May 2002

In this issue

3 Fixing a CICS hung terminal problem using the XMEOUT message exit

11 Moving large amounts of data between CICS and Java (or ASP) using ECI

27 Ensuring absolutely trouble-free CICS operation – revisited

27 Automatic PHASEIN with a simple interface between batch jobs and CICS

43 CICS questions and answers

44 CICS news

© Xephon plc 2002
Subscriptions and back-issues
A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the December 1998 issue, are available separately to subscribers for £16.00 ($24.00) each including postage.

*CICS Update* on-line
Code from *CICS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/cics; you will need to supply a word from the printed issue.
Fixing a CICS hung terminal problem using the XMEOUT message exit

We have a lot of TCP/IP terminals that attach to our CICS systems using the functions of E-NETWORK Commserver. These devices are handed a VTAM LU in a ‘next available’ fashion from a large predefined pool of LUs. Occasionally, one of these devices disconnects from CICS without CICS knowing about it. This leaves the terminal in an ACQuired state as far as CICS is concerned, but in an available state as far as TCP/IP and E-NETWORK Commserver are concerned. When the next device tries to attach to CICS, E-NETWORK Commserver hands the next available VTAM LU to it, which in this case is a device that is still ACQuired to CICS. CICS responds with the message:

DFHZC2411 E date time cicsapp1 DUMY CSNE vtamlu attempted invalid logon. ((7 Module name:DFHZATA)

The user is then kicked off and is forced to try again. If this unusable but available LU is not handed off to a terminal that attaches to a different CICS (one that does not have the LU acquired) then the process will start all over again. At times when all the applications being used are on the CICS that has the LU ACQuired, every new user will be presented with this unavailable terminal and get kicked off – resulting in no one being able to logon. This always happens at 3am or some other ungodly hour. The fix is to look for the DFHZC2411 message in the CICS log and release the hung terminal.

IBM has tried to figure this out, but has been unable to without my running a very detailed trace all the time. This causes a lot of overhead that we cannot generally afford. Of course, when I do run the trace, the event does not happen, and the moment I turn it off, it happens. After many frustrating months of early morning telephone calls, I decided to circumvent the problem by using the XMEOUT exit to reroute the DFHZC2411 message to the system log. I then use MPF to trap it, interpret it, and issue the necessary CEMT S TERM(xxxx) REL command, so the user should get kicked off only once. Hopefully that will result in fewer telephone calls.

Below you will find the coding for the exit itself and all the other
In SYS1.PARMLIB(MPFLST00) you must add an entry like this:

```
DFHZC2411,AUTO(YES)
```

This tells MPF to route the console message to NETVIEW AUTOOPERATOR.

In SYS1.OPER1.PARMS(MSG01) you must add some code to cause NETVIEW to execute the proper CLIST:

```
IF MSGID= 'DFHZC2411' & TOKEN(8) = A.
   THEN EXEC( CMD('DFH2411R ' A)
      ROUTE(ONE AUTOMVS3))
      DISPLAY(Y) NETLOG(Y);
```

This will cause autooperator to execute a NETVIEW CLIST, DFH2411R, and pass it the contents of TOKEN(8) from the message. TOKEN(8) in this case is the VTAM LU name involved.

Here is DFH2411R, the NETVIEW CLIST mentioned above:

```
DFH2411R CLIST
&CONTROL ERR
**************************************************************************
* WHEN EXECUTED - WHEN DFH2411 MESSAGE INDICATES HUNG CICS TERM* *
* ACTIONS - RELEASE HUNG TERMINAL
**************************************************************************
PARSEL2R PARMSTR THISTERM
 *
WTO DFH2411R - HUNG TERMINAL DETECTED -&THISTERM-
 *
&TERM = &SUBSTR &THISTERM 5 4
 *
WTO TERMINAL &TERM HUNG UP AND WILL BE RELEASED
 *
MVS S COMMAND,PARM='F CICSP1,CEMT S TERM(&TERM) REL'
&EXIT
```

This uses a program called COMMAND, which is shareware that allows jobs to issue system commands. I have included the source for it here as a convenience if you do not already have it:

```
**************************************************************************
* THIS ROUTINE WILL ALLOW A BATCH JOB TO ISSUE OS OPERATOR COMMANDS. THIS FUNCTION IS USEFUL TO OPERATIONS IN EXECUTING CERTAIN FUNCTIONS SUCH AS SETTING CONSOLE ROUTE CODES JES2/HASP INIT CLASSES ETC.
**************************************************************************
```
Here is the XMEOUT exit itself. This was modified from the CICS supplied sample DFH$SXP4:

************
********
******** MUST USE BATCH COMPIL e FOR THIS EXIT PROGRAM

********

* MODULE NAME = DPKCS107
* DESCRIPTIVE NAME = CICS (RDO) Sample User Exit Program 6

* @BANNER_START@
  5655-147
  CICS 5.3.0
  (Element of CICS Transaction Server
   Version 1 Release 3)
* @BANNER_END@

* STATUS = 5.1.0

* FUNCTION =
  Provides a sample user exit to show how to change the
  routing of a message from a transient data queue to
  a list of consoles.

  This sample shows how to route a message destined for
  transient data queue CSCS, to consoles with route codes
  2 & 11.

* NOTES :  
  DEPENDENCIES = S/370   
  None.

* RESTRICTIONS =
  None.

* PATCH LABEL = Via DFHPATCH Macro
* MODULE TYPE = Executable | Table
* PROCESSOR = Assembler
* ATTRIBUTES = Read only, Serially Reusable, <Authorized>

*------------------------------------------------------------------

* CHANGE ACTIVITY :
  $MOD(DFH$SXP4),COMP(SAMPLES),PROD(CICS):

  PN= REASON REL YYMDD HDXIII : REMARKS
  $P0= 507 320 890814 HD5EISR: Implicit flag.
  $P1= M06695 320 900129 HD6I8S: Change Message Number in code From:
  : 0101 to 0108.
  $P2= M62307 320 900602 HD3BADW: Use UERCNORM return code EQU
  $P3= M96433 510 960205 HD4PALS : Add RMODE ANY & Change SN0108
  : to SN1100

********************************************************************
/*\* Start of ABSTRACT commenting */

This instruction sets up the Sample user exit point.

DFHEXIT TYPE=EP, ID=XMEOUT

The following DSECT maps a storage area you can use to make the exit program re-entrant by storing the address of the storage you acquire in the first four bytes of the 260-byte area provided by the user exit handler (DFHUEH) and addressed by UEPXSTOR.

TRANSTOR DSECT

* Register Equates *

R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
PMNTD EQU R3 Number of TD queues
PMNRC EQU R4 Number of Route codes
PMTDQ EQU R5 Array of TD queues
PMNUM EQU R6 Message number
PMDOM EQU R7 Domain id
PMROU EQU R8 Route code array
EXIT_RC EQU R15

* The next seven instructions form the normal start of a sample user exit program, setting the addressing mode to 31-bit, saving the calling program's registers, establishing base addressing, and establishing the
* addressing of the user exit parameter list

*******************************************************************************
DPKCS107 CSECT
DPKCS107 AMODE 31
DPKCS107 RMODE ANY
SAVE (14,12)   SAVE REGISTERS
LR   R11,R15
USING DPKCS107,R11  SET UP PROGRAM BASE REGISTER
LR   R2,R1
USING DFHUEPAR,R2   ADDRESS USER EXIT PARAMETER LIST
*
*******************************************************************************
* <<<<< Section to be modified by the Users. >>>>> *
* START. 
*
*******************************************************************************
* */ Is the number of TD queues zero? NTD = Ø? */
* */ If yes, then we have no work to do, and exit */
* */ Return code NORMAL */
*******************************************************************************
L PMNTD,UEPMNTD   Get address of Number of TD queues
CLC Ø(2,PMNTD),=H'Ø'
BE RCNORMAL
*******************************************************************************
* */ Set up Message Number, Domain Id, and transient */
* */ data queue. */
*******************************************************************************
L PMNUM,UEPNUM   Get address of Message Number
L PMDOM,UEPMDOM  Get address of Domain Id
L PMTDQ,UEPMTDQ  Get address of transient data queue
L PMROU,UEPMROU  Get address of Route Codes array
L PMRC,UEPMRC    Get address of Number of Routes
*******************************************************************************
* */ Is Message number = 1100?... */
* */ & Domain Id = SN?... */
* */ & only TD queue = CSCS? */
* */ Yes? Then we've found what we want */
* */ No? Exit - return code NORMAL */
*******************************************************************************
CLC Ø(4,PMNUM),=F'2411' MESSAGE NUMBER = 2411?
BNE RCNORMAL
CLC Ø(2,PMDOM),=C'ZC' DOMAIN ID = ZC?
BNE RCNORMAL
CLC Ø(4,PMTDQ),=C'CSNE' TD QUEUE = CSNE?
BNE RCNORMAL
CLC Ø(2,PMNTD),=H'1' Number TD queues = 1?
BNE RCNORMAL
*******************************************************************************
* */ Having found the correct message, */
*/*| We decrease the number of transient data queues */
*/*| point of route codes to 2 */
*/*| and set the route codes to 2 and 11 */

******************************************************************************
MVC 0(2,PMNTD),=H'0'  Set Number of TD queues to 0
MVC 0(2,PMNRC),=H'2'  Set Number Route codes to 2
MVI 0(PMRou),X'02'   Set first route code to 2
MVI 1(PMRou),X'0B'   Set second route code to 11

******************************************************************************
*  END.  *
* <<<<<< Section to be Modified by the Users. >>>>> *
******************************************************************************
*  */*) Return code NORMAL */
*
******************************************************************************
*  RCFNORMAL will set the return code to UERCNORM *
******************************************************************************
*  RCFNORMAL DS 0H
  LA EXIT_RC,UERCNORM  Set the Return Code to NORMAL
  B  STEND
*
******************************************************************************
*  Restore registers, set return code, and return to user *
*  exit handler.  *
******************************************************************************
*  STEND  DS 0H
  L  R13,UEPESPA
  RETURN (14,12),RC=(15)
  LTORG
  END  DPKCS107

All that is left to do after this is to enable the exit program. The best way
to do this is in a PLTPI program. Here is the source for the one I am
using:

******************************************************************************
IDENTIFICATION DIVISION.
******************************************************************************

PROGRAM-ID.       xxxxxxxxx.
AUTHOR.            BRUCE BORCHARDT.
INSTALLATION.      xxxxxxxxxxxxxxx.
DATE-WRITTEN.      xxxxxxx.
DATE-COMPiled.     xxxxxxxx.

