189

CICS

August 2001

update

In this issue

CICS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

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

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/cicsupdate.html; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

Printed in England.

CICS Integrated Translator for COBOL and PL/I

CICS Transaction Server Version 2 Release 1 for z/OS was announced on 13 March 2001 with General Availability of 30 March 2001. CICS TS Version 2 Release 1 supports the Enterprise Java Bean Version 1.1 level of architecture. It includes the exploitation of a new optimized Java Virtual Machine (also known as the persistent reusable JVM) developed by the Java Technology Centre in Hursley. It also includes enhanced CORBA function, new capabilities from CICSPlex SM for workload management of enterprise beans (applications written to the EJB specification), JDBC/SQLJ access to DB2 data, and JCICS extensions including access to VSAM data. A new function called the CICS Connector for CICS TS enables inter-operability between enterprise beans, applications, and data which use other programming models, and, finally, the subject of this article, the CICS Integrated Translator for COBOL and PL/I. For further details on all of these enhancements see: http://www-4.ibm.com/software/ts/cics/v2/.

Besides the function detailed above, you should read all the currency statements in the announcement letter, some of which are detailed below.

CURRENCY STATEMENTS

The following sentences appear in the Software Requirements section of the above announcement: “Support remains in CICS TS Version 2 Release 1 for the earlier (pre-Language Environment) compilers:

- OS/VS COBOL (5740-CB1, 5740-LM1, and 5734-CB4).
- VS COBOL II (5668-958, 5688-023, and 5688-022).
- OS PL/I Version 2 (5668-910).
- SAA AD/Cycle C/370 (5688-216).

Translator support for these pre-Language Environment compilers will be withdrawn in CICS TS Version 2 Release 2. It is planned that run-time support for load modules created using these compilers, that

do not use Language Environment, will be withdrawn in the release following CICS TS Version 2 Release 2".

This means you will not be able to translate CICS application programs for non-LE conforming compilers in CICS TS Version 2 Release 2 or later, and that the run-time support for those load modules produced by those compilers will be withdrawn in the release that follows CICS TS Version 2 Release 2. You will probably already know that service for almost all the non-LE compilers and libraries was withdrawn on 31 March 2001.

It may be apposite to remind you that in 1999 it was announced that CICS/ESA Version 4 Release 1 will go out of service on 31 December 2002.

CICS TRANSLATORS

Ever since the Command Level API was created, CICS application programs have to be translated before they can be compiled. The translators scan the input source to find EXEC CICS commands, comment them out, and generate CALL statements adding any defaults appropriate to the language with the arguments from the EXEC CICS command. The CICS-supplied catalog procedures for compiling user CICS application programs all contain an initial job step that invokes the stand-alone translator appropriate to the language compiler invoked in the following job step.

When the stand-alone translators add the generated CALL statements, they neccessarily change the line numbers in source programs, which means that an intermediate listing with the translator-generated CALL statements must be used when debugging an application program rather than the source program the programmer wrote.

CICS INTEGRATED TRANSLATOR

With the Integrated Translator announced and shipped in CICS TS Version 2 Release 1, COBOL and PL/I application development is made easier. This is because the compiler reads the real source, which includes the EXEC CICS commands, so it understands which

statements have been generated from the translation, so now there needs to be only one listing – the original source statements the programmer wrote – when debugging.

Another usability problem with the stand-alone CICS translator is that, if the source program contains copybooks containing CICS commands, you will have to translate them separately before compilation of the program in which they will be included. With the Integrated Translator, because the compiler is in control, copybooks and INCLUDE files will be included automatically during compilation. As a result, the new process should be less error-prone because it is no longer necessary to remember to translate changed members included in the source separately.

The software requirements for the integrated translator are as follows:

- IBM COBOL for OS/390 and VM, Version 2 Release 2 5648-A25, for COBOL programs with APAR PQ45462.
- IBM VisualAge PL/I for OS/390, Version 2 Release 2.1 5655-B22, for PL/I programs with APAR PQ45562.

The CICS-supplied procedures that can be used to translate, compile, and link-edit application programs continue to be supported. However, it is recommended that these procedures are not used if the integrated CICS translator is supported by the language compiler.

The LE-conforming language compilers, which support the integrated CICS translator, scan the application source and call the integrated CICS translator at relevant points.

CICS INTEGRATED TRANSLATOR OPTIONS

There are no changes to CICS external interfaces for the integrated CICS translator, but compilers that support the Integrated Translator make some CICS-supplied catalog procedures redundant. Many of the translator options, such as those associated with translator listings, do not apply when using the integrated CICS translator. The translator options that can be used effectively with the integrated CICS translator are:

- APOST or QUOTE
- CPSM or NOCPSM
- CICS, DBCS, DEBUG, or NODEBUG
- DLI, EDF or NOEDF
- FEPI or NOFEPI
- GRAPHIC, LENGTH, or NOLENGTH
- LINKAGE or NOLINKAGE
- NATLANG, OOCOBOL, SP, and SYSEIB.

The traditional CICS-supplied catalog procedures that are used to translate, compile, and link-edit application programs continue to be supported, but it is recommended that these procedures are not used if the language compiler in use supports the integrated CICS translator.

USING THE PL/I CICS INTEGRATED TRANSLATOR

To use the integrated translator for PL/I you must specify the compiler option SYSTEM(CICS).

To specify CICS translator options when using PL/I with the integrated translator, specify the compiler pre-processor option, PP(CICS), followed by the CICS translator options inside parentheses. For example:

```
PP(CICS('option1 option2 ...optionn'))
```

where optionn is the *n*th option to be passed to the CICS translator. You can specify the PP compiler option wherever PL/I compiler options can be specified.

For more information on the PL/I implementation, see APAR PQ45562.

USING THE PL/I AND COBOL INTEGRATED TRANSLATOR

To invoke the integrated translator when using the COBOL compiler, specify the compiler option CICS, and follow this with the CICS translator options inside parentheses.

Note: the XOPTS translator option is not accepted when using the integrated CICS translator and must be changed to the CICS compiler option.

To use the integrated CICS translator for COBOL the compiler options CICS, LIB, NODYNAM, RENT, and NOANALYZE must be in effect. You cannot use SIZE(MAX) when compiling large programs using CICS statements because storage must be left in the user region for the integrated CICS translator services.

Options may be passed using a PARM string, for example:

```
// PARM='NODYNAM,LIB,OBJECT,RENT,MAP,XREF,CICS(''NOEDF,SP'')'
```

If the CICS translator options are specified in a PARM string, they must be enclosed in quotes and, since in the above example the options are within a quoted string, two single quotes result.

For more information on the CICS compiler option for COBOL, see APAR PQ45462.

RETROFIT OF INTEGRATED TRANSLATOR TO CICS TS 1.3

IBM hopes to retrofit Integrated Translator to CICS TS 1.3 as APAR PQ47120.

Andy Krasun
IBM (UK)

Testing CICS-PL/I transactions

The June issue (187) of *CICS Update* contained an article describing a system to trace CICS programs, which can be used in a test as well as a production environment. This month we are able to publish the programs. The first program writes an entry into the log file (or DB2 table) depending on the value of a field from another file (or DB2 table) and some other fields which indicate whether the program is for testing or not.

A second program writes an entry about program testing characteristics

and at the same time a transaction to view the log of a particular program.

