CICS update

227

October 2004

In this issue

A tc PUBLICATION

CICS Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs $270.00 in the USA and Canada; £175.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) 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.

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. 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.

Enhanced application program debugging support in CICS Transaction Server Version 2.3

ABSTRACT

CICS Transaction Server Version 2.3 ships enhanced support for the interactive debugging of application programs. The support assists with defining the circumstances under which a debugging session is to commence, and greatly assists in the setting up of debugging sessions for Java application programs and mixed-language applications. The ability to debug applications across multiple CICS regions is also enhanced.

BACKGROUND

Prior to CICS TS Version 2.3, the interactive debugging of application programs was effected either by the use of CICS-provided transactions (such as CEDF) or through the use of debugging tools (such as the IBM Debug Tool via the DTCN transaction). CEDF is limited to CICS and DB2 commands and so does not provide much assistance with source level debugging.

The DTCN transaction allows an end user to set up ‘debugging profiles’. These are specifications of circumstances that must be satisfied by a running instance of an application program before the Debug Tool will be invoked. The DTCN profiles do not persist across a CICS restart and cannot be shared across multiple CICS regions or by multiple users. A major difficulty is that DTCN profiles do not cater for Java programs at all. Also there is no Web interface to DTCN, only a 3270 interface.

In order for IBM Debug Tool to work properly with COBOL and PL/I programs, it was necessary to link an exit called CEEBXITA with the source.

Many of these difficulties concerning the set up of a debugging environment have been overcome by the solution implemented in CICS TS Version 2.3.

THE CICS TRANSACTION SERVER VERSION 2.3 SOLUTION

The CICS enhancements described here are completely separate from the operation of any debug tool. They are enhancements to the application program set-up and CICS set-up required in order to trigger debugging tools.

With CICS TS Version 2.3, the preparation of application programs for debugging has been simplified. It is no longer necessary to link the programs with CEEBXITA. If the TEST (ALL,SEPARATE) option is used when compiling COBOL or PL/I applications, the compiler will produce the side files needed by IBM Debug Tool (and equivalent vendor products) automatically.

CICS TS Version 2.3 ships with two new transactions, CADP and CIDP, as well as a new Web interface for debugging. The following discussion will explain the use of the new transactions and demonstrate some of the advantages of using the new support provided.

MAKE SURE CICS IS SET UP PROPERLY

An application programmer needing to test an application program must first ensure that CICