BETWEEN, WE MUST ALSO RETURN TO
* CICS, SETTING XXXX AS THE NEXT TRANSACTION TO BE STARTED.
*
* START TRANSACTION 'SHOW' AT INVOKER'S TERMINAL TO DISPLAY SCREEN *
*
STRTSHOW EXEC  CICS START
    TRANSID('SHOW') X
    TERMID(PEEKTERM) X
    FROM(CURSOR) X
    LENGTH(LPASDATA) X
CLC LCOMAREA(2),=H'Ø'  WAS A COMMAREA PRESENT
BE NOSHOW      NO, RETURN NO COMMAREA
EXEC CICS INQUIRE TERMINAL(EIBTRMID) NEXTTRANSID(LASTTRAN)
CLC LASTTRAN(4),=CL4' ' WAS A TRANSACTION TO BE STARTED?
BNE EXIT YES, RETURN WITH TRANS-ID
NOSHOW EXEC  CICS RETURN ,
  NO, RETURN
* RETURN TO CICS WITH TRANSID *
EXIT EXEC  CICS RETURN
    COMMAREA(LCOMAREA) X
    TRANSID(LASTTRAN) X
    LENGTH(LCOMAREA) X

* CONSTANTS *
ITPEEK

***********************************************************************
* PROGRAM : ITPEEK                                                   *
* TRANSID : PEEK                                                      *
***********************************************************************

DFHEISTG DSECT ,
  DFHEISTG
TRANSID DS CL4       MAP SCREEN
FILLER DS CL1
USERID DS CL8
AAAAAAA DS CL40
PEEK
  EXEC CICS RECEIVE INTO(TRANSID) LENGTH(SLEN) NOHANDLE
  MVC TARGET,USERID       MOVE IN TARGET USER-ID
  OC TARGET,=CL8' '      CONVERT TO UPPER CASEx
  MVC TERMID(4),EIBTRMID   SAVE INVOKER'S TERMINAL-ID
  EXEC CICS HANDLE CONDITION , IF TARGET TERM-ID NOT FOUND X
   TERMIDERR(WRONGTRM)   X
   END(WRONGTRM)
  EXEC CICS INQUIRE TERMINAL START
  LOOP
   EXEC CICS INQUIRE TERMINAL (USTERM) NEXT *
    USERID(USERID)
    CLC USERID,TARGET
  LOOP
  STARTLOO EXEC CICS START , START TRANSACTION LOOK X
    TRANSID('LOOK') , PASSING OUR TERMINAL-ID FOR X
    FROM(PARMS) , TRANSACTION 'SHOW' X
LENGTH(LPARMS)
TERMID(USTERM)
EXIT EXEC CICS RETURN , RETURN TO CICS
*
WRONGTRM MVC ERRTERM,TARGET MOVE TARGET NAME TO ERROR MSG
EXEC CICS SEND , DISPLAY ERROR MESSAGE X
FROM(EMSG) X
LENGTH(LEMSG) X
CTLCHAR(WCC) X
ERASE
EXEC CICS SEND CONTROL X
FREEKB
B EXIT EXIT
*
CONSTANTS
*
WCC DC X'F1' WRITE CONTROL CHARACTER
SMMSG DC X'1DC1' ATTRIBUTE, UNPROTECTED, MDT ON
DC C'TRANSACTION LOOK STARTED FOR USER '
TARGET DC CL8' '
LSMSG DC AL2(*-SMMSG)
EMSG DC X'1DF2' ERROR MESSAGE 1
DC C'USER '
ERRTERM DC CL8' '
DC C' IS NOT CURRENTLY LOGGED ON ' 
LEMSG DC AL2(*-EMSG)
PARMS DS ØCL5 COMMAREA PASSED TO 'LOOK'
TERMID DC CL5' '
LPARMS DC AL2(*-PARMS)
HEXNULL DS XL14'ØØ'
USERMAND DS XL1'ØØ'
USERNAME DS XL12'ØØ'
LREC DC H'3ØØØ'
SLEN DC H'57'
RECORD DS 3ØØØC
USTERM DS 4C
END PEEK

ITSHOW

***********************************************************************
* PROGRAM : ITSHOW
* TRANSID : SHOW
***********************************************************************
DFHEISTG DSECT , DFHEISTG
SHOW CSECT ,
MVC LRBUFFER(2),SLRBUF SET LENGTH OF SEND BUFFER
MVC LRECbuf(2),SLRECbuf SET LENGTH OF RETRIEVE BUFFER
EXEC CICS RETRIEVE , RETRIEVE DATA PASSED BY 'LOOK'
INTO(CURSOR) X
LENGTH(LRECVBUF)
LH 4,LRECVBUF  LENGTH
SH 4,-H'10'    LESS 10 BYTES WHICH ARE FOR US
STH 4,LRBUFFER SET SEND LENGTH
MVC LASTTRAN(4),NEXTTRAN  TRAN NAME SCHEDULED ON TARGET
EXEC CICS INQUIRE TERMINAL(PEEKTERM) USERID(PEEKED)
EXEC CICS SEND , DISPLAY DATA ON SCREEN OF TARGETX
FROM(RBUFFER) X
LENGTH(LRBUFFER) X
ERASE X
CTLCHAR(WCC)
EXEC CICS SEND CONTROL , SET CURSOR POSITION X
FREEKB X
CURSOR(CURSOR)
EXEC CICS DELAY FOR SECONDS (2)
CLI EIBAID,X'F3'
BE EXIT
EXEC CICS RETURN IMMEDIATE X
TRANSID ('PEEK') X
INPUTMSG (REPEAT) X
INPUTMSGLEN (14)
EXEC CICS RETURN , RETURN TO CICS
EXIT EQU *
EXEC CICS SEND , SEND 'PEEK USER-ID' WHICH IS X
FROM(ENDMESS) , UNPROTECTED, ENABLING THE USER X
LENGTH(LENDMSL) , TO JUST PRESS ENTER TO REINVOKE X
ERASE , 'PEEK'
EXEC CICS RETURN , RETURN TO CICS

* CONSTANTS *
DS ØF
WCC DC X'F3'  WRITE CONTROL CHARACTER
REPEAT DC CL5'PEEK '  END MESSAGE
PEEKED DC CL8' '
INFO DC C' NEXT TRANSACTION ID : '
LASTTRAN DC CL5' '
LREPEAT DC AL2(*-REPEAT)
CURSOR DC H'Ø'  RETRIEVE BUF,PREFIXED BY CURPOS,
NEXTTRAN DC CL4' '  NEXT TRANS-ID OF TARGET TERMINAL
PEEKTERM DC CL4' '  NAME OF THEN TERMINAL
RBUFFER DC 3000C' ' DATA OBTAINED BY 'LOOK'
LRECVBUF DC AL2(*-RBUFFER) LENGTHS
SLRBUFFER DC AL2(*-RBUFFER-2)
LRECVBUF DC AL2(LRBUFFER-PEEKTERM)
SLRECVBU DC AL2(LRBUFFER-PEEKTERM)
ENDMESS DC XL12'1140403C5D7C5F114EC31DE8'
    DC CL33'OK, I'm GOING then - See Yer !!!'
    DC XL2'1D40'
LENDMSL DC H'47'
END SHOW
Managing CICS printers – part 2

This month we conclude the article on managing CICS SNA printers without the need to invoke the master terminal transaction CEMT.

IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
IF RESPONSE = DFHRESP(NOTFND) THEN
    MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
    GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*
*    THE SCREEN IS BUILT WITH SIX RECORDS.
*
    MOVE PRID   TO PRT1O.
    MOVE NETNAM TO NETNAM1O.
    MOVE JESPRT TO JESPRT1O.
    MOVE USER   TO USER1O.
    MOVE FIRM   TO FIRM1O.
*
*    RIDB NEEDS EXISTING PRID
*
    MOVE RIDF IN COMMAREA TO RIDB IN COMMAREA.
*
*    READ 2ND RECORD.
*
    PERFORM READ-NEXT.
*
*    CHECK RESPONSES
*
    IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
    IF RESPONSE = DFHRESP(NOTFND) THEN
        MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
        GO TO MENU.
    IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*
*    MOVE FIELDS ---> MAP
*
    MOVE PRID   TO PRT2O.
    MOVE NETNAM TO NETNAM2O.
    MOVE JESPRT TO JESPRT2O.
    MOVE USER   TO USER2O.
    MOVE FIRM   TO FIRM2O.
*
*    READ 3RD RECORD.
*
    PERFORM READ-NEXT.
*
* CHECK RESPONSES
* IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
IF RESPONSE = DFHRESP(NOTFND) THEN
    MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
    GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*
* MOVE FIELDS ---> MAP
*
   MOVE PRID   TO PRT3O.
   MOVE NETNAM TO NETNAM3O.
   MOVE JESPRT TO JESPRT3O.
   MOVE USER   TO USER3O.
   MOVE FIRM   TO FIRM3O.
*
* READ 4TH RECORD.
*
PERFORM READ-NEXT.
*
* CHECK RESPONSES
*
   IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
   IF RESPONSE = DFHRESP(NOTFND) THEN
       MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
       GO TO MENU.
   IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*
* MOVE FIELDS ---> MAP
*
   MOVE PRID   TO PRT4O.
   MOVE NETNAM TO NETNAM4O.
   MOVE JESPRT TO JESPRT4O.
   MOVE USER   TO USER4O.
   MOVE FIRM   TO FIRM4O.
*
* READ 5TH RECORD.
*
PERFORM READ-NEXT.
*
* CHECK RESPONSES
*
   IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
   IF RESPONSE = DFHRESP(NOTFND) THEN
       MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
       GO TO MENU.
   IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*
* MOVE FIELDS ---> MAP
MOVE PRID TO PRT50.
MOVE NETNAM TO NETNAM50.
MOVE JESPRT TO JESPRT50.
MOVE USER TO USER50.
MOVE FIRM TO FIRM50.

* *
READ 6TH RECORD.
* *
PERFORM READ-NEXT.
* *
CHECK RESPONSES
* *
IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOHIGH.
IF RESPONSE = DFHRESP(NOTFND) THEN
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
   GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
* *
MOVE FIELDS ---> MAP
* *
MOVE PRID TO PRT60.
MOVE NETNAM TO NETNAM60.
MOVE JESPRT TO JESPRT60.
MOVE USER TO USER60.
MOVE FIRM TO FIRM60.
* *
The screen is erased and the page is displayed at the terminal.
* *
EXEC CICS SEND MAP('BROWSE') MAPSET('PRTMAPC')
   ERASE END-EXEC.
* *
CONTROL IS RETURNED TO CICS, ALONG WITH A COMMAREA AND A TRANSACTION IDENTIFIER NAMING THE NEXT TRANSACTION. THE COMMAREA CONTAINS THE PROGRAM'S FILE POINTERS (RECORD KEYS) TO ENABLE SUBSEQUENT INVOCATIONS OF THE PROGRAM TO CONTINUE BROWSING BY USING THESE Pointers AS A Reference.
* *
EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
   LENGTH(9) END-EXEC.
* *
The backward browse is similar to the forward browse.
* *
THE BACKWARD BROWSE IS SIMILAR TO THE FORWARD BROWSE.
* *
LOW END OF FILE
* *
RESET MAP PRTMAPC
*
MOVE LOW-VALUES TO BROWSEO.
* RIDF ---> NEXT FPAGE
* IF EIBCALEN = Ø THEN GO TO TEST-STATS.
* START BROWSE WHERE WE LEFT OFF LAST TIME.
* EXEC CICS STARTBR FILE('PRINT') RIDFLD(RIDB IN COMMAREA) RESP(RESPONSE) END-EXEC.
* CHECK RESPONSES
* IF RESPONSE = DFHRESP(NOTFND) THEN
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
   GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
TEST-STATS.
* IF STATS IN COMMAREA = 'H' THEN GO TO PREV-LINE.
* READ AND DISCARD THE RECORD POINTED TO BY RIDB ONLY
* IF THE HIGH END OF THE FILE HAS NOT BEEN REACHED
* PERFORM READ-PREV.
* CHECK RESPONSES
* IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW.
IF RESPONSE = DFHRESP(NOTFND) THEN
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
   GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN
   GO TO ERRORS.
PREV-LINE.
* READ SIX RECORDS IN DESCENDING ORDER.
* MOVE ' ' TO STATS IN COMMAREA.
PERFORM READ-PREV.
* CHECK RESPONSES
* IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW.
IF RESPONSE = DFHRESP(NOTFND) THEN
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
   GO TO MENU.
IF RESPONSE NOT = DFHRESP(NORMAL) THEN
   GO TO ERRORS.
* MOVE FIELDS ---> MAP

* MOVE PRID TO PRT60.
  MOVE NETNAM TO NETNAM60.
  MOVE JESPRT TO JESPRT60.
  MOVE USER TO USER60.
  MOVE FIRM TO FIRM60.
  MOVE RIDB IN COMAREA TO RIDF IN COMAREA

* READ 2ND RECORD.

