CICS update

213

August 2003

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 December 1999 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.

Editor

Trevor Eddolls

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/nfc.

‘Killing’ CICS tasks with CICS Performance Monitor for z/OS

A question often asked is, “What capabilities exist to ‘kill’ a task from a given CICS system via CICS Performance Manager?”. (Here we are using the term ‘kill’ to refer to the action provided by many performance management monitors.)

CICS Performance Monitor bases its facilities on those of CICSPlex System Manager. CICSPlex SM itself is an EXEC CICS application, and therefore provides the ability to PURGE and FORCEPURGE tasks. Those facilities are exploited by CICS PM.

Killing a task in a CICS region can have catastrophic consequences; for instance, no data integrity is guaranteed. As an example, consider the consequences of purging a task in a DB2WAIT condition. Having killed the task, the DB2 subsystem could subsequently post an ECB in the CICS system where the storage is no longer the requestor’s ECB. This could have disastrous results.

Nevertheless, the customer may deem this risk acceptable. Whilst this is a concern to CICS system programmers or operators, this function is used in the scenario where a region would have to be brought down anyway to resolve the problem, and killing the task will allow the region to continue processing critical work for some time. An example would be resolving a problem in a trading organization just before the market closes.

The problem is actually subtler than this, and can be broken down into the following items:

- Which system is the problem in?
- Which task do I want to kill?
- How do I kill it?

So, we need to know which task to kill before we can kill it.

In order to understand this topic, some appreciation is required of the underlying architecture of CICSPlex SM (upon which CICS PM is built), along with a classification of types of work running within the CICS system, and the CICS system's current state.

CICSPLEX SM AGENT ARCHITECTURE

CICSPlex SM (CPSM) provides single system image management through a network of CICS Managing Address Spaces (CMASs). These CMASs communicate with CPSM agents running in the target CICS regions. For example, these agents are responsible for INQUIREing and SETting the attributes of CICS resources.

Communication from the CMAS to the CPSM agent is achieved through a communications agent, also running in the CICS region, which mediates requests from the CMAS. Communication with this agent is via MVS cross-memory services (not CICS communication services).

The communications agent, query, and set agents, along with other CPSM infrastructure services, are initialized at CICS initialization via PLT processing. This establishes long-running tasks that, essentially, process requests for the lifetime of the CICS region. These resources run on the QR TCB. This does expose them to the possibility of being blocked because looping tasks dominate the QR TCB, or delayed through resource shortage (eg storage).

CICS WORKLOAD TYPES

CICS workloads can be classified into various types for the purposes of this discussion as follows:

- Normally executing tasks – the vast majority of CICS tasks. These are susceptible to a CEMT PURGE TASK command.
- Looping tasks – these are tasks that can be looping within themselves (and therefore susceptible to CICS runaway task detection process, ICVR process). They are looping

through the CICS exec layer issuing EXEC CICS commands.

- Tasks in a purgable wait – these are susceptible to a CEMT FORCEPURGE TASK command.
- Tasks in a non-purgable wait (eg a DB2WAIT) – these are not susceptible to a CEMT FORCEPURGE command, but are susceptible to a kill command. It should be noted that the number of situations in which tasks are placed into a non-purgable wait has reduced with each CICS release. For example, dispatcher and DB2 changes in CICS TS 2.2 have resulted in the ability to purge tasks in a DB2WAIT state without the need for kill.

CICS STATE

The state of the CICS system can also play a role in being able to kill a task. If the system is at maxtask or is short on storage, then a new task (such as CEMT) cannot be dispatched. This is an instance where CPSM's long-running task architecture will be able to issue the appropriate command, whilst base CEMT would not.

Class	CEMT	CPSM, CICS PM	Third-party tools
A	Yes	Yes	Yes – normal stuff
B	No	Yes	Yes – eg region at maxtask
C	No	No	Yes – unable to dispatch on QR TCB
D	No	No	No – the rest

Figure 1: CICS states

The scenarios are summarized in Figure 1.

Class A is the majority of situations; Class B can still be performed by CPSM; Class C is a very small (and reducing) class because of CICS internal changes; Class D is minute and no-one addresses it.

CICS PURGE PROCESSING

The current CICS purge processing is as follows.

Upon receipt of a purge request, the CICS dispatcher saves the purge request information. If the purge is not successful, a purge is retried each time the task is suspended, and a deferred abend request issued. The purge may be rejected because the task is not suspended, or the task is purge/forcepurge protected.

OPPORTUNITIES TO EXTEND CICS TS IN THIS AREA

CICS PM supports CICS TS 1.3 and CICS TS 2.2. Open Transaction Environment TCBs were introduced in CICS TS 1.3. These TCBs execute independently of the CICS QR TCB. It would be possible to provide communications endpoints and INQuire/SET services from an OTE TCB. In this way, management services would still be available when the QR TCB is hung up.

Extensions to the ICVR and PURGE mechanisms could establish the need to kill a task; if not immediately attainable, this would be performed on subsequent traversal of the CICS exec layer. Other obvious places include RMI calls and DL/I command interfaces. This could exclude some of the current checks that are made to ensure data integrity. Purging of tasks could also be prioritized by age of the tasks.

SUMMARY

Some third-party performance products provide the ability to kill tasks in a CICS system. Extensions to CICS Transaction server have, in each release, reduced the need for such a function. The long-running agent architecture of CICSplex SM also reduces the number of cases in which such a function is necessary. There is, however, a very small subset of scenarios where such a function is required. We are looking at ways in which to provide such function via CICS Transaction Server, which will be utilized by a future release of CICS Performance Monitor.

Paul Johnson
CICS Transaction Server Systems Management Planning/Development
IBM Hursley (UK)

Displaying task information

The following program displays the current tasks running under CICS, with its main characteristics. The screen produced is illustrated below:

```
    CICS51TA                                         Ø3/Ø4/Ø1   Ø8:39:34
 Tasknum Tran Userid   Term Type Pri Status  Susptype Suspval  Susptime Sc Tcl
  ---------------------------------------------------------------------------
 ØØØØØ23 JNL2 STCCICS       Task 255 Suspend                   ØØØØØØØ4 S   Ø1
 ØØØØØ45 OMEG STCCICS       Task 255 Suspend USERWAIT SRVWORK  ØØØ23156 S   Ø1
 ØØØØØ46 OMEG STCCICS       Task 255 Suspend USERWAIT SR2WORK  ØØØ23156 S   Ø1
 ØØØ3364 P8EE ECCSTCV  IX76 Term ØØ1 Running                   ØØØØØØØ2 TP  Ø1
 ØØØ3365 F5RE ECCSTAW  IZ11 Term ØØ1 Running                   ØØØØØØØ2 TP  Ø1
 ØØØ3366 P8AC ECCSDEQ  IAØ1 Term ØØ1 Running                   ØØØØØØØ1 TP  Ø1
 ØØØ3367 F1CA ECCSIUG  ID29 Term ØØ1 Running                   ØØØØØØØ1 TP  Ø1
 ØØØ3368 VTAS ECICLP1  IX23 Term ØØ1 Running                   ØØØØØØØ1 TP  Ø1
```

Each time you press *Enter*, the display is refreshed. The screen is limited to 19 lines. If, in a given instant, there are more than 19 active transactions, only those fitting the screen will be dispayed.

The fields shown are task number, transaction name, userid, facility (the terminal id, if there is one), facilitytype, transaction priority, its status, suspend type and suspend value (if the task status is suspend), suspend time, startcode, and tclass. This application consists of a COBOL program and a BMS map. The transaction associated with the program (VTAS) is declared in the program's last 77 variable, and you can change it to some other name of your choice.

VITASKP SOURCE CODE

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. VITASKP.
       *
        ENVIRONMENT DIVISION.
        DATA DIVISION.
       ****************************************************************
        WORKING-STORAGE SECTION.
       ****************************************************************
        77  X                 PIC S9(4)  COMP VALUE +Ø.
        77  Z                 PIC S9(4)  COMP VALUE +Ø.
```

```
 77  W-RESP             PIC S9(8)  COMP VALUE +Ø.
 77  W-RESP2            PIC S9(8)  COMP VALUE +Ø.
 77  ABSTIME            PIC S9(15) COMP-3.
 77  NUM-AUX            PIC 9(8)   VALUE Ø.
 77  LISTSIZE1          PIC S9(8)  COMP VALUE +Ø.
 77  LISTPTR            USAGE IS POINTER.
 77  END-MESSAGE        PIC X(3)   VALUE 'END'.
 77  TRANS-NAME         PIC X(4)   VALUE 'VTAS'.
 COPY DFHAID.
***************************************************************
 Ø1 COMMAREA.
************************************************ 1625 bytes ***
     Ø2        W-TASKNUMBER    PIC S9(7)  COMP-3.
     Ø2        T-TRANSACTION   PIC  X(4).
     Ø2        T-USERID        PIC  X(8).
     Ø2        T-FACILITY      PIC  X(4).
     Ø2        W-FACILITYTYPE  PIC S9(8)  COMP.
     Ø2        W-PRIORITY      PIC S9(8)  COMP.
     Ø2        W-RUNSTATUS     PIC S9(8)  COMP.
     Ø2        T-SUSPENDTYPE   PIC  X(8).
     Ø2        T-SUSPENDVALUE  PIC  X(8).
     Ø2        W-SUSPENDTIME   PIC S9(8)  COMP.
     Ø2        T-STARTCODE     PIC  X(2).
     Ø2        W-TCLASS        PIC S9(8)  COMP.
*
     Ø2  VITASKSI.
        Ø4     FILLER          PIC  X(12).
        Ø4     CICSNL    COMP  PIC  S9(4).
        Ø4     CICSNF          PIC  X.
        Ø4     CICSNI          PIC  X(8).
        Ø4     DDATEL    COMP  PIC  S9(4).
        Ø4     DDATEF          PIC  X.
        Ø4     DDATEI          PIC  X(1Ø).
        Ø4     DTIMEL    COMP  PIC  S9(4).
        Ø4     DTIMEF          PIC  X.
        Ø4     DTIMEI          PIC  X(8).
        Ø4     SCREEN-LINES    PIC  X(152Ø).
        Ø4     LINEI REDEFINES SCREEN-LINES OCCURS 19.
           Ø6  LINEL     COMP  PIC  S9(4).
           Ø6  LINEA           PIC  X.
           Ø6  TASKNUMBER      PIC  X(7).
           Ø6  FILLER          PIC  X(1).
           Ø6  TRANSACTION     PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  USERID          PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  FACILITY        PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  FACILITYTYPE    PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  PRIORITY        PIC  X(3).
           Ø6  FILLER          PIC  X(1).
           Ø6  RUNSTATUS       PIC  X(7).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDTYPE     PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDVALUE    PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDTIME     PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  STARTCODE       PIC  X(2).
           Ø6  FILLER          PIC  X(2).
           Ø6  TCLASS          PIC  X(2).
*
     Ø2  VITASKSO REDEFINES VITASKSI PIC X(1567).
     Ø2  FILLER                PIC X(1ØØ).
*
***************************************************************
 LINKAGE SECTION.
***************************************************************
 Ø1  DFHCOMMAREA.
     Ø2  FILLER          PIC X(2ØØØ).
 Ø1  TASKLIST.
     Ø4  TASKL OCCURS 3Ø PIC S9(7) COMP-3.
***************************************************************
 PROCEDURE DIVISION.
***************************************************************
*
 FIRST-TIME-ONLY.
*===============*
     IF EIBCALEN = Ø
        MOVE LOW-VALUES TO COMMAREA
        MOVE 1625 TO EIBCALEN
        PERFORM INITIATE-SCREEN
        PERFORM INQUIRE-CICS
        PERFORM SEND-SCREEN-ERASE
        GO TO RETURN-TRANSID
     END-IF.
*
 OTHER-TIMES.
*===========*
     MOVE DFHCOMMAREA TO COMMAREA
     PERFORM RECEIVE-SCREEN
     PERFORM CLEAN-SCREEN
     PERFORM INQUIRE-CICS
     PERFORM SEND-SCREEN
     GO TO RETURN-TRANSID.
***************************************************************
*    Subroutines                                              *
***************************************************************
```

```
*
 INQUIRE-CICS.
*============*
     MOVE Ø TO X.
     EXEC CICS INQUIRE TASK LIST
          SET     (LISTPTR)
          LISTSIZE(LISTSIZE1)
     END-EXEC
     SET ADDRESS OF TASKLIST TO LISTPTR
     PERFORM INQUIRE-CICS-LOOP THRU
             INQUIRE-CICS-LOOP-EXIT UNTIL X > 19.
*
 INQUIRE-CICS-LOOP.
*=================*
     ADD 1 TO X.
     IF X > LISTSIZE1
        MOVE 99 TO X
        GO TO INQUIRE-CICS-LOOP-EXIT
     END-IF

     MOVE TASKL(X) TO W-TASKNUMBER
     EXEC CICS INQUIRE TASK        (W-TASKNUMBER)
                  TRANSACTION (T-TRANSACTION)
                  USERID      (T-USERID)
                  FACILITY    (T-FACILITY)
                  FACILITYTYPE(W-FACILITYTYPE)
                  PRIORITY    (W-PRIORITY)
                  RUNSTATUS   (W-RUNSTATUS)
                  SUSPENDTYPE (T-SUSPENDTYPE)
                  SUSPENDVALUE(T-SUSPENDVALUE)
                  SUSPENDTIME (W-SUSPENDTIME)
                  STARTCODE   (T-STARTCODE)
                  TCLASS      (W-TCLASS)
                  RESP        (W-RESP)
                  RESP2       (W-RESP2)
     END-EXEC
     IF W-RESP2 > Ø
        MOVE 99 TO X
        GO TO INQUIRE-CICS-LOOP-EXIT
     END-IF

     MOVE   W-TASKNUMBER    TO   NUM-AUX
     MOVE   NUM-AUX(2:7)    TO   TASKNUMBER(X)
     MOVE   T-TRANSACTION   TO   TRANSACTION(X)
     MOVE   T-USERID        TO   USERID(X)
     MOVE   T-FACILITY      TO   FACILITY(X)
     MOVE   W-PRIORITY      TO   NUM-AUX
     MOVE   NUM-AUX(6:3)    TO   PRIORITY(X)
     MOVE   T-SUSPENDTYPE   TO   SUSPENDTYPE(X)
     MOVE   T-SUSPENDVALUE  TO   SUSPENDVALUE(X)
     MOVE   W-SUSPENDTIME   TO   NUM-AUX
     MOVE   NUM-AUX         TO   SUSPENDTIME(X)
     MOVE   T-STARTCODE     TO   STARTCODE(X)
     MOVE   W-TCLASS        TO   NUM-AUX
     MOVE   NUM-AUX(7:2)    TO   TCLASS(X)

     IF W-RUNSTATUS    = DFHVALUE(SUSPENDED)
        MOVE 'Suspend' TO RUNSTATUS(X)
     END-IF
     IF W-RUNSTATUS    = DFHVALUE(RUNNING)
        MOVE 'Running' TO RUNSTATUS(X)
     END-IF
     IF W-RUNSTATUS    = DFHVALUE(DISPATCHABLE)
        MOVE 'Dispatc' TO RUNSTATUS(X)
     END-IF.
     IF W-FACILITYTYPE = DFHVALUE(TASK)
        MOVE 'Task'    TO FACILITYTYPE(X)
     END-IF
     IF W-FACILITYTYPE = DFHVALUE(TERM)
        MOVE 'Term'    TO FACILITYTYPE(X)
     END-IF
     IF W-FACILITYTYPE = DFHVALUE(DEST)
        MOVE 'Dest'    TO FACILITYTYPE(X)
     END-IF.
*
 INQUIRE-CICS-LOOP-EXIT.
*======================*
     EXIT.
*
 CLEAN-SCREEN.
*============*
     PERFORM CLEAN-SCREEN-LINES
             VARYING Z FROM 1 BY 1 UNTIL Z > 19.
*
 CLEAN-SCREEN-LINES.
*==================*
     MOVE SPACES TO LINEI(Z).
*
 INITIATE-SCREEN.
*===============*
     EXEC CICS ASSIGN APPLID (CICSNI)
     END-EXEC
     EXEC CICS ASKTIME ABSTIME (ABSTIME)
     END-EXEC
     EXEC CICS FORMATTIME
               ABSTIME (ABSTIME)
               DATE    (DDATEI)
               DATESEP ('/')
               TIME    (DTIMEI)
               TIMESEP (':')
```

```
              END-EXEC.
       *
        RECEIVE-SCREEN.
       *==============*
            EXEC CICS HANDLE CONDITION MAPFAIL(RETURN-EXIT)
            END-EXEC
            EXEC CICS RECEIVE MAP('VITASKS')
            END-EXEC.
            IF EIBAID = DFHPF3  OR EIBAID = DFHPF15
               GO TO RETURN-EXIT
            END-IF.
       *
        SEND-SCREEN.
       *===========*
            EXEC CICS SEND MAP('VITASKS')
                           DATAONLY
            END-EXEC.
       *
        SEND-SCREEN-ERASE.
       *=================*
            EXEC CICS SEND MAP('VITASKS')
                           ERASE
            END-EXEC.
       *
        RETURN-TRANSID.
       *==============*
            EXEC CICS RETURN
                      TRANSID  (TRANS-NAME)
                      COMMAREA (COMMAREA)
                      LENGTH   (EIBCALEN)
            END-EXEC.
       *
        RETURN-EXIT.
       *===========*
            EXEC CICS SEND
                      FROM   (END-MESSAGE)
                      LENGTH (3)
                      ERASE
            END-EXEC
            EXEC CICS RETURN
            END-EXEC
            GOBACK.
```

BMS MAP

```
MAPSET   DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB),               *
               LANG=COBOL,TIOAPFX=YES,EXTATT=MAPONLY
*
VITASKS  DFHMDI SIZE=(24,8Ø)
```

```
*
CICSN    DFHMDF POS=(Ø1,Ø4),LENGTH=Ø8,ATTRB=(ASKIP,PROT,FSET),          *
               COLOR=PINK
DDATE    DFHMDF POS=(Ø1,57),LENGTH=1Ø,ATTRB=(ASKIP,PROT),               *
               COLOR=PINK
DTIME    DFHMDF POS=(Ø1,68),LENGTH=Ø8,ATTRB=(ASKIP,PROT),               *
               COLOR=PINK
         DFHMDF POS=(Ø2,Ø1),LENGTH=Ø7,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Tasknum'
         DFHMDF POS=(Ø2,Ø9),LENGTH=Ø4,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Tran'
         DFHMDF POS=(Ø2,14),LENGTH=Ø6,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Userid'
         DFHMDF POS=(Ø2,23),LENGTH=Ø4,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Term'
         DFHMDF POS=(Ø2,28),LENGTH=Ø4,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Type'
         DFHMDF POS=(Ø2,33),LENGTH=Ø3,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Pri'
         DFHMDF POS=(Ø2,37),LENGTH=Ø6,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Status'
         DFHMDF POS=(Ø2,45),LENGTH=Ø8,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Susptype'
         DFHMDF POS=(Ø2,54),LENGTH=Ø7,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Suspval'
         DFHMDF POS=(Ø2,63),LENGTH=Ø8,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Susptime'
         DFHMDF POS=(Ø2,72),LENGTH=Ø2,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Sc'
         DFHMDF POS=(Ø2,75),LENGTH=Ø3,ATTRB=(ASKIP,PROT),               *
               COLOR=YELLOW,INITIAL='Tcl'
         DFHMDF POS=(Ø3,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=RED,                                               *
               INITIAL='-------------------------------------------*
               ----------------------------'
*
LINE-Ø1  DFHMDF POS=(Ø4,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø2  DFHMDF POS=(Ø5,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø3  DFHMDF POS=(Ø6,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø4  DFHMDF POS=(Ø7,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø5  DFHMDF POS=(Ø8,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø6  DFHMDF POS=(Ø9,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
LINE-Ø7  DFHMDF POS=(1Ø,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),               *
               COLOR=TURQUOISE
```

```
LINE-Ø8  DFHMDF POS=(11,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-Ø9  DFHMDF POS=(12,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-1Ø  DFHMDF POS=(13,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-11  DFHMDF POS=(14,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-12  DFHMDF POS=(15,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-13  DFHMDF POS=(16,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-14  DFHMDF POS=(17,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-15  DFHMDF POS=(18,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-16  DFHMDF POS=(19,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-17  DFHMDF POS=(2Ø,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-18  DFHMDF POS=(21,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-19  DFHMDF POS=(22,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
*
         DFHMDF POS=(23,Ø1),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=RED,                                              *
               INITIAL='-----------------------------------------------*
               ------------------------------'
         DFHMDF POS=(24,6Ø),LENGTH=13,ATTRB=(ASKIP,PROT),              *
               COLOR=NEUTRAL,INITIAL='PF3/PF15  End'
*
         DFHMSD TYPE=FINAL
         END
```

Systems Programmer (Portugal) © Xephon 2003

Using socket programs to send e-mail from CICS

INTRODUCTION

Most of the transaction processing systems running on the mainframe, including CICS, communicate through networks based on the SNA (System Network Architecture) protocol, which was developed by IBM. Although SNA has proved to be a reliable and secure protocol, the increasing need to connect to more open systems necessitated the introduction of the TCP/IP stack on the mainframe. Now, most CICS systems around the world provide the TCP/IP stack and the necessary socket interface to connect to any other system supporting the TCP/IP protocol. Socket programs provide the most elegant way to connect to other systems hosted on widely-different platforms.

This article deals with one such practical usage of socket programming wherein sockets are used to connect to a local SMTP server and send a mail from a CICS application program.

PRACTICAL APPLICATIONS

The following are some of the practical cases where such a socket program that sends mail from a CICS system can prove to be useful:

- With the increasing demand for higher availability of transactional systems, it has become imperative to keep the users and system programmers of any CICS system informed about major problems occurring in the production CICS region. Whenever a major application or CICS ABEND occurs, the system programmers/developers/users can be informed of the same, using CICS generated e-mails.
- CICS socket programs can be used to send daily MIS reports/extracts at the end of the day to relevant project people.

E-MAIL FROM CICS – A SYNOPSIS OF COMMON METHODS USED

The following are some of the most common methods used to send SMTP mail from CICS:

- Spool is provided by CICS as an interface to JES. The spool can be used to send e-mails from CICS where the OUTPUT NODE is set to the SMTP class.

- An extra-partitioned TDQ defined in the DCT can be used for this but has the inherent problem of not throwing any mail to the SMTP server unless the file associated with it is de-allocated from CICS.
- Sockets can be used to connect to any local SMTP server. Currently, this is the most elegant and widely-used method of sending e-mail from a transaction-processing environment.

SOCKET APIS

The socket Application Programming Interface (API) provides a set of system programs that establish a connection to other systems on the network, send and receive data between applications, and close the connections. The following is a schematic representation of the socket APIs in the OSI (Open Systems Interconnection) model – see Figure 1.

The first COBOL Socket API provided with TCP/IP 2.2.1 for MVS is EZACICAL. Presently, the COBOL socket API most widely

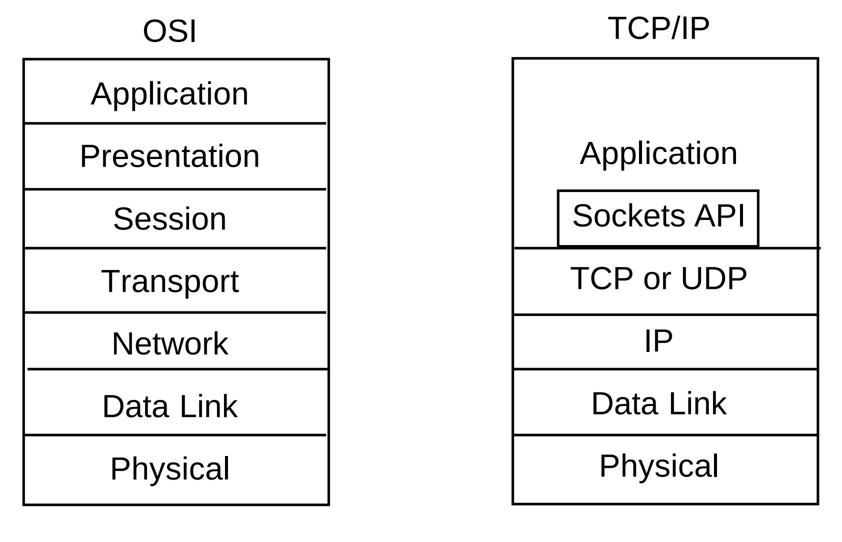

Figure 1: OSI APIs

used is EZASOKET, and this is used for the socket application described in this article.

SYSTEM REQUIREMENTS

A complete description of all the system requirements for this application is beyond the scope of this article. As a pre-requisite for running this application, it has been assumed that a TCP/IP stack along with the socket-programming interface is in the CICS system and is active.

Furthermore, the following points should be kept in mind before attempting to run the application:

- The TCPIP SEZALINK library should be concatenated to the STEPLIB of the CICS IPL job.
- The TCPIP SEZATCP library should be concatenated to the RPL of the CICS IPL job.
- The TCPDATA SYSOUT dataset should be allocated.
- The EZACONFG VSAM dataset should be defined to the local CICS region where this application will run. This dataset contains local TCP/IP definitions.
- The CICS Socket Interface and the CICS Listener should be started using the EZAO transaction.
- There should be an INCLUDE SYSLIB (EZACICAL) in the link-edit step of the job while compiling this application program.

APPLICATION DESCRIPTION

The following are the steps for completing a socket connection to the SMTP server and sending an e-mail from a CICS application program. The application code should also be referred to at this stage for easy understanding:

1 Initialize the TCP/IP environment using the EZASOKET

SOKET-INITAPI call. A non-negative return code indicates that the call was successful.

2 Create a socket using the EZASOKET SOKET-SOCKET call. A positive return code indicates that the call was successful and is actually the socket ID that is used for further EZASOKET calls.

3 Open a TCP connection to the mail server using the EZASOKET SOKET-CONNECT call. The IP address, in Network Byte Address format, and the port number of the SMTP server have to be passed as parameters in the call.

4 An EZASOKET SOKET-RECV is done to test that the SMTP server is ready to talk. An SMTP response of 220 indicates that the SMTP server is ready to talk.

5 The ‘HELO’ command identifying the mainframe system ID is written to the SMTP server using the EZASOKET SOKET-WRITE call. An SMTP response of 250 indicates that the call was successful.

6 The ‘MAIL FROM’ command identifying the mail sender is written to the SMTP server again using the EZASOKET SOKET-WRITE call. Again the SMTP response should be 250.

7 The ‘RCPT TO’ command identifying the intended recipient of the mail is written to the SMTP server using the EZASOKET SOKET-WRITE call. Similarly the SMTP response should be 250.

8 The ‘DATA’ command is issued using another EZASOKET SOKET-WRITE call, which notifies the SMTP server that the text of the message is coming.

9 The actual message text is then written using the EZASOKET SOKET-WRITE call again. The mail message consists of two parts – the header containing the Subject, From, Date, and To tagged lines, and the actual message body. A period (full stop) indicates the end of the message.

10 A ‘QUIT’ command indicates the completion of the conversation to the SMTP server.

11 Finally, the socket is closed using the EZASOKET SOKET-CLOSE call.

Notes:

- The EZASOKET call syntax is similar for all the EZASOKET calls. The call syntax is something like:

  ```
  CALL ‘EZASOKET’USING SOCK-FUNC OTHER-PARMS ERRNO RETCODE.
  ```

  The first parameter SOCK-FUNC is a 16-byte socket function. The ERRNO and RETCODE fields return the results of the call. In the case of a SOKET-SOCKET call, which creates a socket, the RETCODE is actually the Socket-ID that is used to identify the socket in further EZASOKET calls. The OTHER-PARMS parameter varies a little depending on the socket call being done. For example, in a SOKET-CONNECT call to connect to the SMTP server, the OTHER-PARMS comprises the socket description and the IP address/port number of the SMTP server that we are connecting to.

- We need to terminate our command sequences correctly. All SMTP commands must be terminated with CRLF.

- Before sending any text to the SMTP server, we must convert it to ASCII. The program EZACIC04 is used for the purpose. Similarly, when we receive the response from the SMTP server, we have to convert it into EBCDIC. The program EZACIC05 is used for the purpose.

CONCLUSION

This article has demonstrated a simple CICS application that sends a basic text message from a CICS region by connecting to a local SMTP server. This article aimed to open the door of a mainframe, hitherto considered to be a closed system, to the outside world of Unix, Windows, and more open systems. It should be noted that this application supports the sending of only a basic text mail message, but there should be nothing to prevent readers from extrapolating these basic concepts to develop a mail application that supports file attachments or binary data like a JPEG or an executable.Also, there does not seem to be any technical impediment to developing an SMTP server application in CICS.

REFERENCE

Z/OS V1R4.0 Communications Server IP CICS Sockets Guide.

ACKNOWLEDGEMENT

The author wishes to acknowledge the contribution of all his colleagues who helped him at some time or the other in the preparation of this article and the mail application. Acknowledgement is also due to the company management for allowing the author to use the mainframe environment for developing this application.

APPENDIX

FDIPG18 is the CICS program. Please note that you should have a PCT entry for FDIPG18 and FI07 (the related transaction) in your system. Otherwise, modify the program to change the program and transaction name according to your installation standard.

The file FDIMP21PHY is the physical map. I have assumed a PPT map entry by the name of FDIMP21. Please modify the map to reflect your installation standard PPT entry.

The file FDIMP21SYM is the corresponding copybook (symbolic map) for FDIMP21PHY.

FDIPG18