PROGRAM 1

```
PRTEST: PROC(POINT) OPTIONS(MAIN);
/*************************************************************/
/* DESCRIPTION: MAKING TESTING PROGRAMS ENTRIES WITH THEIR FLAGS*/
/*              SETTING FLAGS ON/OFF, FORMATTED READING OF LOG  */
/* KSDS VSAM: PROTEST                                          */
/*************************************************************/

 DCL (SELECT,SUBSTR,DATE,ADDR,BASED,VERIFY) BUILTIN;
 DCL (STG,NULL,CSTG,LENGTH,LOW,DATETIME) BUILTIN;
 DCL POINT PTR;
 DCL 1 COM_AREA BASED(POINT),
       2 COM_PROG    CHAR(8),
       2 PAGE        BIN FIXED(15),
       2 TOTAL_PAGES BIN FIXED(15);

 DCL UTIME          DEC FIXED(15);
 DCL NAS_DATE       CHAR(1Ø);
 DCL USA_DATE       CHAR(1Ø);
 DCL N_TIME         CHAR(8);
 DCL RES            BIN FIXED(31);
 DCL I              BIN FIXED(31);
 DCL ENTRY_NUM      BIN FIXED(31);
 DCL N_USER         CHAR(8) INIT(' ');
 DCL MSG            CHAR(65) INIT(' ');
 DCL TEMP           CHAR(8);
 DCL PAGE_P3        PIC'999';
 DCL TOTAL_PAGES_P3 PIC'999';

 RES = Ø;
 UTIME     = Ø;
 NAS_DATE = ' ';
 USA_DATE = ' ';
 N_TIME     = ' ';
 ENTRY_NUM = Ø;
 PAGE_P3 = Ø;
 TOTAL_PAGES_P3 = Ø;
/*************************************************************/
/*** INCLUDE *************************************************/
/*************************************************************/
 %INCLUDE MAPXXXX;     /* PL/I MAP DESCRIPTION                */
 %INCLUDE RD_PROTEST   /* RECORD DESCRIPTION OF FILE PROTEST   */
 %INCLUDE RD_PROGLOG   /* RECORD DESCRIPTION OF FILE PROGLOG   */
 %INCLUDE DFHAID;
```

```
  %INCLUDE DFHBMSCA;
 /*****************************************************************/
 /**********************    PROGRAM    ****************************/
 /*****************************************************************/
  EXEC CICS ASSIGN USERID(N_USER);
Ø99
  TEMP = EIBTRMID || 'TRXX';
  IF SUBSTR(N_USER,1,3) ¬= 'XXX' & SUBSTR(N_USER,1,3) ¬= 'YYY' THEN
  DO;
     MSG='*** YOU ARE NOT AUTHORIZED FOR TRANSACTION ***';
     EXEC CICS SEND FROM(MSG) ERASE;
     EXEC CICS RETURN;
  END;
  IF EIBCALEN=Ø THEN
  DO;
     ALLOCATE COM_AREA;
     COM_AREA.COM_PROG=' ';
     COM_AREA.PAGE=Ø;
     COM_AREA.TOTAL_PAGES = Ø;
     MPROGL = -1;
     CALL NMAPA(' ',1);
  END;

  EXEC CICS HANDLE ABEND PROGRAM('PROGZZZZ');
  EXEC CICS HANDLE CONDITION PGMIDERR(PGMERR);
  EXEC CICS ASKTIME ABSTIME(UTIME) RESP(RES);

  EXEC CICS FORMATTIME ABSTIME(UTIME)
            DATESEP('.') DDMMYYYY(NAS_DATE)
            TIME(N_TIME) TIMESEP RESP(RES);

  EXEC CICS FORMATTIME ABSTIME(UTIME)
            DATESEP('.') YYYYMMDD(USA_DATE)
            TIME(N_TIME) TIMESEP RESP(RES);

SELECT (EIBAID);
    WHEN (DFHPF3)
    DO;
        EXEC CICS DELETEQ TS QUEUE(TEMP) RESP(RES);
        IF RES¬=DFHRESP(NORMAL) THEN;

        MSG='*** END ***';
        EXEC CICS SEND FROM(MSG) ERASE;
        EXEC CICS RETURN;
    END;
    WHEN (DFHENTER)
    DO;
       PAGE = Ø;
       TOTAL_PAGES = Ø;
       EXEC CICS RECEIVE MAP('MAPXXXX') MAPSET('MAPXXXX') RESP(RES);
```

```
   IF RES = DFHRESP(MAPFAIL)
   THEN
   DO;
      MPROGL = -1;
      CALL NMAPA(' ',1);
   END;
   IF COM_AREA.COM_PROG=MPROGI THEN CALL ENTERCHANGE;
                               ELSE
                               DO;
                                   COM_AREA.COM_PROG=MPROGI;
                                   CALL FILL_MAP;
                                   MTESTL = -1;
                                   CALL NMAPA(' ',2);
                               END;
END;
WHEN (DFHPF12)
DO;
   EXEC CICS RECEIVE MAP('MAPXXXX') MAPSET('MAPXXXX') RESP(RES);
   IF RES = DFHRESP(MAPFAIL)
   THEN
   DO;
      MPROGL = -1;
      CALL NMAPA(' ',1);
   END;
   IF COM_PROG ¬= ' '
   THEN
   DO;
      CALL VALID_DATA(6Ø);
      CALL VALID_DATA(12);
      CALL MAKE_TEMP_STOR;
   END;
   ELSE
   DO;
      MPROGL = -1;
      CALL NMAPA('YOU HAVE TO CHOOSE THE TRANSACTION FIRST.',1);
   END;
END;
WHEN (DFHPF7)
DO;
    IF PAGE <= 1 | TOTAL_PAGES <= 1 | COM_PROG = ' ' THEN
    DO;
       MPROGL = -1;
       CALL NMAPA('THE PREVIOUS PAGE DOESN'T EXIST.',2);
    END;
    PAGE = PAGE - 1;
    EXEC CICS READQ TS QUEUE(TEMP) INTO(MAPXXXXO) ITEM(PAGE);
    MPROGL = -1;
    PAGE_P3 = PAGE;
    TOTAL_PAGES_P3 = TOTAL_PAGES;
    CALL NMAPA('PAGE ' || PAGE_P3 || '/' || TOTAL_PAGES_P3,2);
```

```
END;
WHEN (DFHPF8)
DO;
    IF PAGE = TOTAL_PAGES | TOTAL_PAGES <= 1 |
       COM_PROG = ' ' THEN
    DO;
       MPROGL = -1;
       CALL NMAPA('THE NEXT PAGE DOESN'T EXIST.',2);
    END;
    PAGE = PAGE + 1;
    EXEC CICS READQ TS QUEUE(TEMP) INTO(MAPXXXXO) ITEM(PAGE);
    MPROGL = -1;
    PAGE_P3 = PAGE;
    TOTAL_PAGES_P3 = TOTAL_PAGES;
    CALL NMAPA('PAGE ' || PAGE_P3 || '/' || TOTAL_PAGES_P3,2);
END;
WHEN (DFHPF24)
DO;
    MPDATUMO = USA_DATE;
    MPVREMEO = '00:00:00:000';
    DO I=1 TO 12;
       REDLOG2(I).REDLOGO=' ';
    END;
    TEST_KEY = COM_PROG;
    IF COM_PROG = ' ' THEN
    DO;
       EXEC CICS STARTBR FILE('PROTEST') RIDFLD(TEST_KEY)
                 KEYLENGTH(8) GTEQ RESP(RES);
       IF RES=DFHRESP(NOTOPEN) THEN
       DO;
          MPROGL = -1;
          CALL NMAPA('FILE PROTEST IS CLOSED.',2);
       END;
       IF RES=DFHRESP(NOTFND) THEN
       DO;
         MPROGL = -1;
         CALL NMAPA('ENTRY DOESN'T EXIST.',2);
       END;
       IF RES¬=DFHRESP(NORMAL) THEN
       DO;
            MPROGL = -1;
            CALL NMAPA('ERROR WITH READING FILE PROTEST.',2);
       END;

       EXEC CICS READNEXT FILE('PROTEST') INTO(TEST_C70)
                   RIDFLD(TEST_KEY) KEYLENGTH(8) RESP(RES);
       IF RES¬=DFHRESP(NORMAL)
       THEN
       DO;
            MPROGL = -1;
```

```
          CALL NMAPA('ERROR WITH READING FILE PROTEST.',2);
      END;
      ELSE /* READNEXT PROTEST OK*/
      DO;
        MPROGO        = PROGRAM;
        MTRANO        = STR_PROTEST.TRANS_ID;
        MTUSERO       = STR_PROTEST.TESTNI_USER;
        MTERMO        = TESTNI_TERMINAL;
        MRACUNO       = TESTNI_ACCOUNT;
        MTESTO        = PRO_TEST;
        MDATUMO       = DATE_OF_CHANGE;
        MUSERO        = USER_OF_CHANGE;
        COM_PROG      = PROGRAM;
        COM_AREA.PAGE=Ø;
        COM_AREA.TOTAL_PAGES = Ø;
        EXEC CICS ENDBR DATASET('PROTEST');
        MPROGL = -1;
        CALL NMAPA(' ',2);
      END;
   END;  /* IF COM_PROG = ' ' */
   ELSE
   DO;
      EXEC CICS STARTBR FILE('PROTEST') RIDFLD(TEST_KEY)
                KEYLENGTH(8) EQUAL RESP(RES);
      IF RES=DFHRESP(NOTOPEN) THEN
      DO;
         MPROGL = -1;
         CALL NMAPA('FILE PROTEST IS CLOSED.',2);
      END;
      IF RES=DFHRESP(NOTFND) THEN
      DO;
        MPROGL = -1;
        CALL NMAPA('NO PREVIOUS ENTRY.',2);
      END;
      IF RES¬=DFHRESP(NORMAL) THEN
      DO;
          MPROGL = -1;
          CALL NMAPA('ERROR WITH READING FILE PROTEST.',2);
      END;

      EXEC CICS READNEXT FILE('PROTEST') INTO(TEST_CØ)
                 RIDFLD(TEST_KEY) KEYLENGTH(8) RESP(RES);
      IF RES¬=DFHRESP(NORMAL)
      THEN
      DO;
          MPROGL = -1;
          CALL NMAPA('ERROR WITH READING FILE PROTEST.',2);
      END;
      /* NEXT READNEXT */
      EXEC CICS READNEXT FILE('PROTEST') INTO(TEST_CØ)
```

```
                    RIDFLD(TEST_KEY) KEYLENGTH(8) RESP(RES);
          IF RES=DFHRESP(ENDFILE)
          THEN
          DO;
               MPROGL = -1;
               CALL NMAPA('NO MORE ENTRIES.',2);
          END;
          IF RES¬=DFHRESP(NORMAL)
          THEN
          DO;
               MPROGL = -1;
               CALL NMAPA('ERROR WITH READING FILE PROTEST.',2);
          END;
          ELSE /* OK READNEXT PROTEST */
          DO;
            MPROGO        = PROGRAM;
            MTRANO        = STR_PROTEST.TRANS_ID;
            MTUSERO       = STR_PROTEST.TESTNI_USER;
            MTERMO        = TESTNI_TERMINAL;
            MRACUNO       = TESTNI_ACCOUNT;
            MTESTO        = PRO_TEST;
            MDATUMO       = DATE_OF_CHANGE;
            MUSERO        = USER_OF_CHANGE;
            COM_PROG      = PROGRAM;
            COM_AREA.PAGE=Ø;
            COM_AREA.TOTAL_PAGES = Ø;
            EXEC CICS ENDBR DATASET('PROTEST');
            MPROGL = -1;
            CALL NMAPA(' ',2);
          END;
       END;
   END;  /* WHEN (DFHPF24) */
   OTHERWISE
   DO;
     MPROGL = -1;
     CALL NMAPA('WRONG PFKEY!',2);
   END;
 END;

KIKSGRE: PROC; MSG = 'ERROR IN CICS!';
  EXEC CICS SEND FROM(MSG) ERASE;
  EXEC CICS RETURN;
END KIKSGRE;

NOFILE: PROC;
  MSG = 'FILE PROTEST DOESN'T EXIST.';
  EXEC CICS SEND FROM(MSG) ERASE;
  EXEC CICS RETURN;
END NOFILE;
```

```
ENTERCHANGE: PROC;

   CALL VALID_DATA(60);

   CALL AZUR_PROTEST;
   CALL FILL_MAP;
   MPROGL = -1;
   CALL NMAPA('RECORD IS SAVED!',2);

END ENTERCHANGE;

FILL_MAP: PROC;

   PROGRAM = MPROGI;
   EXEC CICS READ DATASET('PROTEST') INTO(TEST_C70)
                         RESP(RES) RIDFLD(TEST_KEY);

   IF RES = DFHRESP(FILENOTFOUND) THEN CALL NOFILE;
   IF RES = DFHRESP(NOTOPEN) THEN
   DO;
      MPROGL = -1;
      CALL NMAPA('FILE ' || EIBDS || 'IS SLOSED.',1);
   END;
   IF RES = DFHRESP(NORMAL) THEN
   DO;
     MPROGO        = PROGRAM;
     MTRANO        = STR_PROTEST.TRANS_ID;
     MTUSERO       = STR_PROTEST.TESTNI_USER;
     MTERMO        = TESTNI_TERMINAL;
     MRACUNO       = TESTNI_ACCOUNT;
     MTESTO        = PRO_TEST;
     MDATUMO       = DATE_OF_CHANGE;
     MUSERO        = USER_OF_CHANGE;
     MPDATUMO = USA_DATE;
     MPVREMEO = '00:00:00:000';
     DO I=1 TO 12;
        REDLOG2(I).REDLOGO=' ';
     END;
   END;
   IF RES = DFHRESP(NOTFND)
   THEN
   DO;
     MTRANO        = ' ';
     MTERMO        = ' ';
     MTUSERO       = ' ';
     MRACUNO       = 0;
     MTESTO        = 1;
     MDATUMO       = NAS_DATE;
     MUSERO        = N_USER;
     MPDATUMO      = USA_DATE;
```

```
    MPVREMEO      = '00:00:00:000';
    DO I=1 TO 12;
       REDLOG2(I).REDLOGO=' ';
    END;
  END;
END FILL_MAP;

AZUR_PROTEST: PROC;
  PROGRAM = MPROGI;
  EXEC CICS READ DATASET('PROTEST') INTO(TEST_C70)
                        RESP(RES) RIDFLD(TEST_KEY) UPDATE;

  IF RES = DFHRESP(FILENOTFOUND) THEN CALL NOFILE;
  IF RES = DFHRESP(NOTOPEN) THEN
  DO;
     CALL FILL_MAP;
     MPROGL = -1;
     CALL NMAPA('FILE ' || EIBDS || 'IS CLOSED.',2);
  END;

  IF RES = DFHRESP(NORMAL)
                    /* WE NEED TO UPDATE */
  THEN
  DO;
     PROGRAM                 = MPROGI;
     STR_PROTEST.TRANS_ID    = MTRANI;
     STR_PROTEST.TESTNI_USER = MTUSERI;
     TESTNI_TERMINAL         = MTERMI;
     TESTNI_ACCOUNT          = MRACUNI;
     PRO_TEST                = MTESTI;
     DATE_OF_CHANGE          = NAS_DATE;
     USER_OF_CHANGE          = N_USER;
     STR_PROTEST.REZERVA     = ' ';

     EXEC CICS REWRITE DATASET('PROTEST') FROM(TEST_C70)
                         RESP(RES);
     IF RES = DFHRESP(NORMAL)
     THEN
     DO;
           /*           */
     END;
     ELSE
     DO;
         CALL FILL_MAP;
         MPROGL = -1;
         CALL NMAPA('RECORD ISN'T SAVED!',2);
      END;
  END;

  IF RES = DFHRESP(NOTFND)
```

```
  THEN             /* RECORD DOESN'T EXIST WE NEED TO WRITE NEW */
  DO;
     PROGRAM                 = MPROGI;
     STR_PROTEST.TRANS_ID    = MTRANI;
     STR_PROTEST.TESTNI_USER = MTUSERI;
     TESTNI_TERMINAL         = MTERMI;
     TESTNI_ACCOUNT          = MRACUNI;
     PRO_TEST                = MTESTI;
     DATE_OF_CHANGE          = NAS_DATE;
     USER_OF_CHANGE          = N_USER;
     STR_PROTEST.REZERVA     = ' ';

     EXEC CICS WRITE DATASET('PROTEST') FROM(TEST_C7Ø)
                              RESP(RES) RIDFLD(TEST_KEY);
     IF RES ¬= DFHRESP(NORMAL)
     THEN
     DO;
          CALL FILL_MAP;
          MPROGL = -1;
          CALL NMAPA('RECORD ISN'T SAVED!',2);
     END;

  END;                        /* END FOR NOTFND PROTEST */

  IF RES ¬= DFHRESP(NORMAL) & RES¬= DFHRESP(NOTFND)
  THEN CALL KIKSGRE;

END AZUR_PROTEST;

NMAPA: PROC(POR,TIP);

  DCL POR    CHAR(78);
  DCL TIP    DEC FIXED(1);

  PORUKANO=POR;
  SELECT (TIP);
    WHEN(1)
    DO;
       EXEC CICS SEND MAP('MAPXXXX') MAPSET('MAPXXXX')
                                     FREEKB CURSOR ERASE;
       EXEC CICS RETURN COMMAREA(COM_AREA) TRANSID('TRXX');
    END;
    WHEN(2)
    DO;
       EXEC CICS SEND MAP('MAPXXXX') MAPSET('MAPXXXX')
                                     FREEKB CURSOR;
       EXEC CICS RETURN COMMAREA(COM_AREA) TRANSID('TRXX');
    END;
    OTHER
    DO;
```

```
        EXEC CICS SEND MAP('MAPXXXX') MAPSET('MAPXXXX')
                              DATAONLY FREEKB CURSOR;
        EXEC CICS RETURN COMMAREA(COM_AREA) TRANSID('TRXX');
      END;
    END;

  END NMAPA;

  VALID_DATA: PROC(PF_KEY_F3);
    DCL PF_KEY_F3 DEC FIXED(3);
    SELECT(PF_KEY_F3);
       WHEN(6Ø)  /* ENTER */
       DO;
           IF (MPROGL = Ø | MPROGI = ' ')
           THEN
           DO;
              MPROGL = -1;
              CALL NMAPA('WRONG PROGRAM!',2);
           END;

           IF (MTRANL = Ø | MTRANI = ' ')
           THEN
           DO;
              MTRANL = -1;
              CALL NMAPA('WRONG TRASACTION!',2);
           END;

           IF (MTERML = Ø | MTERMI = ' ')
           THEN
           DO;
              MTERML = -1;
              CALL NMAPA('WRONG TERMINAL!',2);
           END;

           IF (MTUSERL = Ø | MTUSERI = ' ')
           THEN
           DO;
              MTUSERL = -1;
              CALL NMAPA('WRONG USER!',2);
           END;

           IF (MRACUNL = Ø | VERIFY(MRACUNI,' Ø123456789') ¬= Ø)
           THEN
           DO;
              MRACUNL = -1;
              CALL NMAPA('WRONG ACCOUNT!',2);
           END;

           IF MTESTL = Ø | VERIFY(MTESTI,'Ø1')¬=Ø
           THEN
```

```
            DO;
               MTESTL = -1;
               CALL NMAPA('YOU CAN PUT ONLY Ø/1!',2);
            END;

         END;
         WHEN(12)  /* PF12  */
         DO;
            IF MPDATUML = Ø | MPDATUMI = ' ' |
                VERIFY(MPDATUMI,'.Ø123456789') ¬= Ø |
                SUBSTR(MPDATUMI,5,1) ¬= '.' |
                SUBSTR(MPDATUMI,8,1) ¬= '.'
            THEN
            DO;
               MPDATUML = -1;
               CALL NMAPA('WRONG START DATE!',2);
            END;

            IF MPVREMEL = Ø | MPVREMEI = ' ' |
                VERIFY(MPVREMEI,':Ø123456789') ¬= Ø |
                SUBSTR(MPVREMEI,3,1) ¬= ':' |
                SUBSTR(MPVREMEI,6,1) ¬= ':' |
                SUBSTR(MPVREMEI,9,1) ¬= ':'
            THEN
            DO;
               MPVREMEL = -1;
               CALL NMAPA('WRONG START TIME!',2);
            END;

         END;
         OTHER
         DO;
         END;
      END;

   END VALID_DATA;

   MAKE_TEMP_STOR: PROC;
      DCL K_POINT_P2 PIC'99' INIT(Ø);
      DCL POC_DATE  CHAR(23) INIT(' ');

      POC_DATE = MPDATUMI || ',' || MPVREMEI;
      EXEC CICS DELETEQ TS QUEUE(TEMP) RESP(RES);
      IF RES¬=DFHRESP(NORMAL) THEN;

      ENTRY_NUM = Ø;
      STR_PROGLOG.TRANS_ID = MTRANI;
      EXEC CICS STARTBR FILE('PROGLOG') RIDFLD(PROGLOG_KEY_4)
        KEYLENGTH(4) GENERIC EQUAL RESP(RES);
```

```
IF RES=DFHRESP(NOTOPEN) THEN
DO;
   MPROGL = -1;
   CALL NMAPA('FILE PROGLOG IS CLOSED.',2);
END;
IF RES=DFHRESP(NOTFND) THEN
DO;
  DO I=1 TO 12;
    REDLOG2(I).REDLOGO=' ';
  END;
  MPROGL = -1;
  CALL NMAPA('LOG DOESN'T EXIST.',2);
END;
IF RES¬=DFHRESP(NORMAL) THEN
DO;
     MPROGL = -1;
     CALL NMAPA('ERROR WITH READING PROGLOG.',2);
END;
EXEC CICS READNEXT FILE('PROGLOG') INTO(PROGLOG_C2ØØ)
                                   RIDFLD(PROGLOG_KEY)
              KEYLENGTH(33) RESP(RES);
IF RES¬=DFHRESP(NORMAL)  THEN
DO;
    DO I=1 TO 12;
      REDLOG2(I).REDLOGO=' ';
    END;
    MPROGL = -1;
    CALL NMAPA('ERROR WITH READING PROGLOG.',2);
END;
DO WHILE(STR_PROGLOG.TRANS_ID=MTRANI & RES¬=DFHRESP(ENDFILE));
   DO I=1 TO 12;
        REDLOG2(I).REDLOGO=' ';
   END;
I=1;
L1:DO WHILE(STR_PROGLOG.TRANS_ID=MTRANI & I<=6);
    IF MTERMI = 'EMPTY' | MTERMI = STR_PROGLOG.TERMINAL THEN
    DO;
       IF MTUSERI = 'EMPTY' | MTUSERI = STR_PROGLOG.USER THEN
       DO;
         IF MRACUNI = Ø | MRACUNI = STR_PROGLOG.ACCOUNT THEN
         DO;
           IF STR_PROGLOG.DATUM_VREME > POC_DATE THEN
           DO;
             K_POINT_P2 = STR_PROGLOG.KONT_POINT;
             REDLOG2(I*2-1).REDLOGO='K:'||K_POINT_P2||'/TERM:'||
               STR_PROGLOG.TERMINAL||'/ACC:'||STR_PROGLOG.ACCOUNT||
               '/MSG:'||SUBSTR(STR_PROGLOG.MSG_TEST,1,45);
             REDLOG2(I*2).REDLOGO='    ' ||
                       SUBSTR(STR_PROGLOG.MSG_TEST,46,75);
             I=I+1;
```

```
                ENTRY_NUM = ENTRY_NUM + 1;
              END;
            END;
          END;
        END;
        EXEC CICS READNEXT FILE('PROGLOG') INTO(PROGLOG_C200)
                    RIDFLD(PROGLOG_KEY) KEYLENGTH(33) RESP(RES);
        IF RES=DFHRESP(ENDFILE) THEN LEAVE L1;
        IF RES¬=DFHRESP(NORMAL)  THEN
        DO;
             DO I=1 TO 12;
               REDLOG2(I).REDLOGO=' ';
             END;
             MPROGL = -1;
             CALL NMAPA('ERROR WITH READING PROGLOG.',2);
        END;
       END; /* L1:DO */

       IF ENTRY_NUM > 0 THEN
       DO;
           TOTAL_PAGES = TOTAL_PAGES + 1;
           EXEC CICS WRITEQ TS QUEUE(TEMP) FROM(TRA6321O);
       END;
    END; /* DO WHILE(STR_PROGLOG.TRANS_ID = MTRANI) */

    EXEC CICS ENDBR DATASET('PROGLOG');

    IF ENTRY_NUM > 0 THEN
    DO;
        PAGE=1;
        EXEC CICS READQ TS QUEUE(TEMP) INTO(TRA6321O) ITEM(PAGE);
        MPROGL = -1;
        PAGE_P3 = PAGE;
        TOTAL_PAGES_P3 = TOTAL_PAGES;
        CALL NMAPA('PAGE ' || PAGE_P3 || '/' || TOTAL_PAGES_P3,2);
    END;
    ELSE
    DO;
        MPROGL = -1;
        CALL NMAPA('LOG DOESN'T EXIST.',2);
    END;

  END MAKE_TEMP_STOR;

  PGMERR:
      MPROGL = -1;
      CALL NMAPA('THIS OPTION NOT WORKING AT THE MOMENT!',2);
END PRTEST;
```