* PERFORM READ-PREV.

* CHECK RESPONSES

  IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW.
  IF RESPONSE = DFHRESP(NOTFND) THEN
      MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
      GO TO MENU.
  IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.

* MOVE FIELDS ---> MAP

* MOVE PRID TO PRT50.
  MOVE NETNAM TO NETNAM50.
  MOVE JESPRT TO JESPRT50.
  MOVE USER TO USER50.
  MOVE FIRM TO FIRM50.

* READ 3RD RECORD.

* PERFORM READ-PREV.

* CHECK RESPONSES

  IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW.
  IF RESPONSE = DFHRESP(NOTFND) THEN
      MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES
      GO TO MENU.

* MOVE FIELDS ---> MAP

* MOVE PRID TO PRT40.
  MOVE NETNAM TO NETNAM40.
  MOVE JESPRT TO JESPRT40.
  MOVE USER TO USER40.
  MOVE FIRM TO FIRM40.

* READ 4TH RECORD.

PERFORM READ-PREV.

*  
* CHECK RESPONSES
*  
IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW. 
IF RESPONSE = DFHRESP(NOTFND) THEN 
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES 
   GO TO MENU. 
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.

*  
* MOVE FIELDS ---> MAP
*  
MOVE PRID TO PRT3O. 
MOVE NETNAM TO NETNAM3O. 
MOVE JESPRT TO JESPRT3O. 
MOVE USER TO USER3O. 
MOVE FIRM TO FIRM3O.

*  
* READ 5TH RECORD.
*  
PERFORM READ-PREV.

*  
* CHECK RESPONSES
*  
IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW. 
IF RESPONSE = DFHRESP(NOTFND) THEN 
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES 
   GO TO MENU. 
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.

*  
* MOVE FIELDS ---> MAP
*  
MOVE PRID TO PRT2O. 
MOVE NETNAM TO NETNAM2O. 
MOVE JESPRT TO JESPRT2O. 
MOVE USER TO USER2O. 
MOVE FIRM TO FIRM2O.

*  
* READ 6TH RECORD.
*  
PERFORM READ-PREV.

*  
* CHECK RESPONSES
*  
IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO TOOLOW. 
IF RESPONSE = DFHRESP(NOTFND) THEN 
   MOVE 'END OF FILE - PLEASE RESTART' TO MESSAGES 
   GO TO MENU. 
IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*    MOVE FIELDS ---> MAP
*    MOVE PRID   TO PRT10.
MOVE NETNAM TO NETNAM10.
MOVE JESPRT TO JESPRT10.
MOVE USER   TO USER10.
MOVE FIRM   TO FIRM10.
*    DISPLAY MAP
*    EXEC CICS SEND MAP('BROWSE') MAPSET('PRTMAPC')
           ERASE END-EXEC.
*    RETURN WITH A COMM. AREA.
*    EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA)
           LENGTH(13) END-EXEC.
*    AFTER THE RECEIVE COMMAND EXECUTES, THE PROGRAM TESTS THE
*    OPERATORS RESPONSE. ONLY CLEAR, PF8, PF7, F, OR B
*    KEYS ARE TAKEN AS VALID INPUT.
*    ALL OTHER RESPONSES ARE IGNORED.
*    PROMPT.
    EXEC CICS RECEIVE MAP('BROWSE') MAPSET('PRTMAPC')
           RESP(RESPONSE) END-EXEC.
*    CHECK CLEAR KEY.
*    IF EIBAID = DFHCLEAR THEN
           MOVE 'PRESS CLEAR TO EXIT' TO MESSAGES
           GO TO MENU.
*    CHECK PF KEYS.
*    IF EIBAID = DFHPF8 OR DIRI = 'F' THEN GO TO PAGE-FORWARD.
    IF EIBAID = DFHPF7 OR DIRI = 'B' THEN GO TO PAGE-BACKWARD.
*    CHECK RESPONSES.
*    IF RESPONSE = DFHRESP(MAPFAIL) THEN
           MOVE 'PRESS CLEAR TO EXIT' TO MESSAGES
           GO TO MENU.
    IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
*    INVALID-RESEND
*    EXEC CICS SEND MAP('BROWSE') MAPSET('PRTMAPC') END-EXEC.
*    RETURN WITH A COMM. AREA.
EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA) LENGTH(13) END-EXEC.

* IF THE END OF FILE IS REACHED ON A READNEXT, ANY RECORDS READ TO THAT POINT ARE DISPLAYED, TOGETHER WITH A HIGHLIGHTED MESSAGE 'HI-END OF FILE'.

TOOHIGH.
   MOVE 'H' TO STATS IN COMMAREA.
   MOVE RIDF IN COMMAREA TO RIDB IN COMMAREA.
   MOVE ' ' TO DIRO.
   MOVE 'HI-END OF FILE' TO MSG10.

* BRT+PROT ATTR

   MOVE DFHBMASB TO MSG1A.
   EXEC CICS SEND MAP('BROWSE') MAPSET('PRTMAPC') ERASE END-EXEC.

* RETURN WITH A COMM. AREA.

* EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA) LENGTH(13) END-EXEC.

* IF THE START OF FILE IS REACHED ON A READPREV (BACKWARD BROWSE) THEN THE ENDFILE CONDITION OCCURS AND TOOLOW GETS CONTROL. ANY RECORDS READ UP TO THAT POINT ARE DISPLAYED TOGETHER WITH A HIGHLIGHTED MESSAGE 'LO-END OF FILE'.

TOOLOW.
   MOVE '$$$$' TO RIDF IN COMMAREA.
   MOVE '$$$$' TO RIDB IN COMMAREA.
   MOVE ' ' TO DIRO.
   MOVE 'LO-END OF FILE' TO MSG20.

* BRT+PROT ATTR

   MOVE DFHBMASB TO MSG2A.
   EXEC CICS SEND MAP('BROWSE') MAPSET('PRTMAPC') ERASE END-EXEC.
   EXEC CICS RETURN TRANSID(EIBTRNID) COMMAREA(COMMAREA) LENGTH(13) END-EXEC.

* THIS ROUTINE EXECUTES A READNEXT COMMAND TO READ THE NEXT RECORD FROM THE FILE INTO THE FILE AREA, WITH RESPECT TO THE KEY IN RIDF.

READ-NEXT.
   EXEC CICS READNEXT INTO(PRINT) FILE('PRINT')
RIDFLD(RIDF IN COMMAREA) LENGTH(RECLENGTH) RESP(RESPONSE) END-EXEC.

*  THIS ROUTINE EXECUTES A READPREV COMMAND TO READ THE NEXT
*  RECORD INTO THE FILE AREA, WITH RESPECT TO THE KEY IN RIDB.
*
READ-PREV.
  EXEC CICS READPREV INTO(PRINT) FILE('PRINT')
  RIDFLD(RIDB IN COMMAREA)
  LENGTH(RECLENGTH) RESP(RESPONSE) END-EXEC.

*  IN SOME ERROR SITUATIONS A DUMP IS TAKEN AND THE MESSAGE
*  'TRANSACTION TERMINATED' IS MOVED TO MESSAGES FOR DISPLAY
*  ON THE OPERATOR INSTRUCTION SCREEN.
*
ERRORS.
  EXEC CICS DUMP DUMPCODE('ERRS') END-EXEC.
  MOVE 'TRANSACTION TERMINATED' TO MESSAGES.

*  DISPLAY GENERAL MENU THEN EXIT.
*
MENU.
*
  RESET MAP 'A'
*
  MOVE LOW-VALUE TO MENUO.
  MOVE DFHBMASB TO MSGA.
  MOVE MESSAGES TO MSGO.
*
  THIS CODE DISPLAYS THE OPERATOR INSTRUCTION MENU WITH A
  MESSAGE WHICH HAS BEEN STORED IN MESSAGES.
*
  EXEC CICS SEND MAP('MENU') MAPSET('PRTMAPA') ERASE END-EXEC.
*
  THE PROGRAM TERMINATES BY RETURNING TO CICS.
*
  EXEC CICS RETURN END-EXEC.
GOBACK.

PRINTMNU
*****************************************************************
*                                                               *
* MODULE NAME = PRINTMNU                                       *
*                                                               *
*****************************************************************
IDENTIFICATION DIVISION.
PROGRAM-ID. PRINTMNU.
ENVIRONMENT DIVISION.
DATA DIVISION.
* 
PROCEDURE DIVISION.
EXEC CICS SEND MAP('MENU') MAPSET('PRTMAPA')
   MAPONLY ERASE END-EXEC.
EXEC CICS RETURN END-EXEC.
GOBACK.

PRTMAPA
TITLE 'PRTMAPA - MAP FOR OPERATOR INSTRUCTIONS - COBOL'

PRTMAPA
DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET), LANG=COBOL,TIOAPFX=NO,EXTATT=MAPONLY,COLOR=BLUE

MENU
DFHMDF POS=(1,8),LENGTH=24,INITIAL='Printer Information Tool',*
   HIGHLIGHT=UNDERLINE
DFHMDF POS=(1,33),LENGTH=1,ATTRB=PROT
DFHMDF POS=(3,1),LENGTH=28,INITIAL='OPERATOR INSTR. - ENTER PE*
   NU'
DFHMDF POS=(4,1),LENGTH=43,INITIAL='PRINTER INQUIRY - ENTER PI*
   NO AND PRINTER-ID'
DFHMDF POS=(5,1),LENGTH=43,INITIAL='PRINTER BROWSE - ENTER PB*
   RW AND PRINTER-ID'
DFHMDF POS=(6,1),LENGTH=43,INITIAL='PRINTER ADD - ENTER PA*
   DD AND PRINTER-ID'
DFHMDF POS=(7,1),LENGTH=43,INITIAL='PRINTER UPDATE - ENTER PU*
   LD AND PRINTER-ID'
DFHMDF POS=(8,1),LENGTH=43,INITIAL='PRINTER DELETE - ENTER PD*
   EL AND PRINTER-ID'

MSG
DFHMDF POS=(11,1),LENGTH=39,INITIAL='PRESS CLEAR TO EXIT'
DFHMDF POS=(12,1),LENGTH=18,INITIAL='ENTER TRANSACTION:'
DFHMDF POS=(12,20),LENGTH=4,ATTRB=IC,COLOR=GREEN, *
   HIGHLIGHT=REVERSE
DFHMDF POS=(12,25),LENGTH=10,INITIAL='PRINTER-ID'

KEY
DFHMDF POS=(12,36),LENGTH=4,COLOR=GREEN,ATTRB=UNPROT, *
   HIGHLIGHT=REVERSE
DFHMDF POS=(12,41),LENGTH=1

DFHMSD TYPE=FINAL

END

PRTMAPB
TITLE 'PRTMAPB - MAP FOR PRINTER INQUIRY/UPDATE - COBOL'

PRTMAPB
DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET), LANG=COBOL,TIOAPFX=NO,EXTATT=MAPONLY

DETAIL
DFHMDI SIZE=(24,79)

TITLE
DFHMDF POS=(1,15),LENGTH=15
DFHMDF POS=(4,1),LENGTH=13,INITIAL='Printer-ID :',COLOR=BLUE

PRID
DFHMDF POS=(4,15),LENGTH=4,ATTRB=PROT
DFHMDF POS=(4,20),LENGTH=1
PRTMAPC

TITLE 'PRINTER - MAP FOR FILE BROWSE - COBOL'

PRTMAPC DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET), *
LANG=COBOL,TIOAPFX=YES,EXTATT=MAPONLY

BROWSE DFHMDI SIZE=(24,80)

DIR DFHMDF POS=(1,1),LENGTH=1,ATTRB=IC
DFHMDF POS=(1,3),LENGTH=1
DFHMDF POS=(1,30),LENGTH=14,INITIAL='PRINTER BROWSE', *
COLOR=BLUE,HILIGHT=UNDERLINE
DFHMDF POS=(1,45),LENGTH=01,ATTRB=PROT
DFHMDF POS=(3,1),LENGTH=3,INITIAL='PRT',COLOR=BLUE, *
HILIGHT=UNDERLINE
DFHMDF POS=(3,7),LENGTH=7,INITIAL='NETNAME',COLOR=BLUE, *
HILIGHT=UNDERLINE
DFHMDF POS=(3,17),LENGTH=3,INITIAL='JES',COLOR=BLUE, *
HILIGHT=UNDERLINE
DFHMDF POS=(3,23),LENGTH=5,INITIAL='USERS',COLOR=BLUE, *
HILIGHT=UNDERLINE
DFHMDF POS=(3,65),LENGTH=12,INITIAL='MANUFACTURER',COLOR=BLUE, *
HILIGHT=UNDERLINE

