August 2003

In this issue

3  ‘Killing’ CICS tasks with CICS Performance Monitor for z/OS
7  Displaying task information
14  Using socket programs to send e-mail from CICS
38  CICSPlex/System Manager Report Writer – part 2
48  CICS questions and answers
50  CICS news

© Xephon plc 2003
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

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

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

© Xephon plc 2003. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
‘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.

<table>
<thead>
<tr>
<th>Class</th>
<th>CEMT</th>
<th>CPSM, CICS PM</th>
<th>Third-party tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes – normal stuff</td>
</tr>
<tr>
<td>B</td>
<td>No</td>
<td>Yes</td>
<td>Yes – eg region at maxtask</td>
</tr>
<tr>
<td>C</td>
<td>No</td>
<td>No</td>
<td>Yes – unable to dispatch on QR TCB</td>
</tr>
<tr>
<td>D</td>
<td>No</td>
<td>No</td>
<td>No – the rest</td>
</tr>
</tbody>
</table>

*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:

<table>
<thead>
<tr>
<th>Tasknum</th>
<th>Tran</th>
<th>Userid</th>
<th>Term</th>
<th>Type</th>
<th>Pri</th>
<th>Status</th>
<th>Susptype</th>
<th>Suspval</th>
<th>Susptime</th>
<th>Sc</th>
<th>Tcl</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000023</td>
<td>JNL2</td>
<td>STCCICS</td>
<td>Task</td>
<td>255</td>
<td>Suspend</td>
<td>0000004</td>
<td>S</td>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000045</td>
<td>OMEG</td>
<td>STCCICS</td>
<td>Task</td>
<td>255</td>
<td>Suspend USERWAIT SRVWORK</td>
<td>00023156</td>
<td>S</td>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000046</td>
<td>OMEG</td>
<td>STCCICS</td>
<td>Task</td>
<td>255</td>
<td>Suspend USERWAIT SR2WORK</td>
<td>00023156</td>
<td>S</td>
<td>01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003364</td>
<td>P8EE</td>
<td>ECCSTCV</td>
<td>IX76</td>
<td>Term</td>
<td>001</td>
<td>Running</td>
<td>00000002</td>
<td>TP</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003365</td>
<td>F5RE</td>
<td>ECCSTAW</td>
<td>I211</td>
<td>Term</td>
<td>001</td>
<td>Running</td>
<td>00000002</td>
<td>TP</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003366</td>
<td>F9AC</td>
<td>ECCSDEQ</td>
<td>I021</td>
<td>Term</td>
<td>001</td>
<td>Running</td>
<td>00000001</td>
<td>TP</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003367</td>
<td>FICA</td>
<td>ECCSIUG</td>
<td>I129</td>
<td>Term</td>
<td>001</td>
<td>Running</td>
<td>00000001</td>
<td>TP</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003368</td>
<td>VTAS</td>
<td>ECICLP1</td>
<td>IX32</td>
<td>Term</td>
<td>001</td>
<td>Running</td>
<td>00000001</td>
<td>TP</td>
<td>01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 displayed.

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 tclog. 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 +0.
77 Z            PIC S9(4) COMP VALUE +0.
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 0.
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.  <br>************************************************ 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 CI-CSNL COMP PIC S9(4).
Ø4 CI-CSNF PIC X.
Ø4 CI-CSNI PIC X(8).
Ø4 DDATEL COMP PIC S9(4).
Ø4 DDATEF PIC X.
Ø4 DDATEI PIC X(10).
Ø4 DTIMEL COMP PIC S9(4).
Ø4 DTIMEF PIC X.
Ø4 DTIMEI PIC X(8).
Ø4 SCREEN-LINES PIC X(1520).
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).
**PROCEDURE DIVISION.**

* FIRST-TIME-ONLY.
* ============*
  IF EIBCALEN = 0
    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.
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  
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 'Dispatch' 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.
*==================*
MOVE SPACES TO LINEI(Z).
*
INITIATE-SCREEN.
*================*
EXEC CICS ASSIGN APPLID (CICSNI)
EXEC CICS ASKTIME ABSTIME (ABSTIME)
EXEC CICS FORMATTIME
  ABSTIME (ABSTIME)
  DATE (DDATEI)
  DATESEP ('/')
  TIME (DTIMEI)
  TIMESEP (':')
END-EXEC.

* RECEI VE-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)
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 deallocated 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.

REFERENCES

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

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. FDIPG18.
AUTHOR. MANAS.

DATE-WRITTEN. 03/26/2003.

DATE-COMPILED. 03/26/2003.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

COPY DFHAID.

COPY FDI MP21.

WS-MESSAGE PIC (X(21)) VALUE 'SESSION COMPLETED....'.

WS-COMMAREA PIC X.

WS-ERROR.