PROGRAM2

```
PROXXXX: PROC(POINT) OPTIONS(MAIN);
/*********************************************************************/
/* DESCRIPTION: EXAMPLE OF CALLING WRI_TRA PROGRAM                   */
/* PROGRAM : PROGXXXX                                                */
/* MAP     : MAPXXXX                                                 */
/*********************************************************************/

 %INCLUDE DFHAID;
 %INCLUDE DFHBMSCA;

 DCL USER CHAR(8) INIT(' ');
 /******** DECLARATION OF COMMON AREA FOR WRI_TRA PROGRAM             */
 DCL 1 COMA_WRI_TRA,
       2 THIS_PROGRAM    CHAR(8)         INIT('PROGXXXX'),
       2 TESTING         DEC FIXED(1)    INIT(9),
       2 K_POINT_WRI_TRA PIC'999'        INIT(Ø),
       2 ACCOUNT_WRI_TRA PIC'9999999999' INIT(Ø),
       2 MSG_WRI_TRA     CHAR(137)       INIT(' ');
 /********       END OF DECLARATION
********************** */
 EXEC CICS HANDLE ABEND PROGRAM('PROGZZZZ');
 EXEC CICS ASSIGN USERID(USER);

 /* IN THE FIRST CALL WE ARE CHECKING IF THIS PROGRAM IS REALLY FOR
TESTING */

 CALL WRITE_TRACE(1,'BEFORE PROGYYYY/START');

 EXEC CICS LINK PROGRAM('PROGYYYY') COMMAREA(COMA_PROGYYYY);

 CALL WRITE_TRACE(2,'AFTER PROGYYYY/
'||'COMA_PROGYYYY.CODE='||COMA_PROGYYYY.CODE||
                     '/COMA_PROGYYYY.TEXT='||COMA_PROGYYYY.TEXT);

 WRITE_TRACE: PROC(TRACE_POINT,TRACE_MSG);
   DCL TRACE_POINT PIC'999';
   DCL TRACE_MSG CHAR(137);
   DCL S  BIN FIXED(31);


   IF COMA_WRI_TRA.TESTING = 1 | COMA_WRI_TRA.TESTING = 9
   THEN
   DO;
      ACCOUNT_WRI_TRA = ACCOUNT_NUMBER;
      K_POINT_WRI_TRA = TRACE_POINT;
      MSG_WRI_TRA     = TRACE_MSG;
      EXEC CICS LINK PROGRAM('WRI_TRA') COMMAREA(COMA_WRI_TRA) RESP(S);
      IF S¬=DFHRESP(NORMAL) THEN;
```

```
   END;
 END WRITE_TRACE;

 END PROXXXX;

WRI_TRA: PROC(POINT) OPTIONS(MAIN);
/*************************************************************/
/* DESCRIPTION: MAKING ENTRIES TO THE LOG PROGLOG            */
/* PROGRAM : WRI_TRA                                         */
/*************************************************************/

 %INCLUDE DFHAID;
 %INCLUDE RD_PROGLOG;  /* RECORD DESCRIPTION OF PROGLOG FILE */
 %INCLUDE RD_PROTEST;  /* RECORD DESCRIPTION OF PROTEST FILE */

 DCL (ADDR,DATETIME,STG,CSTG,SUBSTR)   BUILTIN;

 DCL POINT  PTR;
 DCL 1 COMA_WRI_TRA      BASED(POINT),
       2 THAT_PROGRAM    CHAR(8),
       2 TESTING         DEC FIXED(1),
       2 K_POINT         PIC'999',
       2 ACCOUNT_WRI_TRA PIC'9999999999',
       2 MSG             CHAR(137);

 DCL RESP_B31   BIN FIXED(31);
 DCL UTIME      DEC FIXED(15);
 DCL NAS_DATE   CHAR(1Ø);
 DCL USA_DATE   CHAR(1Ø);
 DCL N_TIME     CHAR(8);
 DCL N_USER     CHAR(8);

 RESP_B31 = Ø;
 UTIME     = Ø;
 NAS_DATE = ' ';
 USA_DATE = ' ';
 N_TIME     = ' ';
 N_USER  = ' ';
 EXEC CICS HANDLE ABEND PROGRAM('PROGZZZZ');
 EXEC CICS ASSIGN USERID(N_USER);

 IF COMA_WRI_TRA.TESTING = 9 THEN CALL TEST_PROGRAMA;
 IF COMA_WRI_TRA.TESTING = Ø THEN EXEC CICS RETURN;

 EXEC CICS ASKTIME ABSTIME(UTIME) RESP(RESP_B31);
 IF RESP_B31¬=DFHRESP(NORMAL) THEN;

 EXEC CICS FORMATTIME ABSTIME(UTIME)
           DATESEP('.') YYYYMMDD(USA_DATE)
```

```
          TIME(N_TIME) TIMESEP RESP(RESP_B31);
IF RESP_B31¬=DFHRESP(NORMAL) THEN;

STR_PROGLOG.TERMINAL  = EIBTRMID;
STR_PROGLOG.TRANS_ID= EIBTRNID;
STR_PROGLOG.DATUM_VREME = USA_DATE || ',' ||
                          SUBSTR(DATETIME,9,2)  || ':' ||
                          SUBSTR(DATETIME,11,2) || ':' ||
                          SUBSTR(DATETIME,13,2) || ':' ||
                          SUBSTR(DATETIME,15,3);
STR_PROGLOG.USER  = N_USER;
STR_PROGLOG.ACCOUNT = ACCOUNT_WRI_TRA;
STR_PROGLOG.KONT_POINT = K_POINT;

SELECT(EIBAID);
  WHEN(DFHENTER) STR_PROGLOG.PF_KEY = 'ENTR';
  WHEN(DFHCLEAR) STR_PROGLOG.PF_KEY = 'CLEA';
  WHEN(DFHPF1)   STR_PROGLOG.PF_KEY = 'PF1';
  WHEN(DFHPF2)   STR_PROGLOG.PF_KEY = 'PF2';
  WHEN(DFHPF3)   STR_PROGLOG.PF_KEY = 'PF3';
  WHEN(DFHPF4)   STR_PROGLOG.PF_KEY = 'PF4';
  WHEN(DFHPF5)   STR_PROGLOG.PF_KEY = 'PF5';
  WHEN(DFHPF6)   STR_PROGLOG.PF_KEY = 'PF6';
  WHEN(DFHPF7)   STR_PROGLOG.PF_KEY = 'PF7';
  WHEN(DFHPF8)   STR_PROGLOG.PF_KEY = 'PF8';
  WHEN(DFHPF9)   STR_PROGLOG.PF_KEY = 'PF9';
  WHEN(DFHPF10)  STR_PROGLOG.PF_KEY = 'PF10';
  WHEN(DFHPF11)  STR_PROGLOG.PF_KEY = 'PF11';
  WHEN(DFHPF12)  STR_PROGLOG.PF_KEY = 'PF12';
  WHEN(DFHPF13)  STR_PROGLOG.PF_KEY = 'PF13';
  WHEN(DFHPF14)  STR_PROGLOG.PF_KEY = 'PF14';
  WHEN(DFHPF15)  STR_PROGLOG.PF_KEY = 'PF15';
  WHEN(DFHPF16)  STR_PROGLOG.PF_KEY = 'PF16';
  WHEN(DFHPF17)  STR_PROGLOG.PF_KEY = 'PF17';
  WHEN(DFHPF18)  STR_PROGLOG.PF_KEY = 'PF18';
  WHEN(DFHPF19)  STR_PROGLOG.PF_KEY = 'PF19';
  WHEN(DFHPF20)  STR_PROGLOG.PF_KEY = 'PF20';
  WHEN(DFHPF21)  STR_PROGLOG.PF_KEY = 'PF21';
  WHEN(DFHPF22)  STR_PROGLOG.PF_KEY = 'PF22';
  WHEN(DFHPF23)  STR_PROGLOG.PF_KEY = 'PF23';
  WHEN(DFHPF24)  STR_PROGLOG.PF_KEY = 'PF24';
  OTHER          STR_PROGLOG.PF_KEY = 'EMPTY';
END;

STR_PROGLOG.MSG_TEST = MSG;
EXEC CICS WRITE DATASET('PROGLOG') FROM(PROGLOG_C200)
              RIDFLD(PROGLOG_KEY) RESP(RESP_B31);
IF RESP_B31¬=DFHRESP(NORMAL) THEN;
EXEC CICS RETURN;
```

```
 TEST_PROGRAMA: PROC;
   DCL RESP_TEST BIN FIXED(31) INIT(Ø);
   TEST_KEY = THAT_PROGRAM;
   EXEC CICS READ DATASET('PROTEST') INTO (TEST_C7Ø) RIDFLD(TEST_KEY)
                                      RESP(RESP_TEST);
   IF RESP_TEST=DFHRESP(NOTFND) /* IF WE HAVE NO ENTRY FOR THIS PROGRAM
*/
   THEN
   DO;
       COMA_WRI_TRA.TESTING = Ø;
   END;
   ELSE           /* IF WE HAVE ENTRY FOR THIS PROGRAM */
   DO;
      IF RESP_TEST=DFHRESP(NORMAL) /* READ IS ALL RIGHT */
      THEN
      DO;
         COMA_WRI_TRA.TESTING = PRO_TEST;
         IF PRO_TEST = 1 THEN
         DO;
            IF TESTNI_TERMINAL ¬= 'EMPTY' & EIBTRMID ¬= TESTNI_TERMINAL
            THEN COMA_WRI_TRA.TESTING = Ø;
            IF TESTNI_USER ¬= 'EMPTY' & N_USER ¬= TESTNI_USER
            THEN COMA_WRI_TRA.TESTING = Ø;
            IF TESTNI_ACCOUNT ¬= Ø & ACCOUNT_WRI_TRA ¬= TESTNI_ACCOUNT
            THEN COMA_WRI_TRA.TESTING = Ø;
         END;
      END;
      ELSE COMA_WRI_TRA.TESTING = Ø;
                  /* IF SOMETHING IS WRONG WE DON'T WRITE THE LOG */
   END;
 END TEST_PROGRAMA;

 END WRI_TRA;
```