```
ØØØ1ØØ IDENTIFICATION DIVISION.
ØØØ2ØØ PROGRAM-ID. FDIPG18.
```

```
ØØØ3ØØ AUTHOR. MANAS.
ØØØ4ØØ DATE-WRITTEN. Ø3/26/2ØØ3.
ØØØ5ØØ DATE-COMPILED. Ø3/26/2ØØ3.
ØØØ51Ø*
ØØØ52Ø ENVIRONMENT DIVISION.
ØØØ53Ø*
ØØØ54Ø DATA DIVISION.
ØØØ55Ø WORKING-STORAGE SECTION.
ØØØ551 COPY DFHAID.
ØØØ553 COPY FDIMP21.
ØØØ561 Ø1 WS-MESSAGE               PIC X(21)
ØØØ562                                    VALUE 'SESSION COMPLETED....'.
ØØØ563 Ø1 WS-COMMAREA              PIC X.
ØØØ564
ØØØ565 Ø1 WS-ERROR.
ØØØ566    Ø2 WS-ERROR-1            PIC X(6)  VALUE 'ERROR:'.
ØØØ567    Ø2 WS-ERROR-2            PIC X(2Ø) VALUE SPACES.
ØØØ568    Ø2 WS-ERROR-3            PIC X(11) VALUE 'ERROR CODE:'.
ØØØ569    Ø2 WS-ERROR-4            PIC X(18).
ØØØ57Ø
ØØØ571 Ø1  WS-NUM-ERROR             PIC 9(18).
ØØØ572
ØØØ573 Ø1 SOKET-FUNCTIONS.
ØØØ574    Ø2 SOKET-ACCEPT          PIC X(16) VALUE 'ACCEPT          '.
ØØØ575    Ø2 SOKET-BIND            PIC X(16) VALUE 'BIND            '.
ØØØ576    Ø2 SOKET-CLOSE           PIC X(16) VALUE 'CLOSE           '.
ØØØ577    Ø2 SOKET-CONNECT         PIC X(16) VALUE 'CONNECT         '.
ØØØ578    Ø2 SOKET-FCNTL           PIC X(16) VALUE 'FCNTL           '.
ØØØ579    Ø2 SOKET-GETCLIENTID     PIC X(16) VALUE 'GETCLIENTID     '.
ØØØ58Ø    Ø2 SOKET-GETHOSTBYADDR   PIC X(16) VALUE 'GETHOSTBYADDR   '.
ØØØ581    Ø2 SOKET-GETHOSTBYNAME   PIC X(16) VALUE 'GETHOSTBYNAME   '.
ØØØ582    Ø2 SOKET-GETHOSTID       PIC X(16) VALUE 'GETHOSTID       '.
ØØØ583    Ø2 SOKET-GETHOSTNAME     PIC X(16) VALUE 'GETHOSTNAME     '.
ØØØ584    Ø2 SOKET-GETPEERNAME     PIC X(16) VALUE 'GETPEERNAME     '.
ØØØ585    Ø2 SOKET-GETSOCKNAME     PIC X(16) VALUE 'GETSOCKNAME     '.
ØØØ586    Ø2 SOKET-GETSOCKOPT      PIC X(16) VALUE 'GETSOCKOPT      '.
ØØØ587    Ø2 SOKET-GIVESOCKET      PIC X(16) VALUE 'GIVESOCKET      '.
ØØØ588    Ø2 SOKET-INITAPI         PIC X(16) VALUE 'INITAPI         '.
ØØØ589    Ø2 SOKET-IOCTL           PIC X(16) VALUE 'IOCTL           '.
ØØØ59Ø    Ø2 SOKET-LISTEN          PIC X(16) VALUE 'LISTEN          '.
ØØØ591    Ø2 SOKET-READ            PIC X(16) VALUE 'READ            '.
ØØØ592    Ø2 SOKET-RECV            PIC X(16) VALUE 'RECV            '.
ØØØ593    Ø2 SOKET-RECVFROM        PIC X(16) VALUE 'RECVFROM        '.
ØØØ594    Ø2 SOKET-SELECT          PIC X(16) VALUE 'SELECT          '.
ØØØ595    Ø2 SOKET-SEND            PIC X(16) VALUE 'SEND            '.
ØØØ596    Ø2 SOKET-SENDTO          PIC X(16) VALUE 'SENDTO          '.
ØØØ597    Ø2 SOKET-SETSOCKOPT      PIC X(16) VALUE 'SETSOCKOPT      '.
ØØØ598    Ø2 SOKET-SHUTDOWN        PIC X(16) VALUE 'SHUTDOWN        '.
ØØØ599    Ø2 SOKET-SOCKET          PIC X(16) VALUE 'SOCKET          '.
ØØØ6ØØ    Ø2 SOKET-TAKESOCKET      PIC X(16) VALUE 'TAKESOCKET      '.
```

```
   Ø2 SOKET-TERMAPI         PIC X(16) VALUE 'TERMAPI         '.
   Ø2 SOKET-WRITE           PIC X(16) VALUE 'WRITE           '.

Ø1 SOCTYPE                  PIC 9(8) COMP VALUE 1.
Ø1 PROTO                    PIC 9(8) COMP VALUE Ø.
Ø1 SOCKID                   PIC 9(4) COMP.
Ø1 ERRNO                    PIC 9(8) COMP.
Ø1 RETCODE                  PIC S9(8) COMP.
Ø1 AF-INET                  PIC 9(8) COMP VALUE 2.
Ø1 RECV-FLAGS.
   Ø2 NO-FLAGS              PIC 9(8) COMP  VALUE Ø.
   Ø2 OOB                   PIC 9(8) COMP  VALUE 1.
   Ø2 PEEK                  PIC 9(8) COMP  VALUE 2.

Ø1 RECV-BYTE                PIC 9(8) COMP VALUE 1ØØØ.
Ø1 RECV-BUF                 PIC X(1ØØØ).

Ø1 SOCKADDR.
   Ø5  FAMILY               PIC 9(4)  COMP VALUE Ø.
   Ø5  PORTNO               PIC 9(4)  COMP VALUE Ø.
   Ø5  IPADDR               PIC X(4).
   Ø5  RESERVEM             PIC X(8)  VALUE LOW-VALUES.

Ø1 WRITE-TCPLENG            PIC 9(8) COMP.
Ø1 WRITE-TCPBUF             PIC X(1ØØ).
Ø1 TOEBCDIC-TOKEN           PIC X(16) VALUE 'TCPIPTOEBCDICXLT'.
Ø1 TOASCII-TOKEN            PIC X(16) VALUE 'TCPIPTOASCIIXLAT'.
Ø1 WS-CRLF                  PIC X(2) VALUE X'ØD15'.
Ø1 MAXSOC                   PIC 9(4) COMP VALUE 2Ø.
Ø1 INIT-IDENT.
   Ø5 INIT-NAME             PIC X(8).
   Ø5 INIT-ADRS             PIC X(8).
Ø1 INIT-CICSTSK             PIC X(8).
Ø1 MAXSNO                   PIC 9(8) COMP VALUE 19.
Ø1 TEMP-HEX-IP              PIC 9(4) COMP.
Ø1 TEMP-HEX-IP-NUM REDEFINES TEMP-HEX-IP.
   Ø2 TEMP-WASTE-IP         PIC X.
   Ø2 TEMP-HEXIP-X2         PIC X.
Ø1 WS-SUBJECT-INFO          PIC X(44)
            VALUE 'SUBJECT: MAIL FROM CTS MAINFRAME SMTP SERVER'.

LINKAGE SECTION.
Ø1   DFHCOMMAREA            PIC X.
*
PROCEDURE DIVISION.

     IF EIBCALEN = ZERO
        PERFORM 1ØØØ-FIRST-PASS THRU 1ØØØ-EXIT
     ELSE
        PERFORM 2ØØØ-NEXT-PASS  THRU 2ØØØ-EXIT
```

```
ØØØ653      END-IF.
ØØØ654      EXEC CICS RETURN
ØØØ655      END-EXEC.
ØØØ656
ØØØ657 1ØØØ-FIRST-PASS.
ØØØ658
ØØØ659      EXEC CICS
ØØØ66Ø           SEND MAP('FDIMP21')
ØØØ661                MAPSET('FDIMP21')
ØØØ662                MAPONLY
ØØØ663                ERASE
ØØØ664                FREEKB
ØØØ665                FRSET
ØØØ666      END-EXEC.
ØØØ667      EXEC CICS
ØØØ668           RETURN
ØØØ669           TRANSID('FIØ7')
ØØØ67Ø           COMMAREA(WS-COMMAREA)
ØØØ671           LENGTH(LENGTH OF WS-COMMAREA)
ØØØ672      END-EXEC.
ØØØ673
ØØØ68Ø 1ØØØ-EXIT.
ØØØ7ØØ      EXIT.
ØØØ8ØØ
ØØØ9ØØ 2ØØØ-NEXT-PASS.
ØØØ9Ø1
ØØØ9Ø2      IF EIBAID = DFHCLEAR
ØØØ9Ø3          PERFORM 9999-END-SESSION THRU 9999-EXIT
ØØØ9Ø4      END-IF.
ØØØ9Ø5
ØØØ91Ø      EXEC CICS HANDLE CONDITION
ØØØ911           MAPFAIL(21ØØ-MAPFAIL)
ØØØ92Ø      END-EXEC.
ØØØ93Ø      EXEC CICS
ØØØ931           RECEIVE MAP('FDIMP21')
ØØØ932                   MAPSET('FDIMP21')
ØØØ933                   INTO(FDIMP21I)
ØØØ94Ø      END-EXEC.
ØØØ95Ø
ØØØ96Ø      EVALUATE EIBAID
ØØØ961          WHEN DFHCLEAR
ØØØ962               PERFORM 9999-END-SESSION THRU 9999-EXIT
ØØØ963          WHEN DFHENTER
ØØØ964               CONTINUE
ØØØ965          WHEN OTHER
ØØØ966               PERFORM 22ØØ-INVKEY      THRU 22ØØ-EXIT
ØØØ97Ø      END-EVALUATE.
ØØØ98Ø
ØØØ99Ø      PERFORM 23ØØ-DATA-VALID           THRU 23ØØ-EXIT.
ØØØ991      PERFORM 24ØØ-TCP-PROCESS          THRU 24ØØ-EXIT.
```

```
     PERFORM 2500-SUCCESS-SEND         THRU 2500-EXIT.

2000-EXIT.
     EXIT.

2100-MAPFAIL.

     MOVE 'NO DATA ENTERED..PLS ENTER RELEVANT DATA'
                                       TO MESGI.
     PERFORM 9998-SNDDATA              THRU 9998-EXIT.

2100-EXIT.
     EXIT.

2200-INVKEY.

     MOVE 'ONLY ENTER AND CLEAR KEYS ARE ACTIVE'
                                       TO MESGI.
     PERFORM 9998-SNDDATA              THRU 9998-EXIT.

2200-EXIT.
     EXIT.

2300-DATA-VALID.

     IF MAILIDI = SPACES OR LOW-VALUES
        MOVE 'ENTER A VALID MAIL ID..' TO MESGI
        PERFORM 9998-SNDDATA           THRU 9998-EXIT
     END-IF.

     IF MAILMGI = SPACES OR LOW-VALUES
        MOVE 'ENTER A VALID MAIL MSG..'
                                       TO MESGI
        PERFORM 9998-SNDDATA           THRU 9998-EXIT
     END-IF.

2300-EXIT.
     EXIT.

2400-TCP-PROCESS.

     PERFORM 2405-INIT-ENVIRON         THRU 2405-EXIT.
     PERFORM 2410-CREATE-SOCKET        THRU 2410-EXIT.
     PERFORM 2420-TCP-CONNECT          THRU 2420-EXIT.
     PERFORM 2430-TEST-SERVER          THRU 2430-EXIT.
     PERFORM 2440-HELO-WRITE           THRU 2440-EXIT.
     PERFORM 2450-MAIL-WRITE           THRU 2450-EXIT.
     PERFORM 2460-RCPT-WRITE           THRU 2460-EXIT.
     PERFORM 2470-DATA-WRITE           THRU 2470-EXIT.
     PERFORM 2480-HEADER-WRITE         THRU 2480-EXIT.
     PERFORM 249Ø-MAILMSG-WRITE         THRU 249Ø-EXIT.
     PERFORM 2491-END-MAILMSG           THRU 2491-EXIT.
     PERFORM 2492-QUIT-SMTP             THRU 2492-EXIT.
     PERFORM 2499-CLOSE-SOCKET          THRU 2499-EXIT.

 24ØØ-EXIT.
     EXIT.

 24Ø5-INIT-ENVIRON.

*  INITIALIZE THE ENVIRONMENT
     MOVE Ø                             TO ERRNO.
     MOVE Ø                             TO RETCODE.
     MOVE 'TCPIP'                       TO INIT-NAME.
     EXEC CICS
          ASSIGN APPLID(INIT-ADRS)
     END-EXEC.
     MOVE EIBTASKN                      TO INIT-CICSTSK(1:4).
     MOVE 'ØØØ'                         TO INIT-CICSTSK(5:3).
     MOVE 'C'                           TO INIT-CICSTSK(8:1).