PRT1 DFHMDF POS=(4,1),LENGTH=4
NETNAM1 DFHMDF POS=(4,7),LENGTH=8
JESPRT1 DFHMDF POS=(4,17),LENGTH=5
USER1 DFHMDF POS=(4,23),LENGTH=40
FIRM1 DFHMDF POS=(4,65),LENGTH=15

PRT2 DFHMDF POS=(5,1),LENGTH=4
NETNAM2 DFHMDF POS=(5,7),LENGTH=8
JESPRT2 DFHMDF POS=(5,17),LENGTH=5
USER2 DFHMDF POS=(5,23),LENGTH=40
FIRM2 DFHMDF POS=(5,65),LENGTH=15

PRT3 DFHMDF POS=(6,1),LENGTH=4
NETNAM3 DFHMDF POS=(6,7),LENGTH=8
JESPRT3 DFHMDF POS=(6,17),LENGTH=5
USER3 DFHMDF POS=(6,23),LENGTH=40
FIRM3 DFHMDF POS=(6,65),LENGTH=15

PRT4 DFHMDF POS=(7,1),LENGTH=4
NETNAM4 DFHMDF POS=(7,7),LENGTH=8
JESPRT4 DFHMDF POS=(7,17),LENGTH=5
USER4 DFHMDF POS=(7,23),LENGTH=40
FIRM4 DFHMDF POS=(7,65),LENGTH=15
PRINTFIL

*****************************************************************
* MODULE NAME = PRINTFIL                                      *
* DESCRIPTIVE NAME = File layout for printer application     *
*****************************************************************

FILEREC.
  STAT       PIC X.
  PRID       PIC X(4).
  NETNAM     PIC X(8).
  TDQNAM     PIC X(4).
  FIRM       PIC X(15).
  MODEL      PIC X(15).
  PRMODE     PIC X(15).
  JESPRT     PIC X(5).
  OUTC       PIC X.
  APPLIC     PIC X(15).
  USER       PIC X(40).
  ABT        PIC X(3).
  BUILD      PIC X(2).
  FLOOR      PIC X(8).
  ROOM       PIC X(4).
  SERV       PIC X(6).
  CRE        PIC X(6)

Giselher Wieland (Germany)  © Xephon 1998
IBM has announced a four-step roadmap for CICS Transaction Server based around the deployment of new Web-based applications, and also announced details for Version 1.3.

The first step involves extending existing CICS code to the Web, via HTML and Java interfaces. Next comes a focus on consolidating desktop and mainframe programmers, with Version 1.3 allowing programmers to wrap CICS programming such as COBOL with Java, allowing desktop programmers to work with Java code to develop applications that run on a CICS/390 server. Mainframe programmers will be able to wrap mainframe programs with Java code to make them easily used by desktop programmers.

TS Version 1.3 promises the ability to write pure Java CICS applications and CICS Java Beans. The desktop Java code will remain the same, but the CICS programming will be pure Java instead of wrapped code. Finally, there’s the implementation of Enterprise Java Beans, running on CICS/390, adding transactional capabilities to Java Beans.

For further information contact your local IBM representative.

***

Tivoli has announced Version 2.1 of Tivoli Global Enterprise Manager and Release 2.0 of Tivoli Manager for MQSeries, providing end-to-end host and distributed management of applications based on CICS and MQSeries.

Tivoli GEM Version 2.1 includes better scalability for managing large, complex application and middleware environments.

It can also manage CICSplex environments through built-in integration with IBM’s CICSpalx System Manager.

Release 2.0 of the Tivoli Manager for MQSeries covers OS/390 environments and includes management of CICS, enabling applications that use both MQSeries and CICS, on mainframe and distributed environments, to be managed by a single product.

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

***

IBM has announced Version 2 of its COBOL and CICS Command Level Conversion Aid (CCCA) for VSE/ESA, providing the means to convert old COBOL source code and copy modules to new versions of COBOL. Also new is the capability to convert COBOL applications to use the new Millennium Language Extensions (MLEs).

Other facilities and functions include converting EXEC CICS commands, removing and converting the base locator for linkage section mechanism and references, eliminating conflicts between user-defined names and words reserved for new versions of COBOL, and converting both source programs and copy modules.

For further information contact your local IBM representative.

***