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

No more interrupted sleep from this problem.

Bruce Borchardt  
Senior Systems Programmer (USA)  © Xephon 2002

Contributing to CICS Update

Why not share your expertise and earn some financial reward at the same time? CICS Update is looking to swell the number of contributors who send in technical articles, hints and tips, and utility programs, etc. We would also be interested in articles about performance and tuning. If you have an idea for an article contact the editor, Trevor Eddolls, at any of the addresses shown on page 2. A copy of our Notes for Contributors is available from our Web site at www.xephon.com/nfc.
Exchanging large amounts of data between CICS and Java (or ASP) using ECI causes a huge problem. We resolved this problem by dividing data into pages and every page in the Web browser remembers a pointer to the data (which will be shown in the next call of the corresponding CICS program). I’ve selected one typical part of an application to illustrate the solution to this problem.

I002PLI

/*-------------------------------------------------------------*/
I002PLI:PROC(POINT) OPTIONS(MAIN);
/*-------------------------------------------------------------*/
/*  LIST OF TRANSACTIONS  */
/*
DCL (VERIFY, SUBSTR, ADDR, NULL, STG, CSTG) BUILTIN;
DCL S BIN FIXED(31);
DCL POINT POINTER;

DCL 1 ZONE BASED(POINT),
  2 I002IN,
    3 USERNAME CHAR(15),
    3 ACCID PIC '(9)9',
    3 CHTYPE PIC '(2)9',
    3 YEAR PIC '(4)9',
    3 STARTDATE PIC '(4)9',
    3 STOPDATE PIC '(4)9',
  2 MEMORY,
    3 M_STATE CHAR(1),
    3 M_TRNDATE PIC '(8)9',
    3 M_TRNSTAMP CHAR(17),
  2 RESULTS(640),
    3 RESDATE CHAR(10),
    3 RESDESC CHAR(25),
    3 RESAMT PIC '9,99',
  2 I002RESP PIC '9',
  2 I002MSG CHAR(80),
  2 NRESULTS PIC '3'9';

DCL DATFROM PIC '(8)9';
DCL DATTO PIC '(8)9';
DCL NRESULTSMAX BIN FIXED(31);
DCL I002INCH CHAR(38) BASED(ADDR(I002IN));
DCL KTRNCH CHAR(26);
DCL 1 KTRN BASED(ADDR(KTRNCH)),
    2 KACCID BIN FIXED(31),
    2 KTRNDATE DEC FIXED(9),
    2 KTRNSTAMP CHAR(17);
DCL KACCIDCH CHAR(4) BASED(ADDR(KACCID));
DCL ACCIDBI BIN FIXED(31);
DCL DATTMP PIC '(8)9';
DCL 1 DDDD BASED(ADDR(DATTMP)),
    2 YYYY PIC'(4)9',
    2 MM PIC'(2)9',
    2 DD PIC'(2)9';
DCL FILECICS CHAR(8);
/* FUNCTION: RETURN THE DESCRIPTION OF TRANSACTION */
DCL TRN1545 ENTRY;
DCL EVUP316 ENTRY; /* CHECK CONNECTION CICS-DB2 */
DCL INDDB2 BIN FIXED(31); /* INDICATOR FOR CONNECTION CICS - DB2 */
EXEC SQL INCLUDE TBXE008;
EXEC SQL INCLUDE I002DES;
EXEC SQL INCLUDE SQLCA;
EXEC SQL WHENEVER SQLERROR GO TO SQL_GRESKA;
EXEC SQL WHENEVER SQLWARNING GO TO SQL_GRESKA;
EXEC SQL WHENEVER NOT FOUND CONTINUE;

/******************** P R O G R A M ********************/

NRESULTSMAX = 640;

CALL EVUP316(INDDB2); /* CHECK CONNECTION CICS-DB2 */
IF INDDB2 = 0 THEN DO;
    ZONE.I002RESP = 1;
    ZONE.I002MSG = '(204) DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
    EXEC CICS RETURN;
END;

/* CHECK NUMERIC DATA */
IF VERIFY(STRTRANS(I002INCH,16,23),'1234567890') = 0 THEN DO;
    DLCTBXE008.LOGDESC = 201;
    DLCTBXE008.LOGRESP = 1;
    CALL WRITELOG;
    ZONE.I002RESP = 1;
    ZONE.I002MSG = '(201) DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
    EXEC CICS RETURN;
END;

IF CHECKACCID = 0 THEN
EXEC CICS RETURN;

IF PROC1() = Ø THEN DO;
  ZONE.I002RESP= Ø;
  DCLTBXEOB.LOGRESP = Ø;
  DCLTBXEOB.LOGDESC = '';
  CALL WRITELOG;
END;
EXEC CICS RETURN;

SQLGRESKA:
  ZONE.I002RESP = 1;
  ZONE.I002MSG = '(200) DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
  EXEC CICS RETURN;

/**************************** P R O C 1 ****************************/

PROC1: PROC RETURNS(BIN FIXED(31));
DCL DOIT(1) INIT('1' B);
ZONE.NRESULTS = Ø;
ACCIDBI = I002IN.ACCID;

SELECT(YEAR);
  WHEN(2000) FILECICS = 'TRNHI00';
  WHEN(2001) FILECICS = 'TRNHI01';
  WHEN(2002) FILECICS = 'TRNHI02';
  OTHERWISE DO;
    ZONE.I002RESP = 1;
    ZONE.I002MSG = '(205- ' || YEAR || ' ) DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
    RETURN(-1);
  END;
END;

DATTO=YEAR || SUBSTR(STOPDATE,3,2) || SUBSTR(STOPDATE,1,2);
DATFROM=YEAR||SUBSTR(STARTDATE,3,2) || SUBSTR(STARTDATE,1,2);

IF M_STATE ^= '1' THEN
DO;
  M_STATE = '2';
  KACCID = I002IN.ACCID;
  EXEC CICS STARTBR FILE(FILECICS) RIDFLD(KACCIDCH)
    KEYLENGTH(4) GENERIC GTEQ RESP(S);
  IF S ^= DFHRESP(NOTFND) THEN RETURN(Ø);
  ELSE IF S ^= DFHRESP(NORMAL) THEN DO;
    CALL CICSFAIL;
    RETURN(-1);
  END;
  KTRNDATE= Ø;
  KTRNSTAMP = '';
  EXEC CICS READNEXT FILE(FILECICS) INTO(TRNREC)
RIDFLD(KTRNCH) RESP(S);

IF S ≤ DFHRESP(NORMAL) THEN DO;
   CALL CICSFAIL;
   RETURN(-1);
END;
END;
ELSE DO: /* M_STATE = 1 */
   M_STATE = '2';
   KACCID = IØØ2IN.ACCID;
   KTRNDATE = M_TRNDATE;
   KTRNSTAMP = M_TRNSTAMP;

   EXEC CICS STARTBR FILE(FILECICS) RIDFLD(KTRNCH)
      KEYLENGTH(26) RESP(S);
   IF S = DFHRESP(NOTFND) THEN RETURN(Ø);
   ELSE IF S ≤ DFHRESP(NORMAL) THEN DO;
      CALL CICSFAIL;
      RETURN(-1);
   END;
   EXEC CICS READNEXT FILE(FILECICS) INTO(TRNREC)
      RIDFLD(KTRNCH) RESP(S);
   IF S ≤ DFHRESP(NORMAL) THEN DO;
      CALL CICSFAIL;
      RETURN(-1);
   END;
   EXIT;
END;