ERROR-1 PIC (X(6)) VALUE 'ERROR: '

ERROR-2 PIC (X(20)) VALUE SPACES.

ERROR-3 PIC (X(11)) VALUE 'ERROR CODE: '.

ERROR-4 PIC (X(18)).

WS-NUM-ERROR PIC 9(18).

SOKET-FUNCTIONS.

ACCEPT PIC (X(16)) VALUE 'ACCEPT '

BIND PIC (X(16)) VALUE 'BIND '

CLOSE PIC (X(16)) VALUE 'CLOSE '

CONNECT PIC (X(16)) VALUE 'CONNECT '

FCNTL PIC (X(16)) VALUE 'FCNTL '

GETCLIENTID PIC (X(16)) VALUE 'GETCLIENTID '

GETHOSTBYADDR PIC (X(16)) VALUE 'GETHOSTBYADDR '

GETHOSTBYNAME PIC (X(16)) VALUE 'GETHOSTBYNAME '

GETHOSTID PIC (X(16)) VALUE 'GETHOSTID '

GETHOSTNAME PIC (X(16)) VALUE 'GETHOSTNAME '

GETPEERNAME PIC (X(16)) VALUE 'GETPEERNAME '

GETSOCKNAME PIC (X(16)) VALUE 'GETSOCKNAME '

GETSOCKOPT PIC (X(16)) VALUE 'GETSOCKOPT '

GIVESOCKET PIC (X(16)) VALUE 'GIVESOCKET '

INITAPI PIC (X(16)) VALUE 'INITAPI '

IOCTL PIC (X(16)) VALUE 'IOCTL '

LISTEN PIC (X(16)) VALUE 'LISTEN '

READ PIC (X(16)) VALUE 'READ '

RECV PIC (X(16)) VALUE 'RECV '

RECVFROM PIC (X(16)) VALUE 'RECVFROM '

SELECT PIC (X(16)) VALUE 'SELECT '

SEND PIC (X(16)) VALUE 'SEND '

SENDTO PIC (X(16)) VALUE 'SENDTO '

SETSOCKOPT PIC (X(16)) VALUE 'SETSOCKOPT '

SHUTDOWN PIC (X(16)) VALUE 'SHUTDOWN '

SOCKET PIC (X(16)) VALUE 'SOCKET '

TAKESOCKET PIC (X(16)) VALUE 'TAKESOCKET '

COPY DFHAID.

COPY FDI MP21.

WS-MESSAGE PIC (X(21)) VALUE 'SESSION COMPLETED....'.

WS-COMMAREA PIC X.

WS-ERROR.

ERROR-1 PIC (X(6)) VALUE 'ERROR: '

ERROR-2 PIC (X(20)) VALUE SPACES.

ERROR-3 PIC (X(11)) VALUE 'ERROR CODE: '.

ERROR-4 PIC (X(18)).

WS-NUM-ERROR PIC 9(18).

SOKET-FUNCTIONS.

ACCEPT PIC (X(16)) VALUE 'ACCEPT '

BIND PIC (X(16)) VALUE 'BIND '

CLOSE PIC (X(16)) VALUE 'CLOSE '

CONNECT PIC (X(16)) VALUE 'CONNECT '

FCNTL PIC (X(16)) VALUE 'FCNTL '

GETCLIENTID PIC (X(16)) VALUE 'GETCLIENTID '

GETHOSTBYADDR PIC (X(16)) VALUE 'GETHOSTBYADDR '

GETHOSTBYNAME PIC (X(16)) VALUE 'GETHOSTBYNAME '

GETHOSTID PIC (X(16)) VALUE 'GETHOSTID '

GETHOSTNAME PIC (X(16)) VALUE 'GETHOSTNAME '

GETPEERNAME PIC (X(16)) VALUE 'GETPEERNAME '

GETSOCKNAME PIC (X(16)) VALUE 'GETSOCKNAME '

GETSOCKOPT PIC (X(16)) VALUE 'GETSOCKOPT '

GIVESOCKET PIC (X(16)) VALUE 'GIVESOCKET '

INITAPI PIC (X(16)) VALUE 'INITAPI '

IOCTL PIC (X(16)) VALUE 'IOCTL '

LISTEN PIC (X(16)) VALUE 'LISTEN '

READ PIC (X(16)) VALUE 'READ '

RECV PIC (X(16)) VALUE 'RECV '

RECVFROM PIC (X(16)) VALUE 'RECVFROM '

SELECT PIC (X(16)) VALUE 'SELECT '

SEND PIC (X(16)) VALUE 'SEND '

SENDTO PIC (X(16)) VALUE 'SENDTO '

SETSOCKOPT PIC (X(16)) VALUE 'SETSOCKOPT '

SHUTDOWN PIC (X(16)) VALUE 'SHUTDOWN '

SOCKET PIC (X(16)) VALUE 'SOCKET '

TAKESOCKET PIC (X(16)) VALUE 'TAKESOCKET '.