     CALL 'EZASOKET' USING SOKET-INITAPI MAXSOC INIT-IDENT
                           INIT-CICSTSK MAXSNO ERRNO RETCODE.

     IF RETCODE < Ø
        MOVE 'SOCKET INITAPI'           TO WS-ERROR-2
        MOVE ERRNO                      TO WS-NUM-ERROR
        MOVE WS-NUM-ERROR               TO WS-ERROR-4
        MOVE WS-ERROR                   TO MESGI
        PERFORM 9998-SNDDATA            THRU 9998-EXIT
     END-IF.

 24Ø5-EXIT.
     EXIT.

 241Ø-CREATE-SOCKET.

*  CREATE A SOCKET
     MOVE Ø                             TO ERRNO.
     MOVE Ø                             TO RETCODE.

     CALL 'EZASOKET' USING SOKET-SOCKET AF-INET SOCTYPE PROTO
                           ERRNO RETCODE.

     IF RETCODE < Ø
        MOVE 'CREATE SOCKET'            TO WS-ERROR-2
        MOVE ERRNO                      TO WS-NUM-ERROR
        MOVE WS-NUM-ERROR               TO WS-ERROR-4
        MOVE WS-ERROR                   TO MESGI
        PERFORM 9998-SNDDATA            THRU 9998-EXIT
```

```
      ELSE
         MOVE RETCODE                      TO SOCKID
      END-IF.

 2410-EXIT.
      EXIT.

 2420-TCP-CONNECT.

*  OPEN A TCP CONNECTION TO THE MAIL SERVER
      MOVE AF-INET                         TO FAMILY.
      MOVE 25                              TO PORTNO.
      MOVE 162                             TO TEMP-HEX-IP.
      MOVE TEMP-HEXIP-X2                   TO IPADDR(1:1).
      MOVE 44                              TO TEMP-HEX-IP.
      MOVE TEMP-HEXIP-X2                   TO IPADDR(2:1).
      MOVE 9                               TO TEMP-HEX-IP.
      MOVE TEMP-HEXIP-X2                   TO IPADDR(3:1).
      MOVE 99                              TO TEMP-HEX-IP.
      MOVE TEMP-HEXIP-X2                   TO IPADDR(4:1).

      MOVE 0                               TO ERRNO.
      MOVE 0                               TO RETCODE.

      CALL 'EZASOKET' USING SOKET-CONNECT SOCKID SOCKADDR
                            ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'OPEN CONN'                  TO WS-ERROR-2
         MOVE ERRNO                        TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR                 TO WS-ERROR-4
         MOVE WS-ERROR                     TO MESGI
         PERFORM 9998-SNDDATA              THRU 9998-EXIT
      END-IF.

 2420-EXIT.
      EXIT.

 2430-TEST-SERVER.

*  SOCKET RECEIVE TO TEST WHETHER THE SERVER IS READY TO TALK
      MOVE 0                               TO ERRNO.
      MOVE 0                               TO RETCODE.
      MOVE 1000                            TO RECV-BYTE.

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET RECV'                TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

      CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

      IF RECV-BUF(1:3) NOT = '220'
         MOVE 'SOCKET RECV RESP'        TO WS-ERROR-2
         MOVE RECV-BUF                  TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

 2430-EXIT.
      EXIT.

 2440-HELO-WRITE.

*  HELO SOCKET WRITE COMMAND
      STRING 'HELO CPAC' WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 11                           TO WRITE-TCPLENG.
      CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET WRITE HELO'       TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

*  CHECK SMTP RESPONSE
      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      MOVE 1000                         TO RECV-BYTE.

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET RECV HELO'        TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

      CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

      IF RECV-BUF(1:3) NOT = '250'
         MOVE 'HELO SMTP'               TO WS-ERROR-2
         MOVE RECV-BUF                  TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

 2440-EXIT.
      EXIT.

 2450-MAIL-WRITE.

*  MAIL SOCKET WRITE COMMAND
      MOVE SPACES                       TO WRITE-TCPBUF.
      STRING 'MAIL FROM: <BMANAS@CAL.COGNIZANT.COM>'
              WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 39                           TO WRITE-TCPLENG.
      CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET WRITE MAIL'       TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

*  CHECK SMTP RESPONSE
      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      MOVE 1000                         TO RECV-BYTE.

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

      IF RETCODE < Ø
         MOVE 'SOCKET RECV MAIL'        TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

      CALL 'EZACICØ5' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

      IF RECV-BUF(1:3) NOT = '25Ø'
         MOVE 'MAIL SMTP'               TO WS-ERROR-2
         MOVE RECV-BUF                  TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

 245Ø-EXIT.
      EXIT.

 246Ø-RCPT-WRITE.

*  RCPT SOCKET WRITE COMMAND
      MOVE SPACES                       TO WRITE-TCPBUF.
      STRING 'RCPT TO: <' MAILIDI '>'
              WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 39                           TO WRITE-TCPLENG.
      CALL 'EZACICØ4' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE Ø                            TO ERRNO.
      MOVE Ø                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < Ø
         MOVE 'SOCKET WRITE RCPT'       TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

*  CHECK SMTP RESPONSE
      MOVE Ø                            TO ERRNO.
      MOVE Ø                            TO RETCODE.
      MOVE 1ØØØ                         TO RECV-BYTE.

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
```

```
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

     IF RETCODE < 0
         MOVE 'SOCKET RECV RCPT'        TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
     END-IF.

     CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

     IF RECV-BUF(1:3) NOT = '250'
         MOVE 'RCPT SMTP'               TO WS-ERROR-2
         MOVE RECV-BUF                  TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
     END-IF.

 2460-EXIT.
     EXIT.

 2470-DATA-WRITE.

*  DATA SOCKET WRITE COMMAND
     MOVE SPACES                        TO WRITE-TCPBUF.
     STRING 'DATA'
             WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
     MOVE 6                             TO WRITE-TCPLENG.
     CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

     MOVE 0                             TO ERRNO.
     MOVE 0                             TO RETCODE.
     CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

     IF RETCODE < 0
         MOVE 'SOCKET WRITE DATA'       TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
     END-IF.

*  CHECK SMTP RESPONSE
     MOVE 0                             TO ERRNO.
     MOVE 0                             TO RETCODE.
     MOVE 1000                          TO RECV-BYTE.
```

```

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET RECV DATA'        TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

      CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

      IF RECV-BUF(1:3) NOT = '354'
         MOVE 'DATA SMTP'               TO WS-ERROR-2
         MOVE RECV-BUF                  TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

 2470-EXIT.
      EXIT.

 2480-HEADER-WRITE.

*  HEADER SOCKET WRITE COMMAND

*  WRITE THE 'TO' HEADER
      MOVE SPACES                       TO WRITE-TCPBUF.
      STRING 'TO: <' MAILIDI '>'
              WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 34                           TO WRITE-TCPLENG.
      CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET WRITE TO HDR'     TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

```

```
*  WRITE THE 'SUBJECT' HEADER
      MOVE SPACES                       TO WRITE-TCPBUF.
      STRING WS-SUBJECT-INFO WS-CRLF
                      DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 46                           TO WRITE-TCPLENG.
      CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET WRITE SUB HDR'    TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

*  WRITE THE 'FROM' HEADER
      MOVE SPACES                       TO WRITE-TCPBUF.
      STRING 'FROM: <BMANAS@CAL.COGNIZANT.COM>'
              WS-CRLF DELIMITED BY SIZE
                                      INTO WRITE-TCPBUF.
      MOVE 34                           TO WRITE-TCPLENG.
      CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE 0                            TO ERRNO.
      MOVE 0                            TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < 0
         MOVE 'SOCKET WRITE FRM HDR'    TO WS-ERROR-2
         MOVE ERRNO                     TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR              TO WS-ERROR-4
         MOVE WS-ERROR                  TO MESGI
         PERFORM 9998-SNDDATA           THRU 9998-EXIT
      END-IF.

 2480-EXIT.
      EXIT.

 2490-MAILMSG-WRITE.

*  MAILMSG SOCKET WRITE COMMAND
```

```
ØØ231Ø      MOVE SPACES                       TO WRITE-TCPBUF.
ØØ2311      STRING MAILMGI WS-CRLF
ØØ2312                      DELIMITED BY SIZE
ØØ2313                                      INTO WRITE-TCPBUF.
ØØ2314      MOVE 61                           TO WRITE-TCPLENG.
ØØ2315      CALL 'EZACICØ4' USING TOASCII-TOKEN WRITE-TCPBUF
ØØ2316                            WRITE-TCPLENG.
ØØ2317
ØØ2318      MOVE Ø                            TO ERRNO.
ØØ2319      MOVE Ø                            TO RETCODE.
ØØ232Ø      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
ØØ2321                            WRITE-TCPBUF ERRNO RETCODE.
ØØ2322
ØØ2323      IF RETCODE < Ø
ØØ2324         MOVE 'SOCKET WRITE MAILMSG'    TO WS-ERROR-2
ØØ2325         MOVE ERRNO                     TO WS-NUM-ERROR
ØØ2326         MOVE WS-NUM-ERROR              TO WS-ERROR-4
ØØ2327         MOVE WS-ERROR                  TO MESGI
ØØ2328         PERFORM 9998-SNDDATA           THRU 9998-EXIT
ØØ2329      END-IF.
ØØ233Ø
ØØ2365 249Ø-EXIT.
ØØ2366      EXIT.
ØØ2367
ØØ2368 2491-END-MAILMSG.
ØØ2369
ØØ237Ø*  END MAILMSG SOCKET WRITE COMMAND
ØØ2371      MOVE SPACES                       TO WRITE-TCPBUF.
ØØ2372      STRING '.' WS-CRLF
ØØ2373                      DELIMITED BY SIZE
ØØ2374                                      INTO WRITE-TCPBUF.
ØØ2375      MOVE 3                            TO WRITE-TCPLENG.
ØØ2376      CALL 'EZACICØ4' USING TOASCII-TOKEN WRITE-TCPBUF
ØØ2377                            WRITE-TCPLENG.
ØØ2378
ØØ2379      MOVE Ø                            TO ERRNO.
ØØ238Ø      MOVE Ø                            TO RETCODE.
ØØ2381      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
ØØ2382                            WRITE-TCPBUF ERRNO RETCODE.
ØØ2383
ØØ2384      IF RETCODE < Ø
ØØ2385         MOVE 'SOCKET WRITE ENDMAILMSG' TO WS-ERROR-2
ØØ2386         MOVE ERRNO                     TO WS-NUM-ERROR
ØØ2387         MOVE WS-NUM-ERROR              TO WS-ERROR-4
ØØ2388         MOVE WS-ERROR                  TO MESGI
ØØ2389         PERFORM 9998-SNDDATA           THRU 9998-EXIT
ØØ239Ø      END-IF.
ØØ2391
ØØ2392*  CHECK SMTP RESPONSE
ØØ2393      MOVE Ø                            TO ERRNO.
```

```
      MOVE Ø                             TO RETCODE.
      MOVE 1ØØØ                          TO RECV-BYTE.

      CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                            RECV-BYTE RECV-BUF ERRNO RETCODE.

      IF RETCODE < Ø
         MOVE 'SOCKET RECV ENDMAILMSG'  TO WS-ERROR-2
         MOVE ERRNO                      TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR               TO WS-ERROR-4
         MOVE WS-ERROR                   TO MESGI
         PERFORM 9998-SNDDATA            THRU 9998-EXIT
      END-IF.

      CALL 'EZACICØ5' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

      IF RECV-BUF(1:3) NOT = '25Ø'
         MOVE 'ENDMSG'                   TO WS-ERROR-2
         MOVE RECV-BUF                   TO WS-ERROR-4
         MOVE WS-ERROR                   TO MESGI
         PERFORM 9998-SNDDATA            THRU 9998-EXIT
      END-IF.

 2491-EXIT.
      EXIT.

 2492-QUIT-SMTP.

*  QUIT SMTP SOCKET WRITE COMMAND
      MOVE SPACES                        TO WRITE-TCPBUF.
      STRING 'QUIT' WS-CRLF
                      DELIMITED BY SIZE
                                       INTO WRITE-TCPBUF.
      MOVE 6                             TO WRITE-TCPLENG.
      CALL 'EZACICØ4' USING TOASCII-TOKEN WRITE-TCPBUF
                            WRITE-TCPLENG.

      MOVE Ø                             TO ERRNO.
      MOVE Ø                             TO RETCODE.
      CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
                            WRITE-TCPBUF ERRNO RETCODE.

      IF RETCODE < Ø
         MOVE 'SOCKET WRITE QUITSMTP'   TO WS-ERROR-2
         MOVE ERRNO                      TO WS-NUM-ERROR
         MOVE WS-NUM-ERROR               TO WS-ERROR-4
         MOVE WS-ERROR                   TO MESGI
         PERFORM 9998-SNDDATA            THRU 9998-EXIT
      END-IF.

```

```
*  CHECK SMTP RESPONSE
     MOVE 0                             TO ERRNO.
     MOVE 0                             TO RETCODE.
     MOVE 1000                          TO RECV-BYTE.

     CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
                           RECV-BYTE RECV-BUF ERRNO RETCODE.

     IF RETCODE < 0
        MOVE 'SOCKET RECV QUITSMTP'     TO WS-ERROR-2
        MOVE ERRNO                      TO WS-NUM-ERROR
        MOVE WS-NUM-ERROR               TO WS-ERROR-4
        MOVE WS-ERROR                   TO MESGI
        PERFORM 9998-SNDDATA            THRU 9998-EXIT
     END-IF.

     CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.

     IF RECV-BUF(1:3) NOT = '221'
        MOVE 'QUITSMTP'                 TO WS-ERROR-2
        MOVE RECV-BUF                   TO WS-ERROR-4
        MOVE WS-ERROR                   TO MESGI
        PERFORM 9998-SNDDATA            THRU 9998-EXIT
     END-IF.

 2492-EXIT.
     EXIT.

 2499-CLOSE-SOCKET.

*  SOCKET CLOSE COMMAND
     MOVE 0                             TO ERRNO.
     MOVE 0                             TO RETCODE.

     CALL 'EZASOKET' USING SOKET-CLOSE SOCKID ERRNO RETCODE.

     IF RETCODE < 0
        MOVE 'SOCKET CLOSE'             TO WS-ERROR-2
        MOVE ERRNO                      TO WS-NUM-ERROR
        MOVE WS-NUM-ERROR               TO WS-ERROR-4
        MOVE WS-ERROR                   TO MESGI
        PERFORM 9998-SNDDATA            THRU 9998-EXIT
     END-IF.

 2499-EXIT.
     EXIT.

 2500-SUCCESS-SEND.

*  SEND SUCCESS MESSAGE
```

```
     INITIALIZE FDIMP21O.
     MOVE 'MAIL SENT'                     TO MESGI.
     PERFORM 9998-SNDDATA                 THRU 9998-EXIT.

25ØØ-EXIT.
     EXIT.

9998-SNDDATA.

     EXEC CICS
          SEND MAP('FDIMP21')
               MAPSET('FDIMP21')
               FROM(FDIMP21O)
               DATAONLY
               FREEKB
     END-EXEC.
     EXEC CICS
          RETURN
          TRANSID('FIØ7')
          COMMAREA(WS-COMMAREA)
          LENGTH(LENGTH OF WS-COMMAREA)
     END-EXEC.

9998-EXIT.
     EXIT.

9999-END-SESSION.

     EXEC CICS SEND
          FROM(WS-MESSAGE)
          LENGTH(LENGTH OF WS-MESSAGE)
     END-EXEC.
     EXEC CICS
          RETURN
     END-EXEC.

9999-EXIT.
     EXIT.

```

FDIMP21PHY

```
         PRINT ON,NOGEN
FDIMP21  DFHMSD TYPE=MAP,LANG=COBOL,MODE=INOUT,STORAGE=AUTO,SUFFIX=
FDIMP21  DFHMDI SIZE=(24,8Ø),COLUMN=1,LINE=1,DATA=FIELD,TIOAPFX=YES,   *
               OBFMT=NO
         DFHMDF POS=(1,1),LENGTH=1,ATTRB=(PROT,BRT)
         DFHMDF POS=(2,15),LENGTH=4,INITIAL='CICS',ATTRB=(PROT,BRT)
         DFHMDF POS=(2,2Ø),LENGTH=6,INITIAL='SOCKET',ATTRB=(PROT,BRT)
         DFHMDF POS=(2,27),LENGTH=9,INITIAL='INTERFACE',ATTRB=(PROT,BRT*
               )
         DFHMDF POS=(2,37),LENGTH=5,INITIAL='EMAIL',ATTRB=(PROT,BRT)
         DFHMDF POS=(2,43),LENGTH=8,INITIAL='FACILITY',ATTRB=(PROT,BRT)
         DFHMDF POS=(3,15),LENGTH=36,                                   *
               INITIAL='====================================',         *
               ATTRB=(PROT,BRT)
         DFHMDF POS=(5,8),LENGTH=5,INITIAL='ENTER',ATTRB=(PROT,BRT)
         DFHMDF POS=(5,14),LENGTH=3,INITIAL='THE',ATTRB=(PROT,BRT)
         DFHMDF POS=(5,18),LENGTH=5,INITIAL='EMAIL',ATTRB=(PROT,BRT)
         DFHMDF POS=(5,24),LENGTH=2,INITIAL='ID',ATTRB=(PROT,BRT)
         DFHMDF POS=(5,27),LENGTH=2,INITIAL=':-',ATTRB=(PROT,BRT)
* MAILID                          MAILID
MAILID   DFHMDF POS=(5,3Ø),LENGTH=26,ATTRB=(UNPROT,NORM)
         DFHMDF POS=(5,57),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMDF POS=(7,8),LENGTH=5,INITIAL='ENTER',ATTRB=(PROT,BRT)
         DFHMDF POS=(7,14),LENGTH=3,INITIAL='THE',ATTRB=(PROT,BRT)
         DFHMDF POS=(7,18),LENGTH=5,INITIAL='EMAIL',ATTRB=(PROT,BRT)
         DFHMDF POS=(7,24),LENGTH=32,                                   *
               INITIAL='MESSAGE(LIMITED TO 6Ø CHARACTERS',ATTRB=(PROT,B*
               RT)
         DFHMDF POS=(7,57),LENGTH=2,INITIAL=':-',ATTRB=(PROT,BRT)
* MAILMG                          MAILMG
MAILMG   DFHMDF POS=(9,8),LENGTH=59,ATTRB=(UNPROT,NORM)
         DFHMDF POS=(9,68),LENGTH=1,ATTRB=(PROT,NORM)
* MESG                            MESG
MESG     DFHMDF POS=(15,5),LENGTH=64,ATTRB=(PROT,BRT)
         DFHMDF POS=(15,7Ø),LENGTH=1,ATTRB=(PROT,NORM)
         DFHMSD TYPE=FINAL
         END
```

FDIMP21SYM

```
       Ø1   FDIMP21I.
           Ø2 FILLER    PIC X(12).
           Ø2 MAILIDL   COMP PIC S9(4).
           Ø2 MAILIDF   PIC X.
           Ø2 MAILIDI   PIC X(26).
           Ø2 MAILMGL   COMP PIC S9(4).
           Ø2 MAILMGF   PIC X.
           Ø2 MAILMGI   PIC X(59).
           Ø2 MESGL     COMP PIC S9(4).
           Ø2 MESGF     PIC X.
           Ø2 MESGI     PIC X(64).
       Ø1   FDIMP21O REDEFINES FDIMP21I.
           Ø2 FILLER    PIC X(12).
           Ø2 FILLER    PIC X(2).
           Ø2 MAILIDA   PIC X.
           Ø2 MAILIDO   PIC X(26).
           Ø2 FILLER    PIC X(2).
           Ø2 MAILMGA   PIC X.
           Ø2 MAILMGO   PIC X(59).
           Ø2 FILLER    PIC X(2).
           Ø2 MESGA     PIC X.
           Ø2 MESGO     PIC X(64).
```

Manas Biswal
Associate
Cognizant Technology Solutions (USA)

CICSPlex/System Manager Report Writer – part 2

This month we conclude the code for a generalized CPSM report writer.