DO WHILE(TRNREC.ACCID = ACCIDBI & ZONE.NRESULTS < NRESULTSMAX &
   TRNREC.TRNDATE <= DATTO & DOIT = '1'B);
SELECT(IØØ2IN.CHTYPE);
   / FOR ALL TRANSACTIONS */
   WHEN(ØØ) IF ((TRNREC.TRNDATE >= DATFROM & TRNREC.TRNDATE <= DATTO)
      & (TRNREC.TRNTYPE = 2 ! TRNREC.TRNTYPE = 1)) THEN
      CALL WRITERES;
   WHEN(Ø1) /* ONLY PAY IN */
      IF ((TRNREC.TRNDATE >= DATFROM & TRNREC.TRNDATE <= DATTO) &
         TRNREC.TRNTYPE = 1) THEN
      CALL WRITERES;
   WHEN(Ø2) /* ONLY PAY OFF */
      IF ((TRNREC.TRNDATE >= DATFROM & TRNREC.TRNDATE <= DATTO) &
         TRNREC.TRNTYPE = 2) THEN
      CALL WRITERES;
   END; /* SELECT */

   EXEC CICS READNEXT FILE(FILECICS) INTO(TRNREC)
      RIDFLD(KTRNCH) RESP(S);
   IF S = DFHRESP(ENDFILE) THEN DOIT='Ø'B;
ELSE IF S = DFHRESP(NORMAL) THEN DO;
    CALL CICSFAIL;
    RETURN(-1); 
END;
END; /* WHILE */
EXEC CICS ENDBR FILE(FILECICS);
IF (TRNREC.ACCID = ACCIDBI & ZONE.NRESULT = NRESULTS &
    TRNREC.TRNDATE <= DATTO & DOIT = '1'B) THEN DO;
    M_STATE = '1';
    M_TRNDATE = TRNREC.TRNDATE;
    M_TRNSTAMP = TRNREC.TRNSTAMP;
END;
RETURN(0);
END; /* PROC1 */

CICSFAIL: PROC;
    DCL SPIC PIC '(6)9';
    IF S = DFHRESP(NOTOPEN)
        THEN DO;
            DCLTBXE08.LOGDESC = 202;
            DCLTBXE08.LOGRESP = 1;
            ZONE.I002MSG = '(202) DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
        END;
    ELSE DO;
        DCLTBXE08.LOGDESC = 203;
        DCLTBXE08.LOGRESP = 1;
        SPIC=S;
        ZONE.I002MSG = '(203- || SPIC ||
            ') DATABASE CLOSED. PLEASE, TRY AGAIN LATER.';
    END;
    ZONE.I002RESP = 1;
    CALL WRITELOG;
END; /* CICSFAIL */

WRITELOG: PROC;
    DCLTBXE08.INETUSER = I02IN.USERNAME;
    DCLTBXE08.TERMINAL = EIBTRMID;
    DCLTBXE08.INETPRG = 'I002';

    EXEC SQL INSERT INTO INETP.TBXE08
    (INETUSER,LOGRESP,LOGDESC,TERMINAL,INETPRG)
    VALUES(:DCLTBXE08.INETUSER, :DCLTBXE08.LOGRESP,
        :DCLTBXE08.LOGDESC, :DCLTBXE08.TERMINAL, :DCLTBXE08.INETPRG);

END WRITELOG;

WRITERES: PROC;
    ZONE.NRESULT = ZONE.NRESULTS + 1;
    DATMPT=TRNREC.TRNDATE;
    RESULTS(ZONE.NRESULT).RESDATE = DD''.' || MM''.' || YYYY;
    RESULTS(ZONE.NRESULT).RESDESC = TRN1545(TRNREC.TRNDCTYPE);
IF TRNREC.TRTYPE = 1 THEN
  RESULTS(ZONE.NRESULTS).RESAMT = TRNREC.TRNAMT;
END IF TRNREC.TRTYPE = 2 THEN
  RESULTS(ZONE.NRESULTS).RESAMT = -TRNREC.TRNAMT;
END WRITERES;

/*
   CHECK ACCOUNT AND USERNAME
*/

CHECKACCID: PROC RETURNS(BIN FIXED(31));
DCL 1 COM_I003,
  2 USERNAME   CHAR(15),
  2 CACCID     BIN FIXED(31),
  2 CRESP      PIC '9',
  2 MSG        CHAR(80);
COM_I003.USERNAME = I002IN.USERNAME;
COM_I003.CACCID = I002IN.ACACCID;
EXEC CICS LINK PROGRAM('I003PLI') COMMAREA(COM_I003) RESP(S);

IF S =~ DFHRESP(NORMAL) THEN DO;
  CALL CICSFIL;
  RETURN(-1);
END;
IF COM_I003.CRESP =~ Ø THEN
  DO;
    ZONE.I002RESP = 1;
    ZONE.I002MSG = COM_I003.MSG;
    RETURN(-1);
  END;
RETURN(Ø);
END CHECKACCID;

END I002PLI;

I002DES

DCL 1 TRNREC,
  2 ACCID    BIN FIXED (31),
  2 TRNDATE  DEC FIXED(9),
  2 TRNSTAMP CHAR(17),
  2 TRTYPE   DEC FIXED(1),
  2 TRNAMT   DEC FIXED(15),
  2 TRNDTYPE DEC FIXED(3),
  2 TRNDCNUM DEC FIXED(9),
  2 FILLER   CHAR(38);
I002COB

/* Our primary programming language on IBM host is PL/I. */
/* For work with Enterprise Access Builder in VAJ, we must */
/* translate only the common area in the small COBOL program. */

PROGRAM-ID. I002COB.
WORKING-STORAGE SECTION.
LINKAGE SECTION.
   01 DFHCOMMAREA.
      02 USERNAME PIC X(15).
      02 ACCID PIC X(9).
      02 CHTYPE PIC X(2).
      02 YEAR PIC X(4).
      02 STARTDATE PIC X(4).
      02 STOPDATE PIC X(4).
      02 MEMORY PIC X(26).
      02 RESULTS OCCURS 640 TIMES.
      03 RESDATE PIC X(10).
      03 RESDESC PIC X(25).
      03 RESAMT PIC X(15).
      02 I002RESP PIC 9.
      02 I002MSG PIC X(80).
      02 NRESULTS PIC 9(3).

PROCEDURE DIVISION.

Perform the following steps in VisualAge for Java (using Tools/Enterprise Access Builder):

1. Create the I002RecordType by importing from COBOL program I002COB.
2. Create the I002Record from the I002RecordType.
3. Create the I002Command using the CICSConectionSpec, ECIInteractionSpec, and I002Record.

TRANLIST.JAVA

package xweb;

public class TranList {
   public java.lang.String Msg;
   public java.lang.String Memory;
   public xweb.I002Record_RESULTS Results[];
   public int NResults;
   public TranList() {
      super();
   }
   public String fillString(String s,int l) {
      String r;
r=s;
for(int i=0;i<(1-s.length());i++)
{
    r+=" ";
}
return r;
}

public short request(String username,String accid,
        String chtype, String year,
        String start, String stop, String memory) {
    short res;
    try
    {
        xweb.I002Command com = new xweb.I002Command();
        com.setUsername(username);
        com.setAccid(accid);
        com.setChtype(chtype);
        com.setYear(year);
        com.setStartdate(start);
        com.setStopdate(stop);
        com.setMemory(memory);
        com.execute();
        res = com.getI002resp();
        if(res===(short)0)
        {
            Memory=com.getMemory1();
            Results = com.getResults();
            NResults = com.getNresults();
        }
        else
        {
            Msg=com.getI002msg();
        }
    } catch(Exception e) {
            Msg=e.toString();
            res = (short)-1;
        }
    return res;
}

public void sDay(String s, String v, javax.servlet.jsp.JspWriter out) throws Exception {
    String tmp1;
    out.println("<SELECT NAME=" + s + ">");
    for(int i=1 ; i<32 ; i++)
    {
        if(i<10) tmp1 = "0" + String.valueOf(i);
        else tmp1 = String.valueOf(i);
        out.println("<OPTION " + (v.equals(tmp1)?"SELECTED":"") + " VALUE=" + tmp1 + ">" + String.valueOf(i));
    }
}
out.println("</OPTION>");
}
out.println("</SELECT>");
}

public void sMonth(String s, String v, javax.servlet.jsp.JspWriter out)
throws Exception {
out.println("<SELECT NAME=" + s + ">");
out.println("<OPTION VALUE=01 " + (v.equals("01") ? "SELECTED:" : "") + "January</OPTION>");
out.println("<OPTION VALUE=02 " + (v.equals("02") ? "SELECTED:" : "") + "February</OPTION>");
out.println("<OPTION VALUE=03 " + (v.equals("03") ? "SELECTED:" : "") + "March</OPTION>");
out.println("<OPTION VALUE=04 " + (v.equals("04") ? "SELECTED:" : "") + "April</OPTION>");
out.println("<OPTION VALUE=05 " + (v.equals("05") ? "SELECTED:" : "") + "May</OPTION>");
out.println("<OPTION VALUE=06 " + (v.equals("06") ? "SELECTED:" : "") + "June</OPTION>");
out.println("<OPTION VALUE=07 " + (v.equals("07") ? "SELECTED:" : "") + "July</OPTION>");
out.println("<OPTION VALUE=08 " + (v.equals("08") ? "SELECTED:" : "") + "August</OPTION>");
out.println("<OPTION VALUE=09 " + (v.equals("09") ? "SELECTED:" : "") + "September</OPTION>");
out.println("<OPTION VALUE=10 " + (v.equals("10") ? "SELECTED:" : "") + "October</OPTION>");
out.println("<OPTION VALUE=11 " + (v.equals("11") ? "SELECTED:" : "") + "November</OPTION>");
out.println("<OPTION VALUE=12 " + (v.equals("12") ? "SELECTED:" : "") + "December</OPTION>");
out.println("</SELECT>");
}

IO02.JSP

<SCRIPT ID=clientEventHandlersVBS LANGUAGE=vbscript>
<!--
Sub PREVBUTTON_onclick
   history.back
End Sub
-->  
</SCRIPT>
<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>

<META http-equiv="Content-Type"
 content="text/html; charset=iso-8859-2">

<TITLE>List of transactions</TITLE>
</HEAD>

<BODY bgcolor="#FFFF00" topmargin=0 leftmargin=0>

<FORM METHOD=POST NAME=F1 ACTION=1002.jsp>

<% javax.servlet.http.HttpSession ses1 = request.getSession();
xweb.TranList tl=new xweb.TranList();

String lUserName = (String)ses1.getValue("USERNAME");
String lState = (String)ses1.getValue("STATE");
String lACCID = (String)ses1.getValue("ACCID");
// check for login
if(lUserName==null || lUserName.length()==0)
  response.sendRedirect("LOGINFAIL.HTML");
if(lState==null || lState.equals((String)"1"))
  response.sendRedirect("LOGINFAIL.HTML");
if(lACCID==null || lACCID.length()!=9)
  response.sendRedirect("LOGINFAIL.HTML");

short cicsResp;
String memory, pStart, pStop;
String pDayFrom,pMonthFrom,pDayTo,pMonthTo,pYear,pType;
boolean prev = false;
memory=tl.fillString(" ",26);
cicsResp=(short)-1;
String msg = "ENTER DATE AND TYPE OF TRANSACTION.;"

if (request.getMethod().equals("GET"))
{
pDayFrom = "@1";
pMonthFrom = "@1";
pDayTo = "@1";
pMonthTo = "@1";
pYear = "2002";
pType = "@0";
}
else
{
pDayFrom = request.getParameter("DAYFROM");
pMonthFrom = request.getParameter("MONTHFROM");
pDayTo = request.getParameter("DAYTO");
pMonthTo = request.getParameter("MONTHTO");
pYear = request.getParameter("PYEAR");
pType = request.getParameter("TYPE");
if(request.getParameter("NEXT")!=null & &
    request.getParameter("NEXT").equals("NEXT"))
{
  memory = request.getParameter("MEMORY");
}
prev = true;

pStart = pDayFrom + pMonthFrom;
pStop = pDayTo + pMonthTo;

cicsResp=t1.request(t1.fillString(lUserName,15),lACCID,
pType,pYear,pStart,pStop,memory);

if(cicsResp!=(short)0) msg=t1.Msg;
else memory=t1.Memory;

}

if(cicsResp!=(short)0) {

  
  <TABLE BORDER="2" WIDTH="100%"> 
  <TR>
  <TD ALIGN="middle" BGColor="#c0c0c0"><B><%= msg %></B></TD>
  </TR>
  </TABLE>

  
}

out.println("FROM:");
t1.sDay("DAYFROM",pDayFrom,out);
t1.sMonth("MONTHFROM",pMonthFrom,out);
t1.sDay("DAYTO",pDayTo,out);
t1.sMonth("MONTHTO",pMonthTo,out);

out.println("<SELECT NAME=PYEAR>");
out.println("<OPTION " + (pYear.equals("2000")?"SELECTED":"") + 
>2000</OPTION>");
out.println("<OPTION " + (pYear.equals("2001")?"SELECTED":"") + 
>2001</OPTION>");
out.println("<OPTION " + (pYear.equals("2002")?"SELECTED":"") + 
>2002</OPTION>");
out.println("</SELECT>");

out.println("<SELECT NAME=TYPE>");
out.println("<OPTION " + (pType.equals("00")?"SELECTED":"") + 
"VALUE=00">ALL TRANSACTIONS</OPTION>");
out.println("<OPTION " + (pType.equals("01")?"SELECTED":"") + 
"VALUE=01">PAY IN</OPTION>");
out.println("<OPTION " + (pType.equals("02")?"SELECTED":"") + 
"VALUE=02">PAY OFF</OPTION>");
out.println("</SELECT>");

out.println("<INPUT TYPE=SUBMIT VALUE=OK>");
out.println("<INPUT TYPE=HIDDEN NAME=MEMORY VALUE=" + memory + ">");
if(cicsResp==(short)0)
{
    out.println("<BR><BR>");
    if(prev)
        out.println("<INPUT ID=PREVBUTTON TYPE=BUTTON VALUE=PREV NAME=PREV>");
    if(memory.charAt(0)=='1')
        out.println("<INPUT TYPE=SUBMIT NAME=NEXT VALUE=NEXT>");
    out.println("<TABLE BORDER=1>");
    out.println("<TR>");
    out.println("<TD>DATE</TD>");
    out.println("<TD>DESCRIPTION</TD>");
    out.println("<TD>AMOUNT</TD>");
    out.println("</TR>");
    xweb.IO02Record_RESULTS res;
    for(int i=0 ; i<t1.NResults ; i++)
    {
        res=t1.Results[i];
        out.println("<TR><TD>");
        out.println(res.getResdate());
        out.println("</TD><TD>");
        out.println(res.getResdesc());
        out.println("</TD><TD ALIGN=RIGHT>");
        out.println(res.getResamt());
        out.println("</TD></TR>");
    }
    out.println("</TABLE>");
}
%
</FORM>
</BODY>
</HTML>

I002.ASP

<SCRIPT ID=clientEventHandlersVBS LANGUAGE=vbscript>
<!--
Sub PREVBUTTON_onclick
    history.back
End Sub
-->
</SCRIPT>
</HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-2">

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
<TITLE>List of transactions</TITLE>
</HEAD>

<FORM METHOD=POST NAME=F1 ACTION=f002.asp>

Dim Memory ,e1 ,f1 ,b1 ,c1 ,CicsResp , Msg, ResLen
Dim Months(12)
Dim Types(3)
Dim Prev

' Login check
If Trim(Session("USERNAME"))="" Or Session("STATE")<>1
    Or Trim(Session("ACCID"))=""
    Then
    Response.Redirect("LOGINFAIL.HTML")
End If

Months(1) = "January"
Months(2) = "February"
Months(3) = "March"
Months(4) = "April"
Months(5) = "May"
Months(6) = "June"
Months(7) = "July"
Months(8) = "August"
Months(9) = "September"
Months(10) = "October"
Months(11) = "November"
Months(12) = "December"
Types(1) = "ALL TRANSACTIONS"
Types(2) = "PAY IN"
Types(3) = "PAY OFF"
CicsResp = -1
Memory = String(26," ")
Prev = Ø

Msg = "ENTER DATE AND TYPE OF TRANSACTION."
If Request.ServerVariables("REQUEST_METHOD") = "GET" Then
    pDayFrom = "01"
    pMonthFrom = "01"
    pDayTo = "01"
    pMonthTo = "01"
    pYear = "2002"
    pType = "00"
Else
    pDayFrom = Request("DAYFROM")
    pMonthFrom = Request("MONTHFROM")
    pDayTo = Request("DAYTO")
    pMonthTo = Request("MONTHTO")
    pYear = Request("PYEAR")
    pStart = pDayFrom & pMonthFrom
    pStop = pDayTo & pMonthTo
    pType = Request("TYPE")
End If

if Request("NEXT") = "NEXT" Then
    Memory = Request("MEMORY")
    Prev = 1
End If
I002Cics(pType & pYear & pStart & pStop)
End if

If CicsResp <> Ø Then
    <TABLE BORDER="2" WIDTH="100%">
        <TR>
            <TD ALIGN="middle" BGCOLOR="#c0c0c0">%<B><% =Msg %></B>%</TD>
        </TR>
    </TABLE>
    <%
    End If

    Response.Write "FROM:"
sDay pDayFrom,"DAYFROM"
sMonth pMonthFrom,"MONTHFROM"
    Response.Write "TO:"
sDay pDayTo,"DAYTO"
sMonth pMonthTo,"MONTHTO"
    Response.Write "<SELECT NAME=PYEAR>"
    For i = 2000 to 2002
        If Trim(i) = pYear Then
            Response.Write "<OPTION SELECTED>" & i & "</OPTION>"
        Else
            Response.Write "<OPTION>" & i & "</OPTION>"
        End If
    Next
    Response.Write "</SELECT>"

    Response.Write "<SELECT NAME=TYPE>"
    For i = 0 To 2
        Tmp1=String(2-len(i),"0") & i
        If Trim(Tmp1) = pType Then
            Response.Write "<OPTION SELECTED VALUE=" & Tmp1 & ">
                & Types(i+1) & "</OPTION>"
        Else
            Response.Write "<OPTION VALUE=" & Tmp1 & ">
                & Types(i+1) & "</OPTION>"
        End If
    Next
    Response.Write "</SELECT>"

    Response.Write "<INPUT TYPE=SUBMIT VALUE=OK>"
    Response.Write "<INPUT TYPE=HIDDEN NAME=MEMORY VALUE=" & Memory & ">"
If CicsResp = 0 Then
    Response.Write "<BR><BR>
End If

If Prev <> 0 Then
    Response.Write "<INPUT ID=PREVBUTTON TYPE=BUTTON " _
        & "VALUE=PREV NAME=PREV>"
End If

If Left(Memory,1) = "1" Then
    Response.Write "<INPUT TYPE=SUBMIT NAME=NEXT VALUE=NEXT>"
End If

Response.Write "<TABLE BORDER=1>"
Response.Write "<TR>"
Response.Write "<TD>DATE</TD>"
Response.Write "<TD>DESCRIPTION</TD>"
Response.Write "<TD>AMOUNT</TD>"
Response.Write "</TR>"
Off = 64
For i= 1 To ResLen
    Response.Write "<TR><TD>"
    Response.Write b1.ExtractString(Off, 10)
    Response.Write "</TD><TD>"
    Response.Write b1.ExtractString(Off + 10, 25)
    Response.Write "</TD><TD ALIGN=RIGHT>"
    Response.Write b1.ExtractString(Off + 35, 15)
    Response.Write "</TD><TR>
    Off = Off + 50
Next
Response.Write "</TABLE>"
End If ' CicsResp = 0

Sub I002Cics(s)
    ZoneLen = 32148
    ResLen = 0
    set e1 = CreateObject("CIC.ECI")
    set f1 = CreateObject("CIC.Flow")
    set b1 = CreateObject("CIC.Buffer")
    set c1 = CreateObject("CIC.Connect")
    set ul = CreateObject("CIC.UOW")
    Zone = String(ZoneLen," ")
    cl.Details "PTEST29","",""
        pACCID = String(9-Len(Session("ACCID")),"0") _
            & Session("ACCID")
        pUserName = Session("USERNAME") _
            & String(15-Len(Session("USERNAME"))," ")
    Zone = pUserName & pACCID & s
    Zone = Zone & Memory
    Zone = Zone & String(ZoneLen-Len(Zone)," ")
    b1.SetString Zone
    On Error Resume Next
    cl.Link f1,"I002PLI",b1,ul

if Err Then
    Msg = Hex(Err.number) & Err.Description
    Err.Clear
    CicsResp=-1
Else
    Memory = b1.ExtractString (38, 26)
    Msg = b1.ExtractString (ZoneLen - 83, 80)
    ResLen = b1.ExtractString (ZoneLen - 2, 3)
    CicsResp = b1.ExtractString (ZoneLen - 84, 1)
End If

Sub sDay(ix,s)
    Response.Write "<SELECT NAME=" & s & ">
    For i = 1 to 31
        Tmpl = String(2-Len(i),"0") & i
        if Tmpl = ix Then
            Response.Write "<OPTION SELECTED>" _
            & Tmpl & "</OPTION>"
        Else
            Response.Write "<OPTION >" _
            & Tmpl & "</OPTION>"
        End if
    Next
    Response.Write "</SELECT>"
End Sub

Sub sMonth(ix,s)
    Response.Write "<SELECT NAME=" & s & ">
    For i = 1 to 12
        Tmpl = String(2-Len(i),"0") & i
        If Tmpl = ix Then
            Response.Write "<OPTION SELECTED VALUE=" _
            & Tmpl & ">" & Months(i)
        Else
            Response.Write "<OPTION VALUE=" _
            & Tmpl & ">" & Months(i)
        End if
    Next
    Response.Write "</OPTION>"
Next
    Response.Write "</SELECT> "
End Sub

%>
</FORM>
</BODY>
</HTML>

Dejan Jelic
Programmer
Postal Savings Bank (Yugoslavia)
© Xephon 2002
Ensuring absolutely trouble-free CICS operation – revisited

The February 2002 issue of CICS Update contained an article entitled Ensuring absolutely trouble-free CICS operation. Criteria 11 and 13 should have read:

11 The names for Temporary Storage Queues must be documented and correspond to the company standard (for Termid and Transaction name).

13 All programs must be independent from fixed addresses (eg Termid, Netname).

We apologise for any confusion.

Automatic PHASEIN with a simple interface between batch jobs and CICS

Our company provides information services to a group of important Italian banks. For each bank, a lot of CICS and IMS work is required. The CICS command SET PROG is RACF-restricted, accessible only by system programmers. Once, CICS application programmers had to either wait for a CICS restart or use a special RACF-enabled userid to see the new version of their program online.

With the growth in the number of concurrently running CICS sessions, we needed a tool to provide automatic PHASEIN for new program versions and to log when the new version went online for users.

Compilation jobs are automatically built by a TSO option, which uses skeletons. Programmers have to say only where the source program is located and to select for which bank they want to compile or recompile their program(s).
The last step of the compilation job consists of a program which, by calling a simple EXCI driver program, calls the CICS program that really provides PHASEIN to the desired program.

This step is built by reading a DB2 table to identify which CICS applids are to use PHASEIN. So, the batch program reads the CICS applid list and the list of programs to be compiled. Only an EXCI connection has to be defined to CICS.

The automatic PHASEIN process uses three programs:

1. In the compilation job, the program CIXXB045 reads from the JCL the DD names CIXXLIST and PGMLIST, which are the CICS applid list and the list of programs to be compiled in order to have a new version online. This program calls the simple driver program, CIXXEXCI, through the EXCI connection. We don’t mind if the PHASEIN process completes unsuccessfully, because the return code of the step is always forced to 0 or 1. This step is obviously placed after copying the load module to the desired CICS load libraries.

2. The program CIXXEXCI receives four parameters – the applid where we want to call the CICS server program, the name of the desired called CICS program, the COMMAREA length, and the COMMAREA (now max length is 24000). If the called CICS program does not complete successfully, the batch calling program receives an RC=16.

3. The CICS program CIXXNEWC receives the program name or the common prefix of the programs (maximum 100) to which the PHASEIN command is to be given. This program can also be associated with a trancode (this transaction can also be RACF-restricted). For every desired program, the PHASEIN command is given: if the response is not successful, the program tries five further PHASEIN attempts, a 50 series of RELEASE PROGRAM commands, and (only if still necessary) the last five PHASEIN commands. There is also a check to see whether more than 100 programs are using PHASEIN, and a little help regarding the syntax for CICS users. Every activity is logged in CICS MSGUSR DD.

Now we are also testing the use of the EXCI driver program as an
interface between IMS transactions and CICS server programs (without the SYNCONRETURN parameter in the EXEC CICS LINK!). It looks very interesting. In this way, IMS application programmers don’t need to know anything about the CICS EXEC of the called CICS programs, but only the record format of the data to be passed.

Here is the JCL step and the three programs’ source code.

**JCL STEP**

It needs to be in the STEPLIB of the SDFHEXCI CICS library.

```
//PHASEINC EXEC PGM=CIXXBO45
//SYSPRINT DD SYSOUT=*  
//SYSDUMP DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//CIXXLIST DD *  
APPLID1  
APPLID2  
APPLID3  
...  
//PGMLIST DD *  
PGM1  
PGM2  
...  
/*
```

**CIXXBO45 COBOL SOURCE:**

```
* *****************************************  
* CALLED BY EXCI THE CICS PROGRAM CIXXNEWC  
* *****************************************  
* PHASEIN TO MORE CICS TO MORE PROGRAMS:  
* CICS LIST FROM DD CIXXLIST  
* PROGRAM LIST FROM DD PGMLIST  
* *****************************************  
* DON'T MIND IF ERRORS IN CIXXEXCI (ALWAYS RC = 1)  
* *****************************************  
ID DIVISION.  
PROGRAM-ID. CIXXBO45.  

ENVIRONMENT DIVISION.  
CONFIGURATION SECTION.  
SPECIAL-NAMES.  
   DECIMAL-POINT IS COMMA.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
   SELECT CIXXLIST ASSIGN TO CIXXLIST
```
ORGANIZATION IS SEQUENTIAL
ACCESS IS SEQUENTIAL.
SELECT PGMLIST ASSIGN TO PGMLIST
ORGANIZATION IS SEQUENTIAL
ACCESS IS SEQUENTIAL.
SELECT STAMPA ASSIGN TO STAMPA.

DATA DIVISION.
FILE SECTION.
FD STAMPA
  BLOCK CONTAINS 0 CHARACTERS.
  01 ROW-REC PIC X(80).
FD CIXXLIST
  BLOCK CONTAINS 0 CHARACTERS.
  01 REC-CIXXLIST PIC X(80).
FD PGMLIST
  BLOCK CONTAINS 0 CHARACTERS.
  01 REC-PGMLIST PIC X(80).