Nebojsa Cosic
Project Developer
Postal Savings Bank of Yugoslavia (Yugoslavia)

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. If you have an idea for an article contact the editor, Trevor Eddolls, at any of the addresses shown on page 2.

CICS and faxing

IBM's CICS product and OMTOOL's Fax Sr product work together perfectly to allow online transactions to send data to fax machines. In today's electronic age, one would think that fax machines are a thing of the past, but not so. Many businesses rely on fax machines to send and receive data. The fax process is very complicated, for the data is converted into its own format. After reading several books on converting text data to something that a fax machine would understand I was glad when I discovered that OMTOOL had a product that would take a text document and convert it to fax. I was also relieved to know that I didn't have to code to the communications protocol of a fax machine, especially since most programming in CICS does not require this level of communication protocol knowledge. The Fax Sr product that I used runs on an NT server and is basically installed out of the box. CICS is also standard and installed out of the box. The versions of CICS used were 4.1 and TX1.3.

Fax Sr has an SMTP (Simple Mail Transport Protocol) component, which is an e-mail component which allows you to e-mail messages to Fax Sr, and Fax Sr will read the message and fax it. In simpler terms, you send an e-mail message with the fax number as the recipient to Fax Sr, and the product will fax the text of the message to the destined fax machine. Fax Sr also has a component that will e-mail the results of the fax to an e-mail address. This e-mail message can be customized, but basically it states whether the fax transmission was successful or not. If not, it gives the reason for the failure. This message also contains the number of pages faxed.

CICS has supported TCP/IP for years. You can use this interface to communicate with the SMTP gateway of Fax Sr. You can also use this same interface to access an e-mail mailbox to process the messages coming from Fax Sr.

In my shop, applications write their messages to a database. The format of the message is: the first line contains the fax number, and the remaining records are the contents of the fax message. For simplicity's sake, the messages are limited to being 80 columns wide. Each

message has a flag that states the beginning and end of the message. This was needed, since all the records are contained under a single root. In reality, we have our own in-house written message queuing system. This predates IBM's MQ product. Without getting into the architecture of this system, I will leave the above explanation as it is. The bottom line is that the applications have a central repository where they can send their fax messages and not be bothered with the 'how tos' of sending a fax.

A CICS COBOL II program reads these messages and creates an e-mail message and sends this information to Fax Sr. A record of this interaction with Fax Sr is maintained to ensure that the message was sent.

Another CICS COBOL II program reads an e-mail mailbox, correlates the e-mail messages with the list of messages that have been sent to Fax Sr, and upon completion removes the fax from the list of 'in progress' faxes. This process runs automatically without human intervention. I have another CICS transaction that starts every five minutes to launch the above transactions.

Let's first discuss the sending of the message to Fax Sr. Although this topic deals with sending data to Fax Sr, it could be any e-mail type address – the coding is the same, the protocol is the same. It's just that, in this case, I am sending e-mail messages to Fax Sr; it could well have been to any e-mail address in the world.

Let's take a look at the program that sends the e-mail messages to Fax Sr. I have removed a lot of the code that deals with the business and the actual process of reading of the data. I have concentrated on the actual code needed to open a socket with TCP/IP and the related calls required to send the e-mail message and then close the socket. Also, the mainframe deals with the EBCDIC character set whereas the distributed platform deals in the ASCII character set. Any data that is sent to an SMTP client must be converted from EBCDIC to ASCII, and *vice versa* – any data coming from an SMTP client must be converted from ASCII to EBCDIC.

The conversion programs are provided for you with CICS. They are EZACIC05 to convert from ASCII to EBCDIC and EZACIC04 to

convert from EBCDIC to ASCII.