```
/*********** @REFRESH BEGIN SAYDD    2ØØ2/11/16 19:Ø3:46 *************/
/* SAYDD    - Print messages to the requested DD                      */
/*-------------------------------------------------------------------*/
/* MSGDD    - DDNAME to write messages to                             */
/* MSGLINES - number of blank lines to put before and after           */
/* MESSAGE  - Text to write to the MSGDD                              */
/*********************************************************************/
 saydd: module = 'SAYDD'
        if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
        parse arg sparms
        push trace() time('L') module 'From:' sigl 'Parms:' sparms
        call modtrace 'START' sigl
        parse arg msgdd msglines message
        if words(msgdd msglines message) < 3 then
           call rcexit 33 'Missing MSGDD or MSGLINES'
        if datatype(msglines) <> 'NUM' then
           call rcexit 34 'MSGLINES must be numeric'
/*********************************************************************/
/* If this is not background then bypass                              */
/*********************************************************************/
        if tsoenv <> 'BACK' then
           do
            pull tracelvl . module . sigl . sparms
            call modtrace 'STOP' sigl
            interpret 'trace' tracelvl
            return
           end
```

```
/*********************************************************************/
/* Confirm the MSGDD exists                                          */
/*********************************************************************/
        call ddcheck msgdd
/*********************************************************************/
/* If a number is provided, add that number of blank lines before    */
/* and after the message                                             */
/*********************************************************************/
        msgb = 1
        if msglines > Ø then
           do msgb=1 to msglines
              msgline.msgb = ' '
           end
        msgline.msgb = date() time() strip(message)
        if msglines > Ø then
           do msgt=1 to msglines
              msge = msgt + msgb
              msgline.msge = ' '
           end
/*********************************************************************/
/* Write the contents of the MSGLINE stem to the MSGDD               */
/*********************************************************************/
        call tsotrap "EXECIO * DISKW" msgdd "(STEM MSGLINE. FINIS"
        drop msgline. msgb msgt msge
        pull tracelvl . module . sigl . sparms
        call modtrace 'STOP' sigl
        interpret 'trace' tracelvl
        return
/*********** @REFRESH END   SAYDD    2ØØ2/11/16 19:Ø3:46 *************/
/*********** @REFRESH BEGIN JOBINFO  2ØØ2/Ø9/11 Ø1:12:59 *************/
/* JOBINFO  - Get job related data from control blocks               */
/*-------------------------------------------------------------------*/
/* ITEM     - Optional item number desired, default is all           */
/*********************************************************************/
 jobinfo: module = 'JOBINFO'
          if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg item
/*********************************************************************/
/* Chase control blocks                                              */
/*********************************************************************/
          tcb      = ptr(54Ø)
          ascb     = ptr(548)
          tiot     = ptr(tcb+12)
          jscb     = ptr(tcb+18Ø)
          ssib     = ptr(jscb+316)
          asid     = c2d(stg(ascb+36,2))
          jobtype  = stg(ssib+12,3)
          jobnum   = strip(stg(ssib+15,5),'L',Ø)
          stepname = stg(tiot+8,8)
          procstep = stg(tiot+16,8)
          program  = stg(jscb+36Ø,8)
          jobdata  = jobtype jobnum stepname procstep program asid
/*********************************************************************/
/* Return job data                                                   */
/*********************************************************************/
          if item <> '' & (datatype(item,'W') = 1) then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return word(jobdata,item)
             end
          else
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return jobdata
             end
/*********** @REFRESH END   JOBINFO  2ØØ2/Ø9/11 Ø1:12:59 *************/
/*********** @REFRESH BEGIN PTR      2ØØ2/Ø7/13 15:45:36 *************/
/* PTR      - Pointer to a storage location                          */
/*-------------------------------------------------------------------*/
/* ARG(1)   - Storage Address                                        */
/*********************************************************************/
 ptr: return c2d(storage(d2x(arg(1)),4))
/*********** @REFRESH END   PTR      2ØØ2/Ø7/13 15:45:36 *************/
/*********** @REFRESH BEGIN STG      2ØØ2/Ø7/13 15:49:12 *************/
/* STG      - Return the data from a storage location                */
/*-------------------------------------------------------------------*/
/* ARG(1)   - Location                                               */
/* ARG(2)   - Length                                                 */
/*********************************************************************/
 stg: return storage(d2x(arg(1)),arg(2))
/*********** @REFRESH END   STG      2ØØ2/Ø7/13 15:49:12 *************/
/*********** @REFRESH BEGIN MODTRACE 2ØØ2/Ø9/11 Ø1:46:24 *************/
/* MODTRACE - Module Trace                                           */
/*-------------------------------------------------------------------*/
/* TRACETYP - Type of trace entry                                    */
/* SIGLINE  - The line number called from                            */
/*********************************************************************/
 modtrace: if modtrace = 'NO' then return
           arg tracetyp sigline
           tracetyp = left(tracetyp,5)
           sigline = left(sigline,5)
/*********************************************************************/
/* Adjust MODSPACE for START                                         */
```

```
/*********************************************************************/
           if tracetyp = 'START' then
              modspace = substr(modspace,1,length(modspace)+1)
/*********************************************************************/
/* Set the trace entry                                               */
/*********************************************************************/
           traceline = modspace time('L') tracetyp module sigline sparms
/*********************************************************************/
/* Adjust MODSPACE for STOP                                          */
/*********************************************************************/
           if tracetyp = 'STOP' then
              modspace = substr(modspace,1,length(modspace)-1)
/*********************************************************************/
/* Determine where to write the traceline                            */
/*********************************************************************/
           if ispfenv = 'YES' then
/*********************************************************************/
/* Write to the ISPF Log, do not use ISPWRAP here                    */
/*********************************************************************/
              do
               zedlmsg = traceline
               address ISPEXEC "LOG MSG(ISRZ000)"
              end
           else
              say traceline
/*********************************************************************/
/* SAY to SYSTSPRT                                                   */
/*********************************************************************/
           return
/*********** @REFRESH END   MODTRACE 2002/09/11 01:46:24 *************/
/*********** @REFRESH BEGIN CPSMCMAS 2002/09/11 01:05:54 *************/
/* CPSMCMAS - Get CMAS name                                          */
/*-------------------------------------------------------------------*/
/* N/A      - None                                                   */
/*********************************************************************/
 cpsmcmas: module = 'CPSMCMAS'
           if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           call modtrace 'START' sigl
           cmas = 'C'||mvsvar('SYSCLONE')||'XCMAS'
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return cmas
/*********** @REFRESH END   CPSMCMAS 2002/09/11 01:05:54 *************/
/*********** @REFRESH BEGIN CPSMERR  2002/09/11 01:06:31 *************/
/* CPSMERR  - Format a CPSM error message for RCEXIT                 */
/*------------------------------------------------------------------*/
/* CPSMRC   - CPSM Return Code                                       */
/* MODULE   - CPSM subroutine issuing the error                     */
/* VERB     - CPSM API Verb issuing the error                       */
/* REASON   - CPSM Reason Code                                      */
/* RESPONSE - CPSM Response Code                                    */
/********************************************************************/
 cpsmerr: module = 'CPSMERR'
         if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         arg cpsmrc module verb reason resource response
         if response <> eyuresp('OK') then
            do
             msgprefix = module verb
             msg = eyureas(reason) resource eyuresp(response)
             MAXRC = cpsmrc
             call rcexit MAXRC msgprefix msg
            end
         else
            do
             pull tracelvl . module . sigl . sparms
             call modtrace 'STOP' sigl
             interpret 'trace' tracelvl
             return
            end
/*********** @REFRESH END   CPSMERR  2ØØ2/Ø9/11 Ø1:Ø6:31 *************/
/*********** @REFRESH BEGIN CPSMINIT 2ØØ2/Ø9/11 Ø1:Ø7:27 *************/
/* CPSMINIT - Initialize a CPSM session                             */
/*------------------------------------------------------------------*/
/* CMAS     - CPSM CMAS                                             */
/********************************************************************/
 cpsminit: module = 'CPSMINIT'
          if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg cmas
          if cmas = '' then cmas = cpsmcmas()
/********************************************************************/
          cpsm_ver = 'Ø14Ø'     /* Change as CPSM Version changes   */
/********************************************************************/
/* Set TRC=9999 for shutdown check to insure a CPSMTERM is run      */
/********************************************************************/
          TRC = 9999
/********************************************************************/
/* Initialize the CPSM API                                          */
/********************************************************************/
          call rcexit eyuinit() 'Error initializing the CPSM REXX API'
/********************************************************************/
/* Connect to a CMAS                                                */
```

```
/*********************************************************************/
           CRC = eyuapi("CONNECT",
                        "CONTEXT("cmas")",
                        "SCOPE("cmas")",
                        "VERSION("cpsm_ver")",
                        "THREAD(CPSM_THREAD)",
                        "RESPONSE(RESPONSE)",
                        "REASON(REASON)")
/*********************************************************************/
/* Error processing                                                  */
/*********************************************************************/
           cmasmsg = cmas '(Version' cpsm_ver')'
           call rcexit CRC 'Error connecting to' cmasmsg
           call cpsmerr 1Ø 'CPSMINIT CONNECT' reason cmas response
           if cpsm_thread = Ø then call rcexit 1Ø 'No valid CPSM Thread'
/*********************************************************************/
/* Connected OK                                                      */
/*********************************************************************/
           connmsg = 'Connected to' cmasmsg 'on' mvsvar('SYSNAME')
           call saydd msgdd Ø connmsg
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return cpsm_thread
/*********** @REFRESH END   CPSMINIT 2ØØ2/Ø9/11 Ø1:Ø7:27 *************/
/*********** @REFRESH BEGIN CPSMOLEN 2ØØ2/Ø9/11 Ø1:Ø7:44 *************/
/* CPSMOLEN - Get a CPSM Objects Length                              */
/*-------------------------------------------------------------------*/
/* THREAD   - CPSM Thread                                            */
/* OBJECT   - CPSM Object                                            */
/* DETAIL   - CPSM Details, set to any value for debugging details   */
/*********************************************************************/
 cpsmolen: module = 'CPSMOLEN'
           if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           call modtrace 'START' sigl
           arg cpsm_thread object details .
           if cpsm_thread = '' then call rcexit 31 'CPSM Thread missing'
           if object = '' then call rcexit 32 'CPSM Object missing'
/*********************************************************************/
/* Get a CPSM Object                                                 */
/*********************************************************************/
           ORC = eyuapi("GETDEF",
                        "OBJECT(METADESC)",
                        "RESOURCE("object")",
                        "COUNT(GETDEF_COUNT)",
                        "RESULT(GETDEF_RESULT)",
                        "THREAD(CPSM_THREAD)",
                        "RESPONSE(RESPONSE)",
                        "REASON(REASON)")
/*********************************************************************/
/* Error processing                                                  */
/*********************************************************************/
           call rcexit ORC 'GETDEF failed for' object
           call cpsmerr 33 'CPSMOLEN GETDEF' reason object response
/*********************************************************************/
/* Print the detail is details is non blank                          */
/*********************************************************************/
           if details <> '' then
              do
               call msg object 'detail requested'
               call msg getdef_count 'attributes found'
              end
           object_len = 55
           metadesc_len = 24
/*********************************************************************/
/* Loop through the results table                                    */
/*********************************************************************/
           do i=1 to getdef_count
/*********************************************************************/
/* Fetch the results                                                 */
/*********************************************************************/
              ORC = eyuapi("FETCH INTO(GETDEF_ROW)",
                           "LENGTH(METADESC_LEN)",
                           "RESULT(GETDEF_RESULT)",
                           "THREAD(CPSM_THREAD)",
                           "RESPONSE(RESPONSE)",
                           "REASON(REASON)")
/*********************************************************************/
/* Error processing                                                  */
/*********************************************************************/
              call rcexit ORC 'FETCH failed for' object
              call cpsmerr 33 'CPSMOLEN FETCH' reason object response
/*********************************************************************/
/* Convert and calculate the total length of the object record       */
/*********************************************************************/
              name = substr(getdef_row.1,1,12)
              len = x2d(c2x(substr(getdef_row.1,13,2)))
              object_len = object_len + len
/*********************************************************************/
/* Print the detail if details is non-blank                          */
/*********************************************************************/
              if details <> '' then call msg name 'Length='len
           end
/*********************************************************************/
/* Print the detail is details is non blank                          */
/*********************************************************************/
           if details <> '' then call msg object 'Length is' object_len
           if object_len = Ø then call rcexit 35 'Invalid Object Length'
```