WORKING-STORAGE SECTION.
  01 IND PIC 9(1) VALUE 8.
  01 INDCICS PIC 9(3) VALUE 0.
  01 INDPGM PIC 9(3) VALUE 0.
  01 CICS-COUNTER PIC 9(3) VALUE 0.
  01 PGM-COUNTER PIC 9(3) VALUE 0.
  01 APPLID PIC X(8) VALUE SPACE.
  01 PROGRAMMA PIC X(8) VALUE SPACE.
  01 PROBLEM PIC 9(8) VALUE 0.
  01 ROW PIC X(80) VALUE SPACE.
  01 CIXXEXCI PIC X(8) VALUE 'CIXXEXCI'.
  01 RISPOSTA.
     02 RESP1 PIC S9(8) COMP VALUE 0.
     02 RESP2 PIC S9(8) COMP VALUE 0.
     02 ABCODICE PIC S9(8) COMP VALUE 0.
     02 FILLER PIC S9(16).
  01 TABLE-CICS PIC X(800) VALUE SPACE.
  01 TAB-CICS REDEFINES TABLE-CICS.
     02 APPLID-TAB PIC X(8) OCCURS 100 TIMES.
  01 TABLE-PGM PIC X(800) VALUE SPACE.
  01 TAB-PGM REDEFINES TABLE-PGM.
     02 PROGRAMMA-TAB PIC X(8) OCCURS 100 TIMES.
  01 NOME-PGM PIC X(8) VALUE SPACE.
  01 TAB-PROGRAMMA REDEFINES NOME-PGM.
     02 VAL PIC X(1) OCCURS 8 TIMES.
  01 CICSAPPL PIC X(8) VALUE SPACE.
  01 CICSAPPL PIC X(8) VALUE SPACE.
01 CICS.COML PIC S9(4) COMP VALUE 0.
01 CICS.COMM.
02 PROG.COMM PIC X(8) VALUE SPACE.
02 FILLER PIC X(1) VALUE SPACE.
02 COMM.RESTO.
   10 COMM.REST01 PIC X(8) VALUE SPACE.
   10 COMM.REST02 PIC X(8) VALUE SPACE.
   10 COMM.REST03 PIC X(2) VALUE SPACE.
   10 COMM.REST04 PIC X(23973) VALUE SPACE.
01 END-FILE-CICS PIC 9 VALUE 0.
88 END-CICS VALUE 1.
01 END-FILE-PMG PIC 9 VALUE 0.
88 END-PMG VALUE 1.

LINKAGE SECTION.

**************************************************************************
PROCEDURE DIVISION.
**************************************************************************
MAIN.
OPEN INPUT CIXXLST PGMLST.
PERFORM READ-CICS-ARCHIVE
   THRU EX-READ-CICS-ARCHIVE UNTIL END-CICS.
IF END-CICS AND CICS-COUNTER EQUAL 0 THEN
   DISPLAY ' ** NO CICS IN DD CIXXLST ** ' 
   GO TO END-PROGRAM.
PERFORM READ-PMG-ARCHIVE
   THRU EX-READ-PMG-ARCHIVE UNTIL END-PMG.
IF END-PMG AND PGM-COUNTER EQUAL 0 THEN
   DISPLAY ' ** NO PGM IN DD PGMLST ** ' 
   GO TO END-PROGRAM.
MOVE 24000 TO CICS.COML.
MOVE 'CIXXNEWC' TO CICS.PROG.

MOVE 1 TO INDCICS.
PERFORM ELABORA-CICS
   THRU EX-ELABORA-CICS UNTIL INDCICS 
   GREATER THAN CICS-COUNTER.
**************************************************************************
   END-PROGRAM.
**************************************************************************
* DOPO REPETITION
   IF PROBLEM > 0 THEN
      MOVE 1 TO RETURN-CODE.
* EXCI CAN GIVE RETURN-CODE = 16
   IF RETURN-CODE > 0 THEN
      MOVE 1 TO RETURN-CODE.
   CLOSE CIXXLST PGMLST.
   STOP RUN.

ELABORA-CICS.
MOVE 1 TO INDPGM.
PERFORM ELABORA-PGM
  THRU EX-ELABORA-PGM UNTIL INDPGM
  GREATER THAN PGM-COUNTER.
  ADD 1 TO INDICICS.
EX-ELABORA-CICS.

ELABORA-PGM.
  MOVE SPACES TO NOME-PGM.
  MOVE PROGRAMMA-TAB(INDPGM) TO NOME-PGM.
  * NO LOW-VALUE (IF PGM NAME SHORTER THAN 8 CHAR)!
  KILL-LOW-VALUE.
    IF VAL(IND) = LOW-VALUES
      MOVE SPACE TO VAL(IND)
      SUBTRACT 1 FROM IND
      GO TO KILL-LOW-VALUE
    ELSE MOVE 8 TO IND.
  DISPLAY APPLID-TAB(INDICICS) PROGRAMMA-TAB(INDPGM).
  MOVE APPLID-TAB(INDICICS) TO CICSAPPL.
  MOVE SPACES TO PROG-COMM.
  MOVE PROGRAMMA-TAB(INDPGM) TO PROG-COMM.
  MOVE SPACES TO COMM-RESTO.
  CALL CIXXEXCI USING CICSAPPL CICSPROG CISCSCOML CICSCCOMM.

IF COMM-RESTO3 GREATER THAN Ø THEN
  MOVE '********** ATTENTION! **********' TO ROW
  DISPLAY ROW
  STRING
    'PROBLEMS IN PHASEIN OF PGM '
    PROG-COMM DELIMITED BY SIZE
    ' IN CICS '
    CICSAPPL DELIMITED BY SIZE
    INTO ROW
  DISPLAY ROW
  MOVE SPACES TO ROW
  STRING
    'RESP1 '
    COMM-RESTO1 DELIMITED BY SIZE
    ' RESP2 '
    COMM-RESTO2 DELIMITED BY SIZE
    ' CC '
    COMM-RESTO3 DELIMITED BY SIZE
    INTO ROW
  DISPLAY ROW
  ADD 1 TO PROBLEM
END-IF.
  DISPLAY '.
  ADD 1 TO INDPGM.
EX-ELABORA-PGM.

*
READ-CICS-ARCHIVE.
*-------------------*
   READ CIXXLIST INTO APPLID
      AT END  MOVE 1 TO END-FILE-CICS.
   IF NOT END-CICS THEN
      ADD 1 TO CICS-COUNTER
      MOVE APPLID TO APPLID-TAB(CICS-COUNTER)
   END-IF.
   EX-READ-CICS-ARCHIVE.
*
READ-PMG-ARCHIVE.
*-------------------*
   READ PGMLIST INTO PROGRAMMA
      AT END  MOVE 1 TO END-FILE-PGM.
   IF NOT END-PGM
      ADD 1 TO PGM-COUNTER
      MOVE PROGRAMMA TO PROGRAMMA-TAB(PGM-COUNTER)
   END-IF.
   EX-READ-PMG-ARCHIVE.

EXIT.

CIXXEXCI COBOL SOURCE

* **********************************************
* BATCH/CICS ROUTINE INTERFACE
* PROGRAM TO BE CALLED FROM BATCH TO EXECUTE CICS
* **********************************************
* PROGRAM BY EXCI
* EXECUTE CICS LINK WITH CICSAPPL = DESTINATION APPLID
*   CICSPROG = CALLED CICS PROGRAM
*   CICSOML = COMMAREA LENGTH
*   CICSCOMM = COMMAREA
* THESE PARAMETERS ARE RECEIVED FROM THE CALL
* IF BAD RESULT FROM CICS, RETURN CODE = 16
* ID DIVISION.
PROGRAM-ID.  CIXXEXCI.
AUTHOR.    THE MAZ.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL- NAMES.
   DECIMAL-POINT IS COMMA.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
DATA DIVISION.
FILE SECTION.

WORKING- STORAGE SECTION.
IDENTIFICATION DIVISION.

PROGRAM-ID. CIXXNEW.

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

CONFIGURATION SECTION.

COMPUTATION SECTION.

PROCEDURE DIVISION USING CICSPROGL CICSPROGL CICSCOML CICSCOMM.

* MAIN-PROGRAM.

*------------------*

EXEC CICS LINK PROGRAM(CICSPROGL)
COMMA(CICSCOMM)
LEN(CICSCOML)
APPLID(CICSPROGL)
RETCODE(RISPOSTA)
SYNCONRETURN

END-EXEC.

DISPLAY CICSPROGL
IF RESPI NOT EQUAL 0 THEN
MOVE 16 TO RETURN-CODE
DISPLAY '** PROBLEMS IN EXCI EXECUTION **'
DISPLAY '******************************************************************************'
ELSE
MOVE 0 TO RETURN-CODE
END-IF.

DISPLAY '** RESP1 **' RESP1.
DISPLAY '** RESP2 **' RESP2.
DISPLAY '** ABCODICE **' ABCODICE.
MOVE CICSCOML TO COMMALEN.
DISPLAY '** COMMAAREA LENGTH **' COMMALEN.

GOBACK.

CIXXNEW COBOL SOURCE

* ******************************************************************************
* EXECUTES PHASEIN
* ******************************************************************************
* CAN BE CALLED ALSO BY EXCI
* **************************************************************************
* AFTER 5 PHASEIN EXECUTES UNTIL 50 RELEASE FOR
* HOLD PROGRAMS, THEN OTHER 5 PHASEIN
* **************************************************************************
IDENTIFICATION DIVISION.
SKIP3
PROGRAM-ID. CIXXNEWC.
AUTHOR. THE MAZ.
SKIP3
ENVIRONMENT DIVISION.
SKIP2
CONFIGURATION SECTION.
SPECIAL-Names.
   DECIMAL-POINT IS COMMA
   C01 IS CAPO-PAGINA.
   EJECT
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SKIP3
DATA DIVISION.
SKIP3
FILE SECTION.
SKIP3
WORKING-STORAGE SECTION.

01 WHO.
   10 GIORNO PIC X(10) VALUE SPACE.
   10 FILLER PIC X VALUE SPACE.
   10 ORA PIC X(8) VALUE SPACE.
   10 FILLER PIC X VALUE SPACE.
   10 UTENTE PIC X(8) VALUE SPACE.