To convert the data to ASCII before sending it out:

```
CALL 'EZACIC04' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                            TCPLENG TCP-BUF ERRNO RETCODE.
```

To convert the data to EBCDIC after reading the data in:

```
CALL 'EZASOKET' USING SOKET-READ SOCKID
                           TCPLENG TCP-BUF-READ ERRNO RETCODE.
      CALL 'EZACIC05' USING TCP-BUF-READ TCPLENG.
```

Another piece of useful information is how to convert an IP address into decimal. You will need to do this for your SMTP mail gateway. Say your mail domain is abc.fax.com; you can *ping* this name to obtain the IP address. For the sake of this discussion, let's say the IP address is 100.200.300.400. You must convert this address to decimal so your program can use it to connect to the mail server, as with the variable HOSTADDR-VALUE used later in the program:

```
            TO CALCULATE THE VALUE FOR HOSTADDR-VALUE
CONVERT THE 100 TO HEX -> 64
CONVERT THE 200 TO HEX -> C8
CONVERT THE 300 TO HEX -> 12C
CONVERT THE 400 TO HEX -> 190
CONCATENATE THEM ALL TOGETHER
64C812C190
CONVERT THIS TO DECIMAL
1690833600
01  HOSTADDR-VALUE     PIC 9(10) BINARY VALUE 1690833600.
```

You will also need to know the name of the TCP/IP address space running on your system. In our case, it will be TCPATCP. This name will be used when we make our connection.

All interactions with TCP/IP are via the EZASOKET calls. The first call is to initialize the TCP/IP environment:

```
CALL 'EZASOKET' USING SOKET-INITAPI  MAXSOC   IDENT SUBTASK   MAXSNO
ERRNO   RETCODE.

     IF RETCODE <  0 THEN
        GO TO PGM-EXIT.
```

```
Ø1  SOKET-INITAPI               PIC X(16) VALUE 'INITAPI         '.
Ø1  MAXSOC                      PIC 9(4) BINARY VALUE 5Ø.
Ø1  IDENT.
    1Ø  TCPNAME                 PIC X(8) VALUE SPACES.
    1Ø  ADSNAME                 PIC X(8) VALUE SPACES.
Ø1  SUBTASK.
    1Ø SUBTASK-ALPHA            PIC X(1) VALUE SPACES.
    1Ø SUBTASK-NUMBER           PIC 9(7) VALUE Ø.
Ø1  MAXSNO                      PIC 9(8) BINARY VALUE Ø.
Ø1  ERRNO                       PIC 9(8) BINARY VALUE Ø.
Ø1  RETCODE                     PIC S9(8) BINARY VALUE +Ø.
```

ADSNAME is the applid of your CICS, ie:

```
EXEC CICS ASSIGN APPLID(ADSNAME) END-EXEC.
```

TCPNAME is the jobname of the TCP/IP address space running on your system.

SUBTASK has to be unique, eg:

```
MOVE EIBTASKN TO SUBTASK-NUMBER.
MOVE EIBTRNID(3:1) TO SUBTASK-ALPHA.
```

RETCODE will contain a '–1' if the call fails.

The next call is:

```
CALL 'EZASOKET' USING SOKET-SOCKET    AF   SOCTYPE
                      PROTO ERRNO RETCODE.
IF RETCODE <  Ø THEN
         GO TO PGM-EXIT
     ELSE
         MOVE RETCODE TO  SOCKID.

Ø1 SOKET-SOCKET                 PIC X(16) VALUE 'SOCKET          '.
Ø1  AF                          PIC 9(8) BINARY VALUE 2.
Ø1  SOCTYPE                     PIC 9(8) BINARY VALUE 1.
Ø1  PROTO                       PIC 9(8) BINARY VALUE Ø.
```

The SOCKID is our unique connection with TCP/IP. This is one of the ways that TCP/IP can keep track of all the applications it is communicating with.

Our next call is to make the connection to the e-mail client, in our case it is the Fax Sr SMTP client:

```
  MOVE 2 TO FAMILY.
  MOVE 25 TO PORT.
```

```
  COMPUTE IP-ADDRESS = HOSTADDR-VALUE.
  CALL 'EZASOKET' USING SOKET-CONNECT SOCKID
                        S-NAME ERRNO RETCODE.
  IF RETCODE <  Ø THEN
     GO TO PGM-EXIT.
EXEC CICS DELAY INTERVAL(ØØØØØ1) END-EXEC.
MOVE SPACES TO TCP-BUF.
MOVE 8ØØØ TO TCPLENG.
CALL 'EZASOKET' USING SOKET-READ   SOCKID
                          TCPLENG   TCP-BUF   ERRNO   RETCODE.

    IF RETCODE <  Ø THEN
        GO TO PGM-EXIT.
```

IP-ADDRESS is the recipient of the conversion we did earlier of the IP address to a decimal number.

25 is the default SMTP port. This is true of any e-mail server that I have seen.

2 is always the family:

```
Ø1 SOKET-CONNECT                  PIC X(16) VALUE 'CONNECT         '.
Ø1 SOKET-READ                     PIC X(16) VALUE 'READ            '.
Ø1  S-NAME.
     Ø3  FAMILY                   PIC 9(4) BINARY VALUE 2.
     Ø3  PORT                     PIC 9(4) BINARY VALUE 25.
     Ø3  IP-ADDRESS               PIC 9(8) BINARY VALUE Ø.
     Ø3  RESERVED                 PIC X(8).
Ø1  TCP-BUF                       PIC X(8ØØØ) VALUE IS SPACES.
```

This is our first call that we will get a response to back from the e-mail client. We normally do not care what the response is, we just have to read it to follow the proper TCP/IP protocols. If you are interested in seeing what is returned, you can print out the responses to **tempstg** and view them using CEBR.

The message is normally customized by the keeper of the e-mail client so the length can vary.

I found that if you use a large number, like 8000, then you will always get the entire message back. It is normally less than 20 characters. It will probably say something like “Welcome to abc.com”.

The delay is needed to allow enough time for the connection request to reach the e-mail client and for it to respond.

At this stage of the program, we have connected to the Fax Sr SMTP client. We are now ready to start the SMTP protocol process of sending the e-mail headers. After this stage, we will start the actual sending of the message:

```
MOVE HELO-MSG TO TCP-BUF.
MOVE HELO-LENGTH TO TCPLENG.
CALL 'EZACIC04' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE   SOCKID
                      TCPLENG   TCP-BUF   ERRNO   RETCODE.

IF RETCODE <  0 THEN
    GO TO PGM-EXIT.
EXEC CICS DELAY INTERVAL(000001) END-EXEC.
MOVE SPACES TO TCP-BUF.
MOVE 8000 TO TCPLENG.
CALL 'EZASOKET' USING SOKET-READ SOCKID
                          TCPLENG TCP-BUF    ERRNO   RETCODE.

    IF RETCODE <  0 THEN
        GO TO PGM-EXIT.

01 SOKET-WRITE                       PIC X(16) VALUE 'WRITE           '.
01  HELO-MSG.
    05  FILLER                 PIC X(22)  VALUE 'HELO TechnoFax.abc.com'.
    05  FILLER                       PIC X(2) VALUE '  '.
01  HELO-LENGTH                      PIC S9(4) VALUE +24.
```

All messages end with an X'0D15'. This is what is in the filler PIC X(2) field.

HELO is a reserved word in the SMTP protocol. The remainder of the filler is whatever you want to put.

Again, you must issue a read to get the response from the client.

The next commands will inform Fax Sr who the message is from:

```
MOVE MAIL-FROM-MSG TO TCP-BUF.
MOVE MAIL-FROM-LENGTH TO TCPLENG.

CALL 'EZACIC04' USING   TCP-BUF    TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE    SOCKID
                      TCPLENG   TCP-BUF    ERRNO    RETCODE.

IF RETCODE <  0 THEN
   GO TO PGM-EXIT.

EXEC CICS DELAY INTERVAL(000001) END-EXEC.
```

```
     MOVE SPACES TO TCP-BUF.
     MOVE 8000 TO TCPLENG.
     CALL 'EZASOKET' USING SOKET-READ SOCKID
                           TCPLENG TCP-BUF    ERRNO   RETCODE.

      IF RETCODE <  0 THEN
          GO TO PGM-EXIT.
01  MAIL-FROM-MSG.
    05  FILLER                              PIC X(11) VALUE 'MAIL FROM:<'.
    05  FILLER                              PIC X(9) VALUE 'TechnoFax'.
    05  FILLER                              PIC X(9) VALUE '@abc.com>'.
    05  FILLER                              PIC X(2) VALUE '  '.
 01  MAIL-FROM-LENGTH                       PIC S9(4) VALUE +31.
```

As with all messages, they must end with a X'0D15'.

Again, this can be anything you want. In my case it is the name of an e-mail mailbox, for this is the e-mail address that Fax Sr will send the responses to if so desired.

The next step will tell Fax Sr the phone number to fax the message to. In our case it is a phone number. It could just as well have been a real e-mail address:

```
MOVE 'Rcpt to:<' TO TCP-BUF.
MOVE PHONE-NUMBER  TO TCP-BUF(10:22).
PERFORM VARYING I FROM 60 BY -1 UNTIL
  TCP-BUF(I:1) NOT = ' ' OR I < 20 END-PERFORM.
ADD 1 TO I.
MOVE '@fax.abc.com' TO TCP-BUF(I:12).
ADD 12 TO I.
MOVE '>' TO TCP-BUF(I:1).
ADD 1  TO I.
MOVE HLD-CRLF TO TCP-BUF(I:2).
ADD 1 TO I.
MOVE I  TO TCPLENG.
CALL 'EZACIC04' USING TCP-BUF   TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE   SOCKID
                      TCPLENG TCP-BUF   ERRNO RETCODE.

IF RETCODE <  0 THEN
   GO TO PGM-EXIT.
EXEC CICS DELAY INTERVAL(000001) END-EXEC.

MOVE SPACES TO TCP-BUF.
MOVE 8000 TO TCPLENG.
CALL 'EZASOKET' USING SOKET-READ SOCKID
                           TCPLENG TCP-BUF    ERRNO   RETCODE.
```

```
IF RETCODE <  Ø THEN
    GO TO PGM-EXIT.
Ø1  HLD-CRLF            PIC X(Ø2) VALUE '  '.    $ The value is x'ØD15'
```

The ‘Receipt to:’ is the destination of the fax (e-mail) message. It is built in this way to allow the phone number to be variable in size. Some could have 10 digits, others could have 7 digits, and others could have even more, based on where in the world the fax machine is located. In this example, the largest the phone number can be is 22 characters.

This concludes the headers that we have to send to Fax Sr or to an e-mail client.