```
/*********************************************************************/
/* Return the Object Length                                          */
/*********************************************************************/
           call saydd msgdd Ø 'GETDEF on' object 'Length is' object_len
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return object_len
/*********** @REFRESH END   CPSMOLEN 2ØØ2/Ø9/11 Ø1:Ø7:44 *************/
/*********** @REFRESH BEGIN CPSMGET  2ØØ2/Ø9/11 Ø1:Ø6:53 *************/
/* CPSMGET  - Get a CPSM Result Set                                  */
/*------------------------------------------------------------------*/
/* THREAD   - CPSM Thread                                            */
/* CONTEXT  - CPSM Context                                           */
/* SCOPE    - CPSM Scope                                             */
/* OBJECT   - CPSM Object                                            */
/* FILTER   - CPSM Filter                                            */
/*********************************************************************/
 cpsmget: module = 'CPSMGET'
          if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg cpsm_thread context scope object filter
          if cpsm_thread = '' then call rcexit 41 'CPSM Thread missing'
          if context = '' then call rcexit 42 'CPSM Context is missing'
          if scope = '' then call rcexit 43 'CPSM Scope is missing'
          if object = '' then call rcexit 44 'CPSM Object is missing'
/*********************************************************************/
/* Determine whether filter processing is required                   */
/*********************************************************************/
          if filter = '' then
             do
              GRC = eyuapi("GET",
                           "OBJECT("object")",
                           "CONTEXT("context")",
                           "SCOPE("scope")",
                           "COUNT(GET_COUNT)",
                           "RESULT(GET_RESULT)",
                           "THREAD(CPSM_THREAD)",
                           "RESPONSE(RESPONSE)",
                           "REASON(REASON)")
             end
          else
             do
/*********************************************************************/
/* Get the CPSM resource table with a filter                         */
/*********************************************************************/
              filter = filter'.'
              call saydd msgdd Ø 'FILTER' filter 'used'
              filter_len = length(filter)
              GRC = eyuapi("GET",
                           "OBJECT("object")",
                           "CONTEXT("context")",
                           "SCOPE("scope")",
                           "CRITERIA(FILTER)",
                           "LENGTH("filter_len")",
                           "COUNT(GET_COUNT)",
                           "RESULT(GET_RESULT)",
                           "THREAD(CPSM_THREAD)",
                           "RESPONSE(RESPONSE)",
                           "REASON(REASON)")
             end
/*********************************************************************/
/* If NODATA is found, continue                                      */
/*********************************************************************/
             if eyuresp(response) = 'NODATA' then
                nop
/*********************************************************************/
/* Error processing                                                  */
/*********************************************************************/
             else
                do
                 call rcexit GRC 'GET failed for' object
                 call cpsmerr 45 'CPSMGET GET' reason object response
                end
/*********************************************************************/
/* Exit with the RESULT ID and count                                 */
/*********************************************************************/
             if get_result = Ø then call rcexit 46 object 'count=Ø'
             call saydd msgdd Ø 'GET completed' get_count 'rows'
             pull tracelvl . module . sigl . sparms
             call modtrace 'STOP' sigl
             interpret 'trace' tracelvl
             return get_result get_count
/*********** @REFRESH END   CPSMGET  2ØØ2/Ø9/11 Ø1:Ø6:53 *************/
/*********** @REFRESH BEGIN CPSMGRP  2ØØ2/Ø9/11 Ø1:Ø7:Ø8 *************/
/* CPSMGRP  - Group a CPSM Results Set                               */
/*-------------------------------------------------------------------*/
/* THREAD   - CPSM Thread                                            */
/* GROUP    - CPSM GROUP BY Table Attribute                          */
/* FROMRES  - CPSM Source Results Set Handle                         */
/* SUMOPT   - CPSM GROUP Summary Options                             */
/*********************************************************************/
 cpsmgrp: module = 'CPSMGRP'
          if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
          parse arg sparms
          push trace() time('L') module 'From:' sigl 'Parms:' sparms
          call modtrace 'START' sigl
          arg cpsm_thread group fromres sumopt
          if cpsm_thread = '' then call rcexit 51 'CPSM Thread missing'
          if group = '' then call rcexit 52 'CPSM GROUP BY is missing'
          if fromres = '' then call rcexit 53 'CPSM FROMRES missing'
/**********************************************************************/
/* If a SUMOPT is required                                            */
/**********************************************************************/
          if sumopt = '' then
             do
              groupmsg = 'GROUP BY on' group 'and NO summary options'
              call saydd msgdd Ø groupmsg
              SRC = eyuapi("GROUP",
                           "BY("group")",
                           "FROM(FROMRES)",
                           "TO(GRP_RESULT)",
                           "COUNT(GRP_COUNT)",
                           "THREAD(CPSM_THREAD)",
                           "RESPONSE(RESPONSE)",
                           "REASON(REASON)")
             end
          else
             do
              sumopt = sumopt'.'
              sumlen = length(sumopt)
              groupmsg = 'GROUP BY on' group 'summary options' sumopt
              call saydd msgdd Ø groupmsg
              SRC = eyuapi("GROUP",
                           "BY("group")",
                           "FROM(FROMRES)",
                           "TO(GRP_RESULT)",
                           "COUNT(GRP_COUNT)",
                           "SUMOPT(SUMOPT)",
                           "LENGTH("sumlen")",
                           "THREAD(CPSM_THREAD)",
                           "RESPONSE(RESPONSE)",
                           "REASON(REASON)")
             end
/**********************************************************************/
/* Error processing                                                   */
/**********************************************************************/
          call rcexit SRC 'GROUP failed for' group
          call cpsmerr 54 'CPSMGRP GROUP' reason object response
/**********************************************************************/
/* Exit with the RESULT ID and count                                  */
/**********************************************************************/
          if grp_result = Ø then call rcexit 55 object 'count=Ø'
          call saydd msgdd Ø 'GROUP completed' grp_count 'rows'
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return grp_result grp_count
```

```
/*********** @REFRESH END   CPSMGRP  2ØØ2/Ø9/11 Ø1:Ø7:Ø8 *************/
/*********** @REFRESH BEGIN CPSMTERM 2ØØ2/Ø9/11 Ø1:Ø8:1Ø *************/
/* CPSMTERM - Terminate a CPSM session                               */
/*-------------------------------------------------------------------*/
/* CMAS     - CPSM CMAS                                              */
/*********************************************************************/
 cpsmterm: module = 'CPSMTERM'
           if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           call modtrace 'START' sigl
           arg cmas
           if cmas = '' then cmas = cpsmcmas()
           TRC = eyuapi("TERMINATE",
                        "RESPONSE(RESPONSE)",
                        "REASON(REASON)")
           call rcexit TRC 'CPSM Terminate error'
/*********************************************************************/
/* Free the CPSM function package                                    */
/*********************************************************************/
           call rcexit eyuterm() 'Error terminating the CPSM REXX API'
           termmsg = 'Unconnected from' cmasmsg 'on' mvsvar('SYSNAME')
           call saydd msgdd Ø termmsg
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return TRC
/*********** @REFRESH END   CPSMTERM 2ØØ2/Ø9/11 Ø1:Ø8:1Ø *************/
```

Robert Zenuk
Systems Programmer (USA) © Xephon 2003

CICS questions and answers

Q We would like to restrict the maximum users for a TOR because of AOR failures. Is there a way to do this in CICS?

A A good place would be in your Auto-Install Terminal (AITM) exit program – at log-on the program can reject the log-on if, for example 250 users are already logged on and two AORs aren't available. Perhaps a good solution would be to make your AITM check a TSq for max-users, then set max-users from other programs when certain situations arise. For example in the ZNEP, when an AOR fails, decrement 200 users from the max-users limit TSq, and the reverse when the AOR is restored.

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.

CICS news

GT Software has announced BMS/TS 8.0, formerly BMS/GT, which produces, edits, and maintains 3270 green screens for mainframe-based applications. This enhanced maintenance and development tool produces the basic fields and code required for upgrading individual screens.

BMS/TS allows 3270 Bridge enablement, providing a transformation of existing BMS screens to HTML and generating HTML pages directly to CICS or Transaction Server BMS maps.

Type 1 templates generate an HTML page that has all the fields visible in the HTML, users can open the HTML page in any Web development tool, view all the controls and components on the page, and edit each control and component. It is said to be easy to customize the screens to any preference. Type 2 (CICS managed) templates are not customizable by the user, rather their look and feel is dictated by CICS.

The import facility has been enhanced to pull in maps built with other screen generator tools, such as IBM's SDF and SDFII.

For further information contact:
GT Software, 1314 Spring Street NW Atlanta, Georgia 30309-2810, USA.
Tel: (404) 253 1300.
URL: http://www.gtsoftware.com/products/bmsTS/.

* * *

HostBridge Technology is making available, at no cost, a collection of sample programs that it has developed to make it easy for a CICS program to send an outbound TCP/IP or HTTP request.

These programs can serve as sample code to those interested in writing their own CICS socket I/O programs or adding socket support within their own programs. For example, these programs will allow a CICS program to invoke a Java Server Page (JSP), Active Server Page (ASP), or other CGI program via an HTTP GET request. Whatever the JSP/ASP/program returns in response to the GET request will be returned to the CICS program.

These programs do not require HostBridge, but they were originally written for one of its customers.

For further information contact:
HostBridge Technology, 1414 S Sangre Rd, Stillwater, OK 74074, USA.
Tel: (866) 965 2427.
URL: http://www.hostbridge.com/downloads.

* * *

MacKinney Systems has announced CICS/SignOn 1.3 and JSF 4.0.

CICS/SignOn 1.3 adds new features, including 17 API functions and two batch cross-reference reports. A new 150-byte user area in the user profile is available for storing additional information and there's now a password history area, which stores up to six passwords to prevent re-use.

Job and SysLog Facility (JSF) 4.0 is now available. JSF archives JES2 reports, JCL, and syslogs to disk and eventually to tape, based on MSGCLASS or destination.

For further information contact:
MacKinney Systems, 2740 S Glenstone Ave, Suite 103, Springfield, MO 65804-3737 USA
Tel: (417) 882 7569.
URL: http://www.mackinney.com/products/cics.htm.