01 ORARIO PIC S9(15) COMP-3.
01 MESSAGGIO PIC X(80) VALUE SPACE.
01 CONTA PIC S9(5) COMP-3 VALUE +0.
01 PREFIX PIC X(8) VALUE SPACE.
01 RISPOSTA-REL PIC S9(8) COMP VALUE 0.
01 RISPOSTA PIC S9(8) COMP VALUE 0.
01 EIB-RISPOSTA PIC X(8) VALUE SPACE.
01 EIB-RISPOSTA2 PIC X(8) VALUE SPACE.
01 COUNTER PIC 9(3) VALUE 0.
01 INDEX-PGM PIC 9(3) VALUE 0.
01 INDEX-RELEASE PIC 9(3) VALUE 0.
01 INDEX-CHECK PIC 9(3) VALUE 0.
01 INDEX-CHECK-KO PIC 9(3) VALUE 0.
01 INQUIPIPROG PIC X(8) VALUE SPACE.
01 NAME PIC X(8) VALUE SPACE.
01 TESTA.
   10 TESTA-LENGTH PIC S9(4) COMP VALUE 74.
   10 FILLER PIC X(2).
   10 TESTA-TEXT PIC X(74) VALUE SPACE.
01 CTR PIC S9(4) COMP VALUE 2.
77 CTR-ROWS PIC 9(4) VALUE 2.
01 ROW.
 10 ROW-PROG PIC X(8) VALUE SPACE.
 10 FILLER PIC X(10) VALUE SPACE.
 10 ROW-STATUS PIC X(20) VALUE SPACE.
 10 ROW-ERROR PIC X(40) VALUE SPACE.
01 TABLE-PIGM.
 10 VAL OCCURS 100 TIMES PIC X(8) VALUE SPACE.
01 REC-IN.
 10 COD-TRANCODE PIC X(4) VALUE SPACES.
 10 FILLER PIC X(1) VALUE SPACES.
 10 READNAME PIC X(8) VALUE SPACES.

LINKAGE SECTION.
01 DFHCOMMAREA.
 02 COMM-PROG PIC X(8) VALUE SPACE.
 02 FILLER PIC X(1) VALUE SPACE.
 02 COMM-RESTO.
    10 COMM-REST01 PIC X(8) VALUE SPACE.
    10 COMM-REST02 PIC X(8) VALUE SPACE.
    10 COMM-REST03 PIC X(2) VALUE SPACE.
    10 COMM-REST04 PIC X(23973) VALUE SPACE.

PROCEDURE DIVISION.
  EXEC CICS ASSIGN USERID(UTENTE)
  END-EXEC.

  EXEC CICS ASKTIME ABSTIME(ORARIO)
  END-EXEC.

  EXEC CICS FORMATTIME ABSTIME(ORARIO)
    YYYYMMDD(GIORNO)
    TIME(ORA)
    DATESEP
    TIMESSEP
  END-EXEC.

  IF EIBCALEN EQUAL Ø THEN
    EXEC CICS RECEIVE
    INTO(REC-IN)
    RESP(RISPOSTA)
    END-EXEC
  ELSE
    MOVE COMM-PROG TO READNAME
  END-IF.

MAIN-PARAGRAPH.
  IF (READNAME EQUAL SPACE) AND (EIBCALEN EQUAL Ø) THEN
    PERFORM HELP-PARAGRAPH
  ELSE
    STRING
WHO              DELIMITED BY SIZE
'==> CIXXNEWC NEWCOPY PROG: '
READNAME          DELIMITED BY SIZE

INTO MESSAGGIO
MOVE SPACES     TO TESTA-TEXT
MOVE MESSAGGIO TO TESTA-TEXT
PERFORM SEND-MESSAGGIO-PARAGRAPH
PERFORM PHASEIN-PARAGRAPH
IF EIBCALEN EQUAL 0 THEN
   PERFORM WRITE-ROW-ON-SCREEN-LAST
END-IF
PERFORM SEND-MESSAGGIO-PARAGRAPH
END-IF.

GO TO GET-OUT.

PHASEIN-PARAGRAPH.
PERFORM CHECK-PRONAME-PARAGRAPH.
PERFORM UNTIL INDEX-PGM = COUNTER
ADD 1 TO INDEX-PGM
MOVE VAL(INDEX-PGM) TO NAME
EXEC CICS SET PROGRAM(NAME)
   PHASEIN
   RESP(RISPOSTA)
END-EXEC
PERFORM CHECK-PHASEIN-PARAGRAPH
END-PERFORM.

CHECK-PRONAME-PARAGRAPH.
UNSTRING READNAME DELIMITED BY '*' INTO PREFIX
COUNT IN CONTA.
IF CONTA = 0 THEN
STRING
   WHO              DELIMITED BY SIZE
   'NOT ALLOWED "*" ON FIRST POSITION' DELIMITED BY SIZE
INTO MESSAGGIO
PERFORM INVALID-REQUEST-PARAGRAPH
END-IF.
IF CONTA = 8 THEN
EXEC CICS INQUIRE PROGRAM(READNAME)
   RESP(RISPOSTA)
END-EXEC
IF RISPOSTA NOT = ZEROES THEN
   MOVE EIBRESP TO EIB-RISPOSTA
STRING
   WHO              DELIMITED BY SIZE
   '==> NOT EXISTING PGM: '
   READNAME          DELIMITED BY SIZE
INTO MESSAGGIO
PERFORM INVALID-REQUEST-PARAGRAPH
END-IF
MOVE 1 TO COUNTER
MOVE READNAME TO VAL(1)
ELSE
   EXEC CICS INQUIRE PROGRAM START END-EXEC
   PERFORM UNTIL RISPOSTA = DFHRESP(END)
   EXEC CICS INQUIRE PROGRAM(INQUIPROG) NEXT
       RESP(RISPOSTA)
   END-EXEC
   IF PREFIX EQUAL INQUIPROG(1:CONTA)
       THEN
            ADD 1 TO COUNTER
            IF COUNTER = 100 THEN
            STRING
                WHO          DELIMITED BY SIZE
                '===' COMMAND TO MORE THAN 100 PROGS: '
                READNAME     DELIMITED BY SIZE
            INTO MESSAGGIO
            PERFORM INVALID-REQUEST-PARAGRAPH
            END-IF
            MOVE INQUIPROG TO VAL(COUNTER)
   END-IF
END-EXEC
EXEC CICS INQUIRE PROGRAM END END-EXEC
IF COUNTER = 0 THEN
   STRING
        WHO          DELIMITED BY SIZE
        '===' NOT EXISTING PROGS WITH PREFIX: '
        READNAME     DELIMITED BY SIZE
   INTO MESSAGGIO
   PERFORM INVALID-REQUEST-PARAGRAPH
END-IF
END-IF.
CHECK-PHASEIN-PARAGRAPH.
MOVE NAME TO ROW-PROG.
IF RISPOSTA NOT = ZEROES THEN
MOVE 0 TO INDEX-CHECK
   PERFORM UNTIL INDEX-CHECK = 10 OR RISPOSTA = ZEROES
   IF INDEX-CHECK = 5 THEN PERFORM
      CHECK-RELEASE-PARAGRAPH
   END-IF
ADD 1 TO INDEX-CHECK
MOVE VAL(INDEX-PMG) TO NAME
EXEC CICS SET PROGRAM(NAME)
    PHASEIN
    RESP(RISPOSTA)
END-EXEC
MOVE EIBRESP TO EIB-RISPOSTA
MOVE EIBRESP TO EIB-RISPOSTA
MOVE EIBRESP2 TO EIB-RISPOSTA2
STRING
'==> CIXXNEWC NEWCOPY PROG: '
READNAME     DELIMITED BY SIZE
'RESP'      
EIB-RISPOSTA DELIMITED BY SIZE
'RESP2'      
EIB-RISPOSTA2 DELIMITED BY SIZE
'
INDEX-CHECK  DELIMITED BY SIZE
INTO MESSAGGIO
PERFORM SEND-MESSAGGIO-PARAGRAPH
END-PERFORM
END-IF.
IF RISPOSTA = ZEROES THEN
  MOVE 'OK' TO ROW-STATUS
ELSE
  MOVE '<< KO PHASEIN' TO ROW-STATUS
  ADD 1 TO INDEX-CHECK-KO
  PERFORM READ-STATUS-PROG
  IF EIBCALEN NOT EQUAL Ø THEN
    MOVE EIB-RISPOSTA TO COMM-REST01
    MOVE EIB-RISPOSTA2 TO COMM-REST02
    MOVE 'Ø3' TO COMM-REST03
  END-IF
END-IF.
STRING
  NAME     DELIMITED BY SIZE
  ,
  ROW-STATUS DELIMITED BY SIZE
INTO MESSAGGIO.
IF EIBCALEN = Ø THEN
  PERFORM WRITE-ROW-ON-SCREEN.
  PERFORM SEND-MESSAGGIO-PARAGRAPH.