The next communication with the client is to tell it that we are through with this portion of the headers. This is done by sending the text ‘DATA’:

```
MOVE DATA-MSG TO TCP-BUF.
MOVE DATA-LENGTH TO TCPLENG.

CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                      TCPLENG TCP-BUF ERRNO RETCODE.

 IF RETCODE <  Ø THEN
    GO TO PGM-EXIT.
 EXEC CICS DELAY INTERVAL(ØØØØØ1) END-EXEC.

 MOVE SPACES TO TCP-BUF.
 MOVE 8ØØØ TO TCPLENG.
 CALL 'EZASOKET' USING SOKET-READ SOCKID
                            TCPLENG TCP-BUF    ERRNO   RETCODE.

 IF RETCODE <  Ø THEN
     GO TO PGM-EXIT.
Ø1  DATA-MSG.
    Ø5  FILLER                  PIC X(4)  VALUE 'DATA'.
    Ø5  FILLER                  PIC X(2) VALUE '  '.        $  x'ØD15'
Ø1  DATA-LENGTH                 PIC S9(4) VALUE +6.
```

The next section may seem redundant, but it is just part of the SMTP protocol. Earlier we sent the ‘From’ and ‘Receipt to’. We are going to send the ‘To’, the ‘From’, and the ‘Subject’. Although the ‘To’ and ‘From’ are the same as before, we need to send them again because they are used differently by the e-mail client. This also concludes the

header portion of the e-mail. The sending of the CRLF-MSG tells the e-mail client that all the header information has been sent:

```
MOVE SPACES TO TCP-BUF.
MOVE 'To: ' TO TCP-BUF.
MOVE PHONE-NUMBER(1:22) TO TCP-BUF(5:22).
PERFORM VARYING I FROM 5Ø BY -1 UNTIL
       TCP-BUF(I:1) NOT = ' ' OR I < 15 END-PERFORM.
ADD 1 TO I.
MOVE '@fax.abc.com' TO TCP-BUF(I:12).
ADD 12 TO I.
MOVE HLD-CRLF TO TCP-BUF(I:2).
ADD 1 TO I.
MOVE I  TO TCPLENG.
CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
TCPLENG TCP-BUF   ERRNO RETCODE.

 IF RETCODE <  Ø THEN
    GO TO PGM-EXIT.
MOVE FROM-MSG TO TCP-BUF.
MOVE FROM-LENGTH TO TCPLENG.
CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                          TCPLENG TCP-BUF   ERRNO   RETCODE.
MOVE 'Subject: ' to TCP-BUF(1:9).
MOVE PHONE-NUMBER TO TCP-BUF(1Ø:22).
MOVE HLD-CRLF TO TCP-BUF(23:2).
CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                          TCPLENG TCP-BUF ERRNO RETCODE.

IF RETCODE <  Ø THEN
     GO TO PGM-EXIT.

MOVE DATE-FAX TO TCP-BUF.
MOVE DATE-LENGTH TO TCPLENG.
CALL 'EZACICØ4' USING TCP-BUF  TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                          TCPLENG  TCP-BUF  ERRNO  RETCODE.

IF RETCODE <  Ø THEN
GO TO PGM-EXIT.
MOVE CRLF-MSG TO TCP-BUF.
MOVE CRLF-LENGTH TO TCPLENG.
CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                          TCPLENG TCP-BUF ERRNO RETCODE.
```

```
IF RETCODE <  Ø THEN
    GO TO PGM-EXIT.

Ø1  CRLF-MSG.
    Ø5  FILLER                    PIC X(2) VALUE '  '.     $ x'Ød15'
Ø1  CRLF-LENGTH                   PIC S9(4) VALUE +2.
```

You are now ready to start sending the actual message text. The program loops on this section until all data has been sent. You want to insert the CRLF (X'0D15') at the end of each line. You want to convert it to ASCII, then write it:

```
MOVE HLD-CRLF TO TCP-BUF(J:2).
COMPUTE J = J + 1.
MOVE J TO TCPLENG.

CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                          TCPLENG TCP-BUF ERRNO RETCODE.
IF RETCODE <  Ø THEN
   GO TO PGM-EXIT.
```

After you have completed sending all your data, we need to close down the socket.

A couple of calls are made to do this. The DOT-MSG is simply the string X'0D154B0D15'. This signals the end of data and the remaining calls are protocol lines:

```
MOVE DOT-MSG TO TCP-BUF.
MOVE DOT-LENGTH TO TCPLENG.
CALL 'EZACICØ4' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                       TCPLENG TCP-BUF ERRNO RETCODE.

 IF RETCODE <  Ø THEN
    GO TO PGM-EXIT.

 EXEC CICS DELAY INTERVAL(ØØØØØ1) END-EXEC.

 MOVE SPACES TO TCP-BUF.
 MOVE 8ØØØ TO TCPLENG.
 CALL 'EZASOKET' USING SOKET-READ SOCKID
                            TCPLENG TCP-BUF    ERRNO   RETCODE.

 IF RETCODE <  Ø THEN
     GO TO PGM-EXIT.
Ø1  DOT-MSG.
```

```
    05  FILLER                 PIC X(2) VALUE '  '.      $ x'0D15'
    05  FILLER                 PIC X(1)       VALUE '.'.      $ x'4B'
    05  FILLER                 PIC X(2) VALUE '  '.      $ x'0D15'
01  DOT-LENGTH                 PIC S9(4) VALUE +7.
```

Next is the shutdown sequence. The return code is not checked after the last two commands. You could, and print an error flag if necessary, but we are in shutdown mode, so it has no effect:

```
MOVE QUIT-MSG TO TCP-BUF.
MOVE QUIT-LENGTH TO TCPLENG.
CALL 'EZACIC04' USING TCP-BUF TCPLENG.
CALL 'EZASOKET' USING SOKET-WRITE SOCKID
                      TCPLENG TCP-BUF ERRNO RETCODE.

 IF RETCODE <  0 THEN
      GO TO PGM-EXIT.
 EXEC CICS DELAY INTERVAL(000001) END-EXEC.

 MOVE SPACES TO TCP-BUF.
 MOVE 8000 TO TCPLENG.
 CALL 'EZASOKET' USING SOKET-READ SOCKID
                           TCPLENG TCP-BUF    ERRNO   RETCODE.

 IF RETCODE <  0 THEN
     GO TO PGM-EXIT
CALL 'EZASOKET' USING SOKET-SHUTDOWN SOCKID HOW
           ERRNO RETCODE.
CALL 'EZASOKET' USING SOKET-CLOSE SOCKID
           ERRNO RETCODE.

01  QUIT-MSG.
    05  FILLER                  PIC X(4)  VALUE 'QUIT'.
    05  FILLER                  PIC X(2) VALUE '  '.     $ x'0D15'
01  QUIT-LENGTH                 PIC S9(4) VALUE +6.
01  HOW                         PIC 9(8) BINARY VALUE 2.
```

This concludes our example of sending a message to OMTOOLS FAX Sr product. Again, this could be any e-mail client. For more information about the SMTP protocol and other options not listed above refer to RFC 821.

In a future article, I will discuss how to read an e-mail mailbox from CICS.

David Mishoe
IT Architect
CSX Technology (USA)

CICS/TS 1.3 Web documents in memory

This article reviews recent enhancements to the CICS Web Support (CWS) FORMFIELD API and demonstrates an easy mechanism for developers to generate Web content 'off-line' using their favourite workstation tools for delivery by CICS from memory.

WEB CONTENT DEVELOPMENT

Let's face it, TSO option 2 is not the best editing suite for Web developers. Tools such as IBM's Websphere studio, Macromedia's Dreamweaver/Flash products, Microsoft's Frontpage or even Office products offer a much friendlier environment in which to write HTML Web content.

The problem, of course, is that we then have to upload these pages to CICS using transfer mechanisms like FTP or ind$file, which normally involves storing the content in a partitioned dataset.

The PDS member is then assigned a DOCTEMPLATE definition and new-copied. Next, an application program containing the CWS DOCUMENT CREATE and INSERT commands can deliver the content to the Web browser.

PERFORMANCE

The problem with this approach however is that CICS, using the DOCTEMPLATE mechanism, does not cache the Web page in memory.

Every time the CICS application runs it must perform physical I/O on the PDS. Contrast this with traditional BMS, where a popular map can be made resident. This DOCTEMPLATE PDS approach is obviously going to incur significant overheads.

WEB SERVERS VERSUS GATEWAYS VERSUS CICS/TS 1.3

Competing technologies often claim 'superior cacheing facilities' on MVS as their *raison d'etre*. It is often suggested that a three-tier

environment employing various gateways offers a more sensible approach to Web-enabling legacy applications while providing better performance than the native CWS approach. Refer to SG24-5748 Chapter 8 for an unbiased comparison of native CWS access versus various competing technologies (in summary, even with the DOCTEMPLATE PDS approach, CWS beats the competition).

But what if CICS went further and actually stored the Web pages in memory, what if CICS utilized dataspaces or a coupling facility in a sysplex environment to cache its pages?

DOCUMENT API

The Document API has a number of options available on the CREATE and INSERT commands to enhance CICS Web application performance.

The TEMPLATE option, if used, implies DOCTEMPLATES and I/O overhead.

The FROM option or BINARY option lets us load the data from any CICS media, and the DOCTEMPLATE definition has just become obsolete.

All we need to do is get the Web page (developed on the workstation) into CICS and then store the data in some very fast media, like a DB2 database or a CICS/VSAM datatable.

FORMFIELD API APAR PQ35709

As reviewed in the October 2000 edition of *CICS Update*, Issue 179, the EXEC CICS WEB READ FORMFIELD command has had a SET option added to it.

This option is primarily intended to let CICS support file uploads from multipart/form data.

ASCII/EBCDIC CONSIDERATIONS

CLNTCODEPAGE() and HOSTCODEPAGE() have no effect when using the SET option.

SET when used will point to a buffer containing the uploaded file.

If the uploaded data requires conversion from ASCII to EBCDIC, the IBM recommendation is to use the C/C++ ICONV utility.

Using this new facility gives CICS developers a very easy way to upload and store any workstation file in CICS media.

Consequently this stored page can then be used by a CWS Web application for delivery to a Web browser.

PROCESSING OVERVIEW

A variation of the T13TXT program presented in Issue 179 of *CICS Update* displays the FILEUPLD HTML page – see Figure 1.

The HTML includes the following code:

```
<form name="FILEUPLD"
 action="/CICS/CWBA/FILEUPLD"
 enctype="multipart/form-data"
 method="post">
```

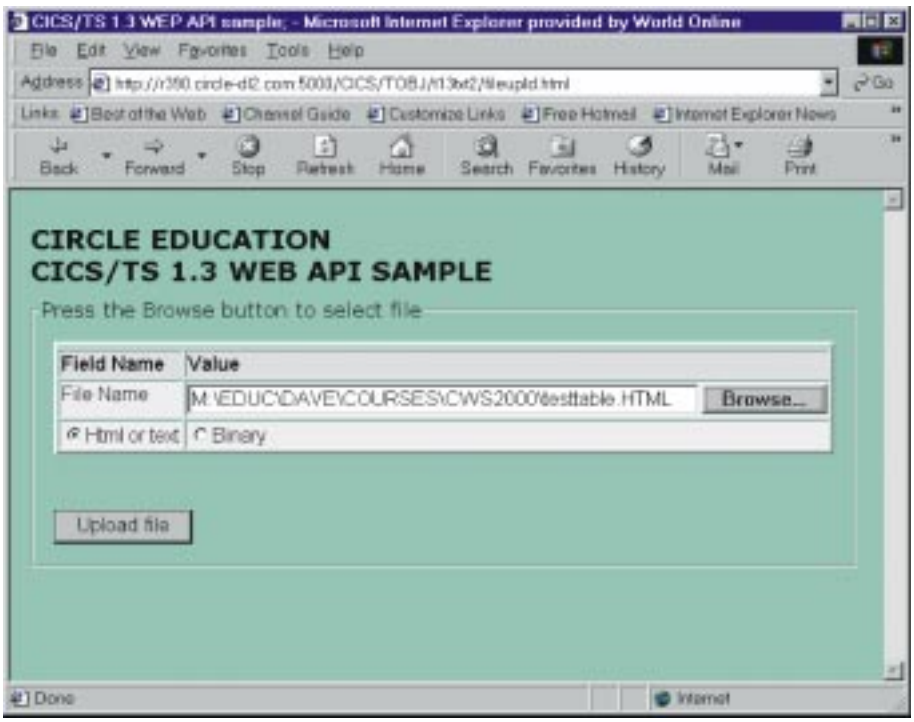

Figure 1: Web API sample

Using the *browse* button you select which file you wish to upload from your local or network drive. When the *Upload file* button is pressed, the FILEUPLD program processes the uploaded file.