000601 02 SOKET-TERMAPI PIC X(16) VALUE 'TERMAPI'.
000602 02 SOKET-WRITE PIC X(16) VALUE 'WRITE'.
000603
000604 01 SOCTYPE PIC 9(8) COMP VALUE 1.
000605 01 PROTO  PIC 9(8) COMP VALUE 0.
000606 01 SOCKID  PIC 9(8) COMP.
000607 01 ERRNO  PIC 9(8) COMP.
000608 01 RETCODE  PIC S9(8) COMP.
000609 01 AF-INET  PIC 9(8) COMP VALUE 2.
000610 01 RECV-FLAGS.
000611 02 NO-FLAGS  PIC 9(8) COMP VALUE 0.
000612 02 OOB  PIC 9(8) COMP VALUE 1.
000613 02 PEEK  PIC 9(8) COMP VALUE 2.
000614
000615 01 RECV-BYTE  PIC 9(8) COMP VALUE 1000.
000616 01 RECV-BUF  PIC X(1000).
000617
000620 01 SOCKADDR.
000621 05 FAMILY  PIC 9(4) COMP VALUE 0.
000622 05 PORTNO  PIC 9(4) COMP VALUE 0.
000623 05 IADDR  PIC X(4).
000624 05 RESERVED  PIC X(8) VALUE LOW-VALUES.
000625
000626 01 WRITE-TCPLENG  PIC 9(8) COMP.
000627 01 WRITE-TCPBUF  PIC X(100).
000628 01 TOEBCDICTOKEN  PIC X(16) VALUE 'TCPIPTOEBCDICXLT'.
000629 01 TOASCII-TOKEN  PIC X(16) VALUE 'TCPIPTOASCIIXLAT'.
000630 01 WS-CRLF  PIC X(2) VALUE 'OD15'.
000631 01 MAXSOC  PIC 9(4) COMP VALUE 20.
000632 01 INIT-IDENT.
000633 05 INIT-NAME  PIC X(8).
000634 05 INIT-ADRS  PIC X(8).
000635 01 INIT-CICSTSK  PIC X(8).
000636 01 MAXSNO  PIC 9(8) COMP VALUE 19.
000637 01 TEMP-HEX-IP  PIC 9(4) COMP.
000638 01 TEMP-HEX-IP-NUM REDEFINES TEMP-HEX-IP.
000639 02 TEMP-WASTE-IP  PIC X.
000640 02 TEMP-HEXIP-X2  PIC X.
000641 01 WS-SUBJECT-INFO  PIC X(44)
             VALUE 'SUBJECT: MAIL FROM CTS MAINFRAME SMTP SERVER'.
000642
000643
000644 LINKAGE SECTION.
000645 01 DFHCOMMAREA  PIC X.
000646*
000647 PROCEDURE DIVISION.
000648
000649 IF EIBCALEN = ZERO
000650 PERFORM 1000-FIRST-PASS THRU 1000-EXIT
000651 ELSE
000652 PERFORM 2000-NEXT-PASS THRU 2000-EXIT