CHECK-RELEASE-PARAGRAPH.
MOVE Ø TO RISPOSTA-REL.
MOVE Ø TO INDEX-RELEASE.
PERFORM UNTIL INDEX-RELEASE = 5Ø
  OR RISPOSTA-REL = DFHRESP(INVREQ)
  EXEC CICS RELEASE PROGRAM(NAME)
    RESP(RISPOSTA-REL)
END-EXEC
ADD 1 TO INDEX-RELEASE
END-PERFORM.
IF RISPOSTA-REL = DFHRESP(INVREQ) THEN
  STRING
    '==> CIXXNEWC NEWCOPY PROG: '
    READNAME     DELIMITED BY SIZE
    ' OK RELEASE AFTER '
    INDEX-RELEASE DELIMITED BY SIZE
    ' TRIES OF RELEASE PROGRAM ' DELIMITED BY SIZE
INTO MESSAGGIO
PERFORM SEND-MESSAGGIO-PARAGRAPH
ELSE
STRING
  '===> CIXXNEWC NEWCOPY PROG: '
  READNAME       DELIMITED BY SIZE
  ' KO RELEASE DOPO '
  INDEX-RELEASE  DELIMITED BY SIZE
  ' TRIES OF RELEASE PROGRAM ' DELIMITED BY SIZE
  INTO MESSAGGIO
  PERFORM SEND-MESSAGGIO-PARAGRAPH
END-IF.

INVALID-REQUEST-PARAGRAPH.
  IF EIBCALEN = Ø THEN
    EXEC CICS SEND TEXT FROM(MESSAGGIO)
        ERASE
        FREEKB
  END-EXEC
END-IF.
PERFORM SEND-MESSAGGIO-PARAGRAPH.
GO TO GET-OUT.

SEND-MESSAGGIO-PARAGRAPH.
  EXEC CICS WRITEQ TD QUEUE('CSMT')
      FROM(MESSAGGIO)
  END-EXEC.
MOVE SPACE TO MESSAGGIO.

WRITE-ROW-ON-SCREEN-LAST.
  ADD 1 TO CTR CTR-ROWS.
  MOVE SPACE TO MESSAGGIO.
  IF INDEX-CHECK-KO = Ø THEN
    STRING
      '===> COMMAND EXECUTED FOR ' DELIMITED BY SIZE
      COUNTER       DELIMITED BY SIZE
      ' PROGS ' DELIMITED BY SIZE
      INTO MESSAGGIO
    ELSE
      STRING
        '===> COMMAND EXECUTED FOR ' DELIMITED BY SIZE
        COUNTER       DELIMITED BY SIZE
        ' PROGS : NOT OK FOR ' DELIMITED BY SIZE
        INDEX-CHECK-KO DELIMITED BY SIZE
        INTO MESSAGGIO
    END-IF.
  EXEC CICS SEND TEXT
        FROM(MESSAGGIO)
        JUSTIFY(CTR)
        HEADER(TESTA)
TERMINAL
ERASE
FREEKB
END-EXEC.
EXEC CICS SEND PAGE
LAST
END-EXEC.

WRITE-ROW-ON-SCREEN.
  ADD 1 TO CTR CTR-ROWS
  IF CTR-ROWS = 18
    PERFORM NEW-PAGE
  ELSE
    EXEC CICS SEND TEXT
      FROM(ROW)
      JUSTIFY(CTR)
      HEADER(TESTA)
      TERMINAL
      ERASE
    END-EXEC
  END-IF.
MOVE SPACES TO ROW.

NEW-PAGE.
EXEC CICS SEND TEXT
  FROM(ROW)
  JUSTIFY(CTR)
  HEADER(TESTA)
  TERMINAL
  ERASE
  FREEKB
END-EXEC.
ADD 1 TO CTR.
MOVE '... TO BE CONTINUED ... (ENTER)' TO ROW.
EXEC CICS SEND TEXT
  FROM(ROW)
  JUSTIFY(CTR)
  HEADER(TESTA)
  TERMINAL
  ERASE
  FREEKB
END-EXEC.
MOVE SPACES TO ROW.
EXEC CICS SEND PAGE
END-EXEC.
EXEC CICS RECEIVE
  INTO(REC-IN)
  NOTRUNCATE
END-EXEC.
MOVE 1 TO CTR-ROWS CTR.
READ-STATUS-PROG.
EVALUATE EIBRESP
   WHEN 16
      IF EIBRESP2 = 6 THEN
         MOVE "HOLD PROG" TO ROW-ERROR
      ELSE
         MOVE "INVREQ" TO ROW-ERROR
      END-IF
   END-IF
WHEN 17   MOVE "IOERR" TO ROW-ERROR
WHEN 27   MOVE "PGMIDERR" TO ROW-ERROR
WHEN 70   MOVE "NOTAUTH" TO ROW-ERROR
WHEN OTHER MOVE "*****" TO ROW-ERROR
END-EVALUATE.

HELP-PARAGRAPH.
MOVE SPACE TO TESTA-TEXT.
MOVE REC-IN TO TESTA-TEXT.
MOVE ' COMMAND SINTAX IS:' TO ROW.
PERFORM WRITE-ROW-ON-SCREEN.
MOVE SPACES TO ROW.
PERFORM WRITE-ROW-ON-SCREEN.
MOVE ' $NEW PROGNAME' TO ROW.
PERFORM WRITE-ROW-ON-SCREEN.
MOVE SPACES TO ROW.
PERFORM WRITE-ROW-ON-SCREEN.
MOVE ' YOU CAN USE SPECIAL CHAR "*" TO ROW.
ADD 1 TO CTR CTR-ROWS.
EXEC CICS SEND TEXT
   FROM(ROW)
   JUSTIFY(CTR)
   HEADER(TESTA)
   TERMINAL
   ERASE
   FREEKB
END-EXEC.
EXEC CICS SEND PAGE
LAST
END-EXEC.
GO TO GET-OUT.

GET-OUT.
EXEC CICS RETURN
END-EXEC.
GOBACK.
STOP RUN.

Frances Comazzon
IMS System Programmer
UniCredit Servizi Informativi (Italy)  © Xophon 2002
CICS questions and answers

Q Is there a way to determine the MVS SYSID from a CICS program?
A The following code will get you the SMCASID, which contains the SMF ID of the MVS system:

```
L R1,CVTPTR
USING CVTMAP,R1
L R1,CVTSMCA
USING SMCA,Base,R1
MVC myfield,SMCASID ===> .
.
.
CVT DSECT=YES,LIST=YES
IEESMCA
```

Q Is there a way to remove a TOR from a VTAM Generic Resource group but leave the CICS region and the existing sessions active?
A CEMT SET VTAM DEREGISTERED. However, there is no command to re-join the group – you need to Close/Open the ACB (or re-start CICS).

Q If I restart two APPC Connected CICS regions but I only start one of them COLD, I get the CICS message DFHRS2111, and the connection between the two regions sits in a pending state. Using CEMT I have to issue a ‘notpending’ command to get the connection working. Is there any way to automate this command?
A Review the new XLNACTION option on the connection definition – the FORCE option will automatically implement the manual ‘notpending’ should a new logname be received from the connected CICS region.

If you have any CICS-related questions, please send them in and we will do our best to find answers. Alternatively, e-mail them directly to cicsq@xephon.net.

© Xephon 2002
CICS news

Rosebud Management Systems has announced the latest version of its Eden Server re-hosting system for legacy CICS and batch COBOL systems, which works with Micro Focus Net Express and Windows. The multi-tiered application server and client environment is aimed at mainframe sites.

The latest release includes new features to help expand applications with GUI interfaces across LANs and the Internet and to provide better inter-application connectivity without re-writing or intrusive code changes.

With the addition of the new Eden Client, traditional green screens are dynamically interpreted and supported as GUI windows. This new Eden Client support is included in several different flavours and supports the new Eden Thin Client interface, allowing Eden-based CICS applications to run as native Windows GUIs across the Internet.

There is also a new suite of callable APIs for use in developing server based add-ons written in COBOL, C, VB, and other languages.

For further information contact: Rosebud Management Systems, 216 Pleasant Hill Road, Flanders, NJ 07836, USA.
Phone: (973) 252 4150.

Key functions include support for applications written in Java, support for Windows 2000, availability, scalability, and recovery of applications from failure, interoperability with WebSphere Application Server and CORBA-compliant servers, and operation with current levels of database managers and languages.

Enhancements to the CICS execution environment include support for Java and an improved ORB, enabling Java applications to interoperate with those under WebSphere or other CORBA servers.

For further information contact your local IBM representative.

* * *

IBM has announced TXSeries for Multiplatforms (TXSeries) V5 for connecting to different client environments, Web-enabling TXSeries-based applications, and creating applications using WebSphere, CICS Transaction Gateway, and TXSeries.

IBM has announced Version 4.1 of its WebSphere Studio Application Developer Integration Edition for Windows, for building, testing, integrating, and deploying J2EE applications.

Among its functions are a Java development environment that includes support for JDK 1.3, a configurable runtime, incremental compilation, scrapbook, dynamic debugging, and a Java text editor. It also has Wizards and visual tools to help create adapters, Web services, JavaBeans, EJBs, and JavaServer Pages.

It also provides development connectors for CICS, IMS, and Host on Demand.

For further information contact your local IBM representative.

* * *

IBM has announced TXSeries for Multiplatforms (TXSeries) V5 for connecting to different client environments, Web-enabling TXSeries-based applications, and creating applications using WebSphere, CICS Transaction Gateway, and TXSeries.

For further information contact your local IBM representative.