Notice that this program, having received the input, simply turns the data around and sends the file back to the Web browser. (NB: as supplied, this sample program works correctly only with HTML files transmitted using this mechanism. You would have to augment the code with appropriate MIME-type processing to get binary files sent back to the browser correctly.)

However, regardless of whether you transmit HTML or binary data, the supplied code stores the uploaded file in some CICS media. This program simply writes to a TSQ, though a more permanent store like VSAM or DB2 would be preferred.

Finally the program uses the DOCUMENT CREATE command using the FROM option instead of the TEMPLATE option to improve performance for text/html files. The comments in the code include a example of the BINARY option that would be used for other types of data (GIF, JPEG, etc).

Using such a technique, you have just achieved three things:

1 File uploads via a browser from a PC to store a document in any CICS media.

2 Improved performance by eliminating physical I/O when accessing the pages (if for instance you have chosen to store the uploaded file in a user-maintained data table).

3 Eliminated the requirement for doctemplates in your system altogether, and as a by-product, the pain of having to do new-copies when the content changes. Here you simply rewrite the VSAM or DB2 record, and the next time you access it (from memory) you pick up the change.

FILEUPLD HTML

```
<!doctype html public "-//IEFT//DTD HTML 4.0//EN">
<html>
<!--#set var=protocol value=http://-->
```

```
<BASE href="&protocol;&hostv;">
<head>
<title> CICS/TS 1.3 WEP API sample;</title>
<META NAME="AUTHOR" CONTENT="Circle">
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=iso-8859-1">
<STYLE>
BODY {font-size: 10pt; font-family: Arial;}
TABLE, INPUT, SELECT {font-size: 10pt; font-family: Arial;}
TH, TD {vertical-align: top; text-align: left;}
</STYLE>
<!-- SAMPLE_STYLE_START -->
<LINK REL="stylesheet" HREF="/cics/tobj/t13txt/bascss" TYPE="text/css">
<!-- SAMPLE_STYLE_END -->
</head>
<body  bgcolor="#90c4b0" TOPMARGIN=0 LEFTMARGIN=0 BGPROPERTIES="FIXED"
link="#0000FF" vlink="#800080" alink="#FF0000">
<BLOCKQUOTE CLASS="body">
<form name="FILEUPLD"
 action="/CICS/CWBA/FILEUPLD"
 enctype="multipart/form-data"
 method="post">
<h2>CIRCLE EDUCATION<br>CICS/TS 1.3 WEB API SAMPLE </H2>
<p>
<FIELDSET STYLE="width: 520;height: 100">
<LEGEND>Press the Browse button to select file</legend>
<BLOCKQUOTE CLASS="body">
<table bgcolor="#B7D9CB" border="2" width="500">
<tr>
<TR><TH colspan=2 nowrap>Field Name</TH><TH colspan=2 nowrap>Value</
TH></TR>
<td colspan=2 nowrap> File Name   </td>
<TD colspan=2 nowrap>
<input type="file" name="input_data" size="50" maxlength="255" value=""
                                  title="enter the file name">
</TD>
</tr>
<tr>
<td colspan=2 nowrap>
<input type="radio" name="type" value="txt" checked>Html or text  </td>
<td><input type="radio" name="type" value="bin">Binary</td>
</tr>
</table>
<br><br>
<p>
<input type="submit" name="submit" TITLE="press to upload" ACCESSKEY="U"
value="Upload file"
</BLOCKQUOTE>
</FIELDSET>
</form>
</BODY>
</HTML>
```

FILEUPLD SOURCE

```
      * This is CWSP class version used to upload html etc
      * CIRCLE COMPUTER GROUP
       IDENTIFICATION DIVISION.
       PROGRAM-ID. FILEUPLD.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
      *        Symbols table
       Ø1 WS-SYMBOLS.
          Ø3 filler             pic x(6)  value 'deptv='.
          Ø3 WS-SYM-DEPT        PIC X VALUE SPACE.
          Ø3 filler             pic x(6)  value '#empv='.
          Ø3 WS-SYM-EMP         PIC X(5) VALUE SPACES.
          Ø3 filler             pic x(7)  value '#namev='.
          Ø3 WS-SYM-NAME        PIC X(2Ø) VALUE SPACES.
          Ø3 filler             pic x(7)  value '#adr1v='.
          Ø3 WS-SYM-ADR1        PIC X(2Ø) VALUE SPACES.
          Ø3 filler             pic x(7)  value '#adr2v='.
          Ø3 WS-SYM-ADR2        PIC X(2Ø) VALUE SPACES.
          Ø3 filler             pic x(7)  value '#adr3v='.
          Ø3 WS-SYM-ADR3        PIC X(2Ø) VALUE SPACES.
          Ø3 FILLER             PIC X(7) VALUE '#cstat='.
          Ø3 WS-STATE           PIC X(4) VALUE LOW-VALUES.
          Ø3 filler             pic x(7) value '#hostv='.
          Ø3 ws-host            pic x(24) value space.
          Ø3 FILLER             PIC X(9) VALUE '#message='.
          Ø3 WS-BAD-MESSAGE     PIC X(2Ø) VALUE SPACES.
          Ø3 FILLER             PIC X(Ø6) VALUE '#path='.
          Ø3 WS-path            PIC X(18) VALUE SPACES.
       Ø1 WS-HTTP-HEADER.
          Ø5 pragma-hdr      PIC X(6) VALUE IS 'Pragma'.
          Ø5 pragma-val      PIC X(7) VALUE IS 'nocache'.
       77  TEST-DATA-SIZE   PIC S9(4) BINARY.
       77  XLATE-DIRECTION  PIC S9(4) BINARY.
           88  ASCII-EBCDIC VALUE Ø.
           88  EBCDIC-ASCII VALUE 1.
       77  A2E-RC           PIC S9(4) BINARY.
       Ø1 WS-SUB             PIC S9(8) COMP VALUE +Ø.
       Ø1 WS-buffer-len      PIC S9(8) COMP VALUE +Ø.
       Ø1 INIT-VAL PIC X VALUE LOW-VALUES.
       Ø1 WS-RESP             PIC S9(8) BINARY.
       Ø1 hdr-name PIC x(1ØØ) value spaces.
       Ø1 hdr-val  PIC x(8ØØ) value spaces.
       Ø1 WS-DELIM PIC x value '#'.
       Ø1 WS-TOKEN       PIC X(16).
       Ø1 WS-TOKEN1      PIC X(16).
       Ø1 WS-RETRIEVE-LENGTH  PIC S9(8) BINARY value +Ø.
       Ø1 WS-file-name   pic x(1Ø) value 'input_data'.
       Ø1 WS-file-address     pointer.
       Ø1 WS-file-length      PIC S9(8) BINARY value 32764.
       Ø1 CRLF              PIC X(2)  VALUE IS X'ØD25'.
```

```
 Ø1 CRNL              PIC X(2)  VALUE IS X'ØD15'.
 Ø1 ws-type           pic x(3).
 77 text-file         pic x(3) VALUE 'txt'.
 77 binary-file       pic x(3) VALUE 'bin'.
 LINKAGE SECTION.
  Ø1 LS-BUFFER.
    Ø5 FILLER              PIC X(32764).
 procedure division.
 aa-main section.
 aa-code.
*  browse through httpheaders loking for host                    *
     exec cics web startbrowse httpheader
               resp(ws-resp)
     end-exec
     perform until ws-resp not equal dfhresp(normal)
     exec cics web readnext
          httpheader (hdr-name)
          namelength(length of hdr-name)
          value(hdr-val)
          valuelength(length of hdr-val)
          resp(ws-resp)
     end-exec
     if hdr-name = "Host"
        move hdr-val to ws-host
        end-if
     end-perform
     exec cics web endbrowse httpheader
          resp(ws-resp)
     end-exec
*  find formfield "type"
     move "type" to hdr-name
     exec cics web read
          formfield(hdr-name)
          namelength(4)
          value(hdr-val)
          valuelength(length of hdr-val)
          resp(ws-resp)
     end-exec
     move hdr-val(1:3) to ws-type
*  use formfield read set to get uploaded file
*  this program imposes 32764 but could be larger
     exec cics web read
          formfield(ws-file-name)
          namelength(length of ws-file-name)
          valuelength(ws-file-length)
          set(address of ls-buffer)
          resp(ws-resp)
     end-exec
*  test if text or binary was selected
*  if text call ISCXLATE to run ascii - ebcdic conversion
*  if binary do accept asis
     evaluate ws-type
```

```
     when text-file
         set ascii-ebcdic to true
         call 'ISCXLATE' using by reference ls-buffer
              by value ws-file-length xlate-direction
              returning a2e-rc
*  ICONV is returning CRNL instead of CRLF so replace
         inspect ls-buffer  replacing all
         crnl by crlf
*  at this point for text files we have the uploaded data
*  so you could save it in any CICS media - VSAM DB2 etc
*  this program simply sends it back out
*  create document using FROM option
         exec cics document create
                   doctoken(ws-token)
                   docsize(ws-retrieve-length)
                   from(ls-buffer)
                   length(ws-file-length)
                   resp(ws-resp)
         end-exec
*  send with EBCDIC - ASCII conversion
         exec cics web send
                   doctoken(ws-token)
                   clntcodepage('819')
         end-exec
*    when binary-file
*  at this point for binary files we have the uploaded data
*  so you could save it in any CICS media - VSAM DB2 etc
*  the sample code demonstrates the BINARY option of
*  document create, however the coding is not complete
*  because you would have to work out the correct mime type
*  before it would be correctly rendered on a browser

*  create document using BINARY option

*         exec cics document create
*                   doctoken(ws-token)
*                   docsize(ws-retrieve-length)
*                   binary(ls-buffer)
*                   length(ws-file-length)
*                   resp(ws-resp)
*         end-exec
*  send without conversion
*         exec cics web send
*                   doctoken(ws-token)
*         end-exec
     end-evaluate
*  this program stores the uploaded file for drill in a TSQ
     exec cics writeq ts
               queue("TEST")
               from(ls-buffer)
     end-exec
     exec cics return end-exec.
```

```
       aa999-exit.
           exit.
           stop run.
```

ISCXLATE SOURCE

```
/*-------------------------------------------------------*/
/* ISCXLATE.c                                            */
/* This subroutine is used to convert the EBCDIC / ASCII   */
/* character set and vice versa.                         */
/* include files         */
#include <errno.h>            /* global error numbers */
#include <iconv.h>            /* translation services */
#include <stdlib.h>           /* getopts functions */
#include <string.h>           /* strcpy, strcat functions */
#include <stdio.h>            /* standard file descriptions */
#include <unistd.h>
/* defines               */
#define ASCII_TO_EBCDIC 0           /* conversion direction */
#define EBCDIC_TO_ASCII 1           /* conversion direction */
/* function prototypes   */
/* int ISCXLATE(char *, size_t, int);  */
/* global storage        */
/* int   errno; */
/* ISCXLATE     ASCII to EBCDIC translation or                    */
/*                   EBCDIC to ASCII translation                 */
int main(char *buffer, size_t size, int direction)
{
  const char         *ip;     /* input work area pointer */
  char               *op;     /* work area pointer */
  char                opBuffer??(32767??); /* translation work area */
  size_t             InBytesLeft, OutBytesLeft; /* string counters */
  iconv_t            cd;                        /*   translate table */
  size_t             irc;             /* return code */
  int                notConverted;    /* bytes not converted */
    /* check the translation size first          */
    /* open the translation table (code pages)    */
    /* open the translation table (to/from) */
    /* from ascii to ebcdic */
    cd = iconv_open(const char "IBM-037", const char "ISO8859-1");
    InBytesLeft = size;          /* length to be converted */
    OutBytesLeft = InBytesLeft;  /* same */
    ip = (char *)buffer;         /* input pointer */
    op = &opBuffer??(0??);       /* output pointer */
    /* convert the data                          */
    irc = iconv(cd, &ip, &InBytesLeft, &op, &OutBytesLeft);
    notConverted=0;
    notConverted = irc;         /* compute unconverted bytes */
    /* error code should go here                 */
                                  /* copy the converted characters */
    memcpy(buffer, opBuffer, size);  /* into the caller's buffer */
```

```
    iconv_close(cd);                /* close the translation table */
    if ( (irc < 0)
        return (irc);               /* return to caller */
       }
    else {
        return (notConverted);      /* return to caller */
        }
}
```