000653  END-IF.
000654  EXEC CICS RETURN
000655  END-EXEC.
000656
000657  1000- FIRST-PASS.
000658
000659  EXEC CICS
000660      SEND MAP('FDI MP21')
000661      MAPSET('FDI MP21')
000662      MAPONLY
000663      ERASE
000664      FREEKB
000665      FRSET
000666  END-EXEC.
000667  EXEC CICS
000668      RETURN
000669      TRANSID('FI07')
000670      COMMAREA(W5-COMMAREA)
000671      LENGTH(LENGTH OF W5-COMMAREA)
000672  END-EXEC.
000673
000680  1000- EXIT.
000681  EXIT.
000682
000690  2000- NEXT-PASS.
000691
000692  IF EIBAID = DFHCLEAR
000693      PERFORM 9999-END-SESSION THRU 9999-EXIT
000694  END-IF.
000695
000696  EXEC CICS HANDLE CONDITION
000697      MAPFAIL(2100-MAPFAIL)
000698  END-EXEC.
000699  EXEC CICS
000700      RECEIVE MAP('FDI MP21')
000701      MAPSET('FDI MP21')
000702      INTO(FDI MP21I)
000703  END-EXEC.
000704
000705  EVALUATE EIBAID
000706      WHEN DFHCLEAR
000707      PERFORM 9999-END-SESSION THRU 9999-EXIT
000708      WHEN DFHENTER
000709      CONTINUE
000710      WHEN OTHER
000711      PERFORM 2200-INVKEY THRU 2200-EXIT
000712  END-EVALUATE.
000713
000714  PERFORM 2300-DATA-VALID THRU 2300-EXIT.
000715  PERFORM 2400-TCP-PROCESS THRU 2400-EXIT.
000992 PERFORM 2500-SUCCESS-SEND THRU 2500-EXIT.
000993
001000 2000-EXIT.
001100 EXIT.
001200
001300 2100-MAPFAIL.
001301
001310 MOVE 'NO DATA ENTERED..PLS ENTER RELEVANT DATA'
001320 TO MESGI.
001330 PERFORM 9998-SNDDATA THRU 9998-EXIT.
001340
001400 2100-EXIT.
001410 EXIT.
001500
001600 2200-INVKEY.
001610
001620 MOVE 'ONLY ENTER AND CLEAR KEYS ARE ACTVE'
001630 TO MESGI.
001640 PERFORM 9998-SNDDATA THRU 9998-EXIT.
001650
001700 2200-EXIT.
001710 EXIT.
001800
001801 2300-DATA-VALID.
001802
001803 IF MAILIDI = SPACES OR LOW-VALUES
001804 MOVE 'ENTER A VALID MAIL ID..' TO MESGI
001805 PERFORM 9998-SNDDATA THRU 9998-EXIT
001806 END-IF.
001807
001808 IF MAILMGI = SPACES OR LOW-VALUES
001809 MOVE 'ENTER A VALID MAIL MSG.'
001810 TO MESGI
001811 PERFORM 9998-SNDDATA THRU 9998-EXIT
001812 END-IF.
001813
001814 2300-EXIT.
001815 EXIT.
001816
001817 2400-TCP-PROCESS.
001818
001819 PERFORM 2405-INIT-ENVIRON THRU 2405-EXIT.
001820 PERFORM 2410-CREATE-SOCKET THRU 2410-EXIT.
001821 PERFORM 2420-TCP-CONNECT THRU 2420-EXIT.
001822 PERFORM 2430-TEST-SERVER THRU 2430-EXIT.
001823 PERFORM 2440-HELO-WRITE THRU 2440-EXIT.
001824 PERFORM 2450-MAIL-WRITE THRU 2450-EXIT.
001825 PERFORM 2460-RCPT-WRITE THRU 2460-EXIT.
001826 PERFORM 2470-DATA-WRITE THRU 2470-EXIT.
001827 PERFORM 2480-HEADER-WRITE THRU 2480-EXIT.
PERFORM 2490-MAILMSG-WRITE THRU 2490-EXIT.
PERFORM 2491-END-MAILMSG THRU 2491-EXIT.
PERFORM 2492-QUIT-SMTP THRU 2492-EXIT.
PERFORM 2499-CLOSE-SOCKET THRU 2499-EXIT.

EXIT.

INITIALIZE THE ENVIRONMENT

MOVE 0 TO ERRNO.
MOVE 0 TO RETCODE.
MOVE 'TCP/IP' TO INIT-NAME.
EXEC CICS
ASSIGN APPLID(INIT-ADRS)
END-EXEC.

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

IF RETCODE < 0
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.

CREATE A SOCKET

MOVE 0 TO ERRNO.
MOVE 0 TO RETCODE.

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

IF RETCODE < 0
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

OPEN A TCP CONNECTION TO THE MAIL SERVER

MOVE AF-I-NET 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 < Ø
MOVE 'OPEN CONN' TO WS-ERROR-2
MOVE ERRNO 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 Ø TO ERRNO.
MOVE Ø 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' TO WS-ERROR-2
Ø01989  MOVE ERRNO            TO WS- NUM-ERROR
Ø01990  MOVE WS- NUM-ERROR    TO WS- ERROR-4
Ø01991  MOVE WS- ERROR        TO MESGI
Ø01992  PERFORM 9998- SNDDATA THRU 9998- EXIT
Ø01993  END-IF.
Ø01994
Ø01995  CALL 'EZACIC05' USING TOEBCDI-C-TOKEN RECV-BUF RETCODE.
Ø01996
Ø01999  IF RECV-BUF(1:3) NOT = '220'
Ø02000  MOVE 'SOCKET REC V RESP' TO WS- ERROR-2
Ø02001  MOVE RECV-BUF          TO WS- ERROR-4
Ø02002  MOVE WS- ERROR         TO MESGI
Ø02003  PERFORM 9998- SNDDATA THRU 9998- EXIT
Ø02004  END-IF.
Ø02005
Ø02006  2430-EXIT.
Ø02007  EXIT.
Ø02008
Ø02009  2440-HELO-WRITE.
Ø02010
Ø02011* HELO SOCKET WRITE COMMAND
Ø02012  STRING 'HELO CPAC' WS-CRLF DELIMITED BY SIZE
Ø02013                                INTO WRITE-TCPBUF.
Ø02014  MOVE 11                       TO WRITE-TCPLENG.
Ø02015  CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
Ø02016                                           WRITE-TCPLENG.
Ø02017
Ø02018  MOVE 0                         TO ERRNO.
Ø02019  MOVE 0                         TO RETCODE.
Ø02020  CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPLENG
Ø02021                                           WRITE-TCPBUF ERRNO RETCODE.
Ø02022
Ø02023  IF RETCODE < 0
Ø02024  MOVE 'SOCKET WRITE HELO'       TO WS- ERROR-2
Ø02025  MOVE ERRNO                     TO WS- NUM-ERROR
Ø02026  MOVE WS- NUM-ERROR            TO WS- ERROR-4
Ø02027  MOVE WS- ERROR                TO MESGI
Ø02028  PERFORM 9998- SNDDATA         THRU 9998- EXIT
Ø02029  END-IF.
Ø02030
Ø02031* CHECK SMTP RESPONSE
Ø02036  MOVE 0                         TO ERRNO.
Ø02037  MOVE 0                         TO RETCODE.
Ø02038  MOVE 1000                      TO RECV-BYTE.
Ø02039
Ø02044  CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
Ø02045                                           RECV-BYTE RECV-BUF ERRNO RETCODE.
Ø02046
Ø02051  IF RETCODE < 0
Ø02052  MOVE 'SOCKET REC V 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 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 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 Ø TO ERRNO.
MOVE Ø TO RETCODE.
MOVE 1000 TO RECV-BYTE.
CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS RECV-BYTE RECV-BUF ERRNO RETCODE.
002107 IF RETCODE < Ø
002108 MOVE 'SOCKET RECV MAIL' TO WS-ERROR-2
002109 MOVE ERRNO TO WS-NUM-ERROR
002110 MOVE WS-NUM-ERROR TO WS-ERROR-4
002111 MOVE WS-ERROR TO MESGI
002112 PERFORM 9998-SNDDATA THRU 9998-EXIT
002113 END-IF.
002114 CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.
002115 IF RETCODE < Ø
002116 MOVE 'SOCKET RECV MAIL' TO WS-ERROR-2
002117 MOVE ERRNO TO WS-NUM-ERROR
002118 MOVE WS-NUM-ERROR TO WS-ERROR-4
002119 MOVE WS-ERROR TO MESGI
002120 PERFORM 9998-SNDDATA THRU 9998-EXIT
002121 END-IF.
002122 2450-EXIT.
002123 EXIT.
002124 2460-RCPT-WRITE.
002125 RCPT SOCKET WRITE COMMAND
002126 MOVE SPACES TO WRITE-TCPBUF.
002127 STRING 'RCPT TO: <MAILIDI>'
002128 WS-CRLF DELIMITED BY SIZE INTO WRITE-TCPBUF.
002129 MOVE 39 TO WRITE-TCPLENG.
002130 CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF WRITE-TCPLENG.
002131 MOVE Ø TO ERRNO.
002132 MOVE Ø TO RETCODE.
002133 MOVE 1000 TO RECV-BYTE.
002134 CALL 'EZASOKET' USING SOCKID WRITE-TCPBUF ERRNO RETCODE.
002135 IF RETCODE < Ø
002136 MOVE 'SOCKET WRITE RCPT' TO WS-ERROR-2
002137 MOVE ERRNO TO WS-NUM-ERROR
002138 MOVE WS-NUM-ERROR TO WS-ERROR-4
002139 MOVE WS-ERROR TO MESGI
002140 PERFORM 9998-SNDDATA THRU 9998-EXIT
002141 END-IF.
002142 MOVE Ø TO ERRNO.
002143 MOVE Ø TO RETCODE.
002144 CALL 'EZASOKET' USING SOCKID WRITE-TCPBUF ERRNO RETCODE.
002145 IF RETCODE < Ø
002146 MOVE 'SOCKET WRITE RCPT' TO WS-ERROR-2
002147 MOVE ERRNO TO WS-NUM-ERROR
002148 MOVE WS-NUM-ERROR TO WS-ERROR-4
002149 MOVE WS-ERROR TO MESGI
002150 PERFORM 9998-SNDDATA THRU 9998-EXIT
002151 END-IF.
002152 CHECK SMTP RESPONSE
002153 MOVE Ø TO ERRNO.
002154 MOVE Ø TO RETCODE.
002155 MOVE 1000 TO RECV-BYTE.
002156 CALL 'EZASOKET' USING SOCKID RCPT NO-FLAGS
002160   IF RETCODE < 0
002161     MOVE 'SOCKET RCPT' TO WS-ERROR-2
002162     MOVE ERRNO TO WS-ERROR-4
002163     MOVE WS-ERROR TO MESGI
002164     PERFORM 9998-SNDDATA THRU 9998-EXIT
002165   END-IF.
002166
002167   CALL 'EZACIC05' USING TOEBCDICTOKEN RECV-BUF RETCODE.
002168
002169   IF RECV-BUF(1:3) NOT = '250'
002170     MOVE 'RCPT SMTP' TO WS-ERROR-2
002171     MOVE RECV-BUF TO WS-ERROR-4
002172     MOVE WS-ERROR TO MESGI
002173     PERFORM 9998-SNDDATA THRU 9998-EXIT
002174   END-IF.
002175
002176 2460-EXIT.
002177 2470-DATA-WRITE.
002178
002179  *DATA SOCKET WRITE COMMAND
002180     MOVE SPACES TO WRITE-TCPBUF.
002181     STRING 'DATA' WS-CRLF DELIMITED BY SIZE INTO WRITE-TCPBUF.
002182     MOVE 6 TO WRITE-TCPLENG.
002183     CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPLENG WRITE-TCPBUF.
002184     MOVE 0 TO ERRNO.
002185     MOVE 0 TO RETCODE.
002186     CALL 'EZASOKET' USING SOKET-WRITE SOCKID WRITE-TCPBUF ERRNO RETCODE.
002187
002188   IF RETCODE < 0
002189     MOVE 'SOCKET WRITE DATA' TO WS-ERROR-2
002190     MOVE ERRNO TO WS-ERROR-4
002191     MOVE WS-ERROR TO MESGI
002192     PERFORM 9998-SNDDATA THRU 9998-EXIT
002193   END-IF.
002194
002195   *CHECK SMTP RESPONSE
002196     MOVE 0 TO ERRNO.
002197     MOVE 0 TO RETCODE.
002198     MOVE 1000 TO RECVBYTE.
CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
RECV-BYTE RECV-BUF ERRNO RETCODE.

IF RETCODE < Ø
  MOVE 'SOCKET RECV DATA' TO WS-ERROR-2
  MOVE ERNNO 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.

2480-HEADER-WRITE.

IF RETCODE < Ø
  MOVE 'SOCKET WRITE TO HDR' TO WS-ERROR-2
  MOVE ERNNO 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.

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.

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.

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.

MAILMSG SOCKET WRITE COMMAND
002310  MOVE SPACES TO WRITE-TCPBUF.
002311  STRING MAILMSG WS-CRLF
002312    DELIMITED BY SIZE
002313    INTO WRITE-TCPBUF.
002314  MOVE 61 TO WRITE-TCPLENG.
002315  CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
002316    WRITE-TCPLENG.
002317 002318  MOVE Ø TO ERRNO.
002319  MOVE Ø TO RETCODE.
002320  CALL 'EZASOKET' USING SOCKID WRITE-TCPLENG
002321    ERRNO RETCODE.
002322 002323  IF RETCODE < Ø
002324    MOVE 'SOCKET WRITE MAILMSG' TO WS-ERROR-2
002325    MOVE ERRNO TO WS-ERROR.
002326    MOVE WS-NUM-ERROR TO WS-ERROR.
002327    MOVE WS-ERROR TO MESGI
002328    PERFORM 9998-SNDDATA THRU 9998-EXIT
002329 002330  END-IF.
002331 002332 2490-EXIT.
002333 002334  EXIT.
002335 002336 2491-END-MAILMSG.
002337 002338  END MAIL MSG SOCKET WRITE COMMAND
002339 002340  MOVE SPACES TO WRITE-TCPBUF.
002341  STRING '.' WS-CRLF
002342    DELIMITED BY SIZE
002343    INTO WRITE-TCPBUF.
002344 002345  MOVE 3 TO WRITE-TCPLENG.
002346 002347  CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
002348    WRITE-TCPLENG.
002349 002350  MOVE Ø TO ERRNO.
002351 002352  MOVE Ø TO RETCODE.
002353 002354  CALL 'EZASOKET' USING SOCKID WRITE-TCPLENG
002355    ERRNO RETCODE.
002356 002357  IF RETCODE < Ø
002358    MOVE 'SOCKET WRITE ENDMAILMSG' TO WS-ERROR-2
002359    MOVE ERRNO TO WS-NUM-ERROR.
002360    MOVE WS-NUM-ERROR TO WS-ERROR.
002361    MOVE WS-ERROR TO MESGI
002362    PERFORM 9998-SNDDATA THRU 9998-EXIT
002363 002364  END-IF.
002365 002366 2490-EXIT.
002367 002368  EXIT.
002369 002370 2491-END-MAILMSG.
002371 002372  END MAIL MSG SOCKET WRITE COMMAND
002373  MOVE SPACES TO WRITE-TCPBUF.
002374  STRING '.' WS-CRLF
002375    DELIMITED BY SIZE
002376    INTO WRITE-TCPBUF.
002377 002378  MOVE 3 TO WRITE-TCPLENG.
002379 002380  CALL 'EZACIC04' USING TOASCII-TOKEN WRITE-TCPBUF
002381    WRITE-TCPLENG.
002382 002383  MOVE Ø TO ERRNO.
002384 002385  MOVE Ø TO RETCODE.
002386 002387  CALL 'EZASOKET' USING SOCKID WRITE-TCPLENG
002388    ERRNO RETCODE.
002389 002390  IF RETCODE < Ø
002391    MOVE 'SOCKET WRITE ENDMail MSG' TO WS-ERROR.
002392    MOVE ERRNO TO WS-ERROR.
002393    MOVE WS-NUM-ERROR TO WS-ERROR.
002394    MOVE WS-ERROR TO MESGI
002395    PERFORM 9998-SNDDATA THRU 9998-EXIT
002396 002397  END-IF.
002398 002399  CHECK SMTP RESPONSE
002400 002401  MOVE Ø 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 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 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.
IF RECV-BUF(1:3) NOT = '250'
  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.
EXIT.
002444* CHECK SMTP RESPONSE
002445   MOVE 0 TO ERRNO.
002446   MOVE 0 TO RETCODE.
002447   MOVE 1000 TO RECV-BYTE.
002448
002449   CALL 'EZASOKET' USING SOKET-RECV SOCKID NO-FLAGS
002450   RECV-BYTE RECV-BUF ERRNO RETCODE.
002451
002452     IF RETCODE < 0
002453       MOVE 'SOCKET RECV QUITSMTP' TO WS-ERROR-2
002454       MOVE ERRNO TO WS-ERROR-4
002455       MOVE WS-ERROR TO MESGI
002456       PERFORM 9998-SNDDATA THRU 9998-EXIT
002457     END-IF.
002458
002459   CALL 'EZACIC05' USING TOEBCDIC-TOKEN RECV-BUF RETCODE.
002460
002462     IF RECV-BUF(1:3) NOT = '221'
002463       MOVE 'QUITSMTP' TO WS-ERROR-2
002464       MOVE RECV-BUF TO WS-ERROR-4
002465       MOVE WS-ERROR TO MESGI
002466       PERFORM 9998-SNDDATA THRU 9998-EXIT
002467     END-IF.
002468
002469 2492-EXIT.
002470 EXIT.
002471
002472 2499-CLOSE-SOCKET.
002473
002474* SOCKET CLOSE COMMAND
002475   MOVE 0 TO ERRNO.
002476   MOVE 0 TO RETCODE.
002477
002478   CALL 'EZASOKET' USING SOKET-CLOSE SOCKID ERRNO RETCODE.
002479
002480     IF RETCODE < 0
002481       MOVE 'SOCKET CLOSE' TO WS-ERROR-2
002482       MOVE ERRNO TO WS-ERROR-4
002483       MOVE WS-ERROR TO MESGI
002484       PERFORM 9998-SNDDATA THRU 9998-EXIT
002485     END-IF.
002486
002487 2499-EXIT.
002488 EXIT.
002489
002490 2500-SUCCESS-SEND.
002491
002492 002493* SEND SUCCESS MESSAGE
2494      INITIALIZE FDIMP21O.
2495      MOVE 'MAIL SENT' TO MESGI.
2496      PERFORM 9998-SNDDATA THRU 9998-EXIT.
2497 2500-EXIT.
2500      EXIT.
2501 9998-SNDDATA.
2502 9998-SNDDATA.
2503      EXEC CICS
2504      SEND MAP('FDIMP21')
2505      MAPSET('FDIMP21')
2506      FROM(FDIMP21O)
2507      DATAONLY
2508      FREEKB
2509      END-EXEC.
2510      EXEC CICS
2511      RETURN
2512      TRANSID('FI07')
2513      COMMAREA(WS-COMMAREA)
2514      LENGTH(LENGTH OF WS-COMMAREA)
2515      END-EXEC.
2516 9998-EXIT.
2517      EXIT.
2518 9999-END-SESSION.
2519 9999-END-SESSION.
2520      EXEC CICS SEND
2521      FROM(WS-MESSAGE)
2522      LENGTH(LENGTH OF WS-MESSAGE)
2523      END-EXEC.
2524      RETURN
2525      END-EXEC.
2526      EXEC CICS
2527      RETURN
2528      END-EXEC.
2529 9999-EXIT.
2530      EXIT.
2540      EXIT.
2600

FDIMP21PHY
PRINT ON, NOGEN
FDIMP21 DFHMSD TYPE=MAP, LANG=COBOL, MODE=INOUT, STORAGE=AUTO, SUFFIX=
FDIMP21 DFHMDI SIZE=(24, 80), 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, 20), 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, 30), 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 60 CHARACTERS)', 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, 70), LENGTH=1, ATTRB=(PROT, NORM)
DFHMDSV TYPE=FINAL
END

FDIMP21SYM

01   FDIMP21I.
   02   FILLER    PIC X(12).
   02   MAILIDL   COMP PIC S9(4).
   02   MAILIDF   PIC X.
   02   MAILIDI   PIC X(26).
   02   MAILMGL   COMP PIC S9(4).
   02   MAILMGF   PIC X.
   02   MAILMGI   PIC X(59).
   02   MESGL     COMP PIC S9(4).
   02   MESGF     PIC X.
   02   MESGI     PIC X(64).
01   FDIMP21O REDINES FDIMP21I.
   02   FILLER    PIC X(12).
   02   FILLER    PIC X(2).
   02   MAILIDA   PIC X.
   02   MAILIDO   PIC X(26).
CICSPlex/System Manager Report Writer – part 2

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

```assembly
* *********** @REFRESH BEGIN SAYDD 2002/11/16 19:03:46 *********** */
* 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 tsoenv <> 'BACK' then do
        pull tracevl . module . sigl . sparms
        call modtrace 'STOP' sigl
        interpret 'trace' tracevl
        return
    end
```

Manas Biswal
Associate
Cognizant Technology Solutions (USA) © Xephon 2003
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 > 0 then
do msgb=1 to msglines
    msgline.msgb = ' '
end
msgline.msgb = date() time() strip(message)
if msglines > 0 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

JOBINFO - Get job related data from control blocks

jobinfo: module = 'JOBINFO'
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 arg item

Chase control blocks

tcb = ptr(540)
ascb = ptr(548)
tiot = ptr(tcb+12)
jscb = ptr(tcb+180)
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+360,8)
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
end

/*********** @REFRESH END   JOBINFO  2002/09/11 01:12:59 *************/
/*********** @REFRESH BEGIN PTR      2002/07/13 15:45:36 *************/
/* PTR      - Pointer to a storage location                            */
/***************************************************************************/
ptr: return c2d(storage(d2x(arg(1)),4))

/*********** @REFRESH END   PTR      2002/07/13 15:45:36 *************/
/*********** @REFRESH BEGIN STG      2002/07/13 15:49:12 *************/
/* STG      - Return the data from a storage location                  */
/***************************************************************************/
stg: return storage(d2x(arg(1)),arg(2))

/*********** @REFRESH END   STG      2002/07/13 15:49:12 *************/
/*********** @REFRESH BEGIN MODTRACE 2002/09/11 01:46:24 *************/
/* MODTRACE - Module Trace                                             */
/***************************************************************************/
modtrace: if modtrace = 'NO' then return
  arg tracetyp sigline
  tracetyp = left(tracetyp,5)
  sigline = left(sigline,5)
end

/*********** @REFRESH END   MODTRACE 2002/09/11 01:46:24 *************/
/* 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)
end

/*********** @REFRESH END   MODTRACE 2002/09/11 01:46:24 *************/
/* Adjust MODSPACE for START                                          */
/*
if tracetyp = 'START' then
    modspace = substr(modspace,1,length(modspace)+1)
endif

/* 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)
endif

/* 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(ISRZØØØ)"
    end
    else
        say traceline
    endif

/* 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

N/A - None

CPSMERR: module = 'CPSMERR'
if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
parse arg sparms
call modtrace 'START' sigl
call modtrace 'STOP' sigl
interpret 'trace' tracelvl
return cmas

@REFRESH END CPSMERR 2002/09/11 01:06:31

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 cpsmr module verb reason resource response
  if response <> eyuresp('OK') then
    do
      msgprefix = module verb
      msg = eyureas(reason) resource eyuresp(response)
      MAXRC = cpsmr
      call rcexit MAXRC msgprefix msg
    end
  else
    do
      pull tracelvl . module . sigl . sparms
      call modtrace 'STOP' sigl
      interpret 'trace' tracelvl
      return
    end
  end
/*====================================================================*/

/* CPSMINIT - Initialize a CPSM session                              */
/*====================================================================*/
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()
/*====================================================================*/
cpsmver = '0140'  /* 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 */
/*====================================================================*/
/* Error processing */

/* Connected OK */

/* CPSMOLEN - Get a CPSM Objects Length */

/* THREAD - CPSM Thread */
/* OBJECT - CPSM Object */
/* DETAIL - CPSM Details, set to any value for debugging details */

/* Get a CPSM Object */
"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

```c
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
```

CPSMGET - Get a CPSM Result Set

```c
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

```c
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

```c
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
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 2002/09/11 01:06:53
@REFRESH BEGIN CPSMGRP 2002/09/11 01:07:08

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

grouopmsg = 'GROUP BY on' group 'and NO summary options'
call saydd msgdd Ø grouopmsg
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
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@xophon.net.*
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.

* * *

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.