CSS Style sheet – BASCSS

```
/* Style sheet optimized for IE4x COOL */
A:link{ color: #000066; }
A:visited { color: #666666; }
A:active { color: #0000FF; }
A:hover { color: #0000FF ; } /* On mouse over changes link color */
/* TOOLBAR, SBNMENU start */
TD.clsTDMSTB {
background-color:#000000;
} /* override global TD for MS Toolbar */
TABLE.clsTableBP {
padding:0;
margin:0;
background-color:#FFFFFF;
} /* override global TABLE for general boilerplate content */
table.clsTableMS {
margin:0;
padding:0;
}
TD.clsTDNB {
background-color:#003399;
}
/* #divSBNMenuBar {visibility:hidden} */
/* TOOLBAR, SBNMENU end */
TD.clsShowHide  {
width: 95;
padding:0;
background-color:#FFFFFF;
}
TABLE.clsNBIT {margin:0}
body{
margin-left: 0pt;
margin-top: 0pt;
font-size: 80%;
font-family: Verdana, Arial, Helvetica, MS Sans Serif;
}
body.samples{
background-image: url();
font-size: 80%;
font-family: Verdana, Arial, Helvetica, MS Sans Serif;
}
```

```
blockquote{
margin-left: 10pt;
margin-right: 10pt;
margin-top: 10pt;
margin-bottom: 10pt;
}
blockquote.body{
margin-left: 10pt;
margin-right: 10pt;
margin-top: 10pt;
margin-bottom: 10pt;
}
blockquote dl{ margin-top: 0pt;}
blockquote table.ref{
margin-top: 0pt;
margin-bottom: 6pt;
}
blockquote table.swtable {
margin-top: 0pt;
margin-bottom: 6pt;
}
blockquote table{
margin-top: 0pt;
margin-bottom: 6pt;
}
BUTTON.showme {
font-size:11;
font-family:Arial;
width:68;
height:23;
margin-bottom: 2pt;
position: relative; top: 2;
background-color:#002F90;
color:#FFFFFF;
font-weight:bold;
}
/* adjust horizontal position of showme for IE5 */
BUTTON.showme5 {
font-size:11;
font-family:Arial;
width:68;
height:23;
margin-left:20pt;
margin-bottom: 2pt;
position: relative; top: 2;
background-color:#002F90;
color:#FFFFFF;
font-weight:bold;
}
dl dl{ margin-top: 5pt;}
font.na { color: #B0B0B0; }
h1{
```

```
font-size: 145%;
margin-top: 1.25em;
margin-bottom: .5em;
}
h2{
font-size: 135%;
margin-top: 1.25em;
margin-bottom: .5em;
}
h3{
font-size: 128%;
    margin-top: 1em;
    margin-bottom: 0em;
}
h4{
font-size: 120%;
    margin-top: .8 em;
    margin-bottom: 0em;
}
h5{
font-size: 110%;
    margin-top: .8 em;
    margin-bottom: 0em;
}
h6{
font-size: 70%;
    margin-top: .6em;
    margin-bottom: 0em;
}
hr{color: #666666;}
hr.clsTitle {
color: #666666;
height: 1px }
p{
margin-top: .6em;
margin-bottom: .6em;
}
p.ref{
font-weight: bold ;
margin-top: 12pt;
margin-bottom: 0pt;
}
ol{
margin-top: .5em;
margin-bottom: 0em;
margin-left: 18pt;
}
ul{
margin-top: .5em;
margin-bottom: 0em;
margin-left: 18pt;
}
```

```
ol ul{
list-style: disc;
margin-top: 1em;
}
li{
padding-bottom: .2em;
margin-right: 1em;
}
li li{ list-style-type: square; }
pre{
font-family: Courier New;
font-size: 130%;
background: #EEEEEE;
margin-top: 1em;
margin-bottom: 1em;
margin-left: 0pt;
padding-top: 5pt;
padding-bottom: 5pt;
padding-left: 5pt;
padding-right: 5pt;
}
pre.syntax{
font-family: Verdana, Arial, Helvetica, MS Sans Serif;
font-size: 120%;
}
P.copyright  {
font-size: 8pt;
font-family: verdana, arial, helvetica, ms sans serif,;
margin-top: 0;
margin-bottom: 0;
}
code{
font-family: Courier New;
font-size: 130%;
background: #EEEEEE;
}
code.text{
font-family: Courier New;
font-size: 130%;
background: #FFFFFF;
}
/* used for Code within Text blocks */
table{
font-size: 100%;
margin-top: 10pt;
margin-bottom: 3pt;
padding-left: 2pt;
padding-right: 2pt;
}
th{
text-align: left;
```

```
background-color: #dddddd;
margin: 3pt;
vertical-align: bottom;
}
tr{ vertical-align: top; }
TABLE.clsTableNL{
background-color: #FFFFFF;
margin: 0;
}
td{
margin: 3pt;
background-color: #EEEEEE;
vertical-align: top;
}
TD.clsTDNL {
background-color: #FFFFFF;
padding:0;
} /* override global TD for NAV_LINKS */
table.ref{
font-size: 100%;
margin-top: 10pt;
margin-bottom: 6pt;
padding-left: 2pt;
padding-right: 2pt;
}
table.ref th{
font-weight: bold ;
text-align: left;
background-color: #CCCCCC;
margin: 3pt;
vertical-align: bottom;
}
table.ref tr{ vertical-align: top; }
table.ref td{
margin: 3pt;
background-color: #EEEEEE;
vertical-align: top;
}
table.paramvls{
margin-top: .25em;
}
a.line{
text-decoration: underline; font-size: 8pt; color: black;
}
a.glossary{
text-decoration: underline; font-size: 10pt; color: green;
}
table.swtable {
  background-color: #ffffff;
margin-top: 10pt;
margin-bottom: 3pt;
padding-left: 2pt;
```

```
padding-right: 2pt;
list-style-type: round;
}
table.swtable th{
  background-color: #CCCCCC;
vertical-align: top;
margin-right : 1em;
margin-bottom: .25em;
list-style-type: square;
}
A.clsBottomItem {
color: #000000;
text-decoration:none;
font-size: 8pt;
}
A.clsLeftMenu {
color: #000000;
text-decoration:none;
font-weight:bold; font-size: 8pt;
}
A.clsLeftMenu:visited{
color: #000000;
}

#TOC { visibility:hidden
}
/* initially hides the TOC link. See SetTOC in common.js */
.defvalue {
font-weight : bold;
font-family : 'courier new'
}
/* use to indicate when text represents a default value */
.literal {
font-family : 'courier new'
}
/* use to indicate when text represents a value the user should type */
.range { font-style : italic }
/* use to indicate when text represents a range of possible values */
DIV.showme {
margin-bottom : .5em;
margin-top: .5em;
}
SPAN.showme {
   width:100%;
filter:dropshadow(color=#000000,OffX=2.5,OffY=2.5,Positive=1);
position: relative; top: -8;
}
DIV.beta {
font-weight:bold;
color:red;
}
/* INDEX CLASSES */
```

```
BODY.index{ margin-left: 0pt; margin-top: 0pt;
font-size: 80%;font-family: Verdana, Arial, Helvetica, MS Sans Serif; }
UL.index{ margin-left: 20pt; margin-top: 0pt;
margin-bottom: 5pt; }
TABLE.index{ font-size: 100%; margin-top: 10pt;
margin-bottom: 3pt; padding-left: 2pt; padding-right: 2pt; }
TR.index{ vertical-align: top; }
TD.index{ margin: 3pt; background-color: #EEEEEE;
vertical-align: top; }
DIV.clsHi{ padding-left:2em;text-indent:-2em } /* produces a hanging
indent for entries */
TR.entry { vertical-align:top; font-size:11}
TD.letters { margin: 3pt;
vertical-align: top;
text-align:center;
background-color: #CCCCCC
}
TD.mainhead { background-color:#FFFFFF;
vertical-align: top;
font-size:145%;
margin-top:1.35em;
margin-bottom:.5em;
font-weight:bold;
} /* main heading on index entry page */
A.disabled { text-Decoration:none; color:black; cursor:text; } /*
disabled hyperlink */
A.enabled {  text-Decoration; color; cursor:auto } /* enabled hyperlink
*/
.access  {  text-decoration:underline; }
IMG.clsNoBorder {border: none 0}
SPAN.clsIDXRange {position:relative; width:100px; top:-3px} /* Position
of Index Range text */
```

David Clancy
Circle Computer Group (UK)

CICS Update on the Web

Code from individual articles of *CICS Update*, and complete issues in Acrobat PDF format, can be accessed on our Web site, at:

http://www.xephon.com/cicsupdate.html

You will be asked to enter a word from the printed issue.

IBM has announced WebSphere Version 4, claiming it handles twice the number of JDBC and EJB transactions as BEA for the same cost.

New features include better JDBC and EJB performance, while additional technologies such as JNDI cacheing and dynamic EJBs re-loading increase performance even more.

The software connects and interoperates with 35 software platforms and applications, including SAP, PeopleSoft, CICS, IMS, and host integration. It's also more tightly integrated with databases including DB2 and SQL Server, and with middleware such as MQSeries, Tivoli, and Lotus Domino.

Transaction integration is addressed through support for open Internet standards and technologies including UDDI, SOAP, Web Services Description Language (WSDL), EJB 2.0 Message Beans, and better integration with leading XML tools.

Products include WebSphere Application Server Developer Version, a free developer-only version with support for J2EE, XML, and Web services development. There are also new versions of WebSphere Studio and VisualAge for Java, all integrated with WebSphere, and including connectors for CICS, SAP, JD Edwards, PeopleSoft, and other applications, as well as support for personalization, portals, and mobile Internet.

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

* * *

Tivoli has announced Version 1.5 of its Business Systems Manager (TBSM) enterprise system management product for monitoring and controlling system management events across sites with multiple data centres and multiple platforms.

Its system management application components integrate all OS/390 and distributed platform management disciplines into a single object model and provide a single console for managing and viewing all resources.

It can be implemented as OS/390-only, distributed-only, or as both an OS/390 and distributed solution.

New features include server support for both Windows NT and Windows 2000 and a new Java GUI that supports clients running on AIX, Linux, Solaris, and Windows.

There's also a new facility to launch products from the TBSM GUI within context, plus updated and new third-party monitoring support for MVS, CICS, DB2, and IMS.

Requirements include OS/390 V2R7 or later, z/OS V1R1, NetView for OS/390 V1R2 or later, for managing CICS, DB2, IMS, or MVS environments, NT 4.0 or later, or Windows 2000. The Distributed Edition Package requires NT 4.0 or Windows 2000 and Tivoli Management Framework 3.6.1 or later.

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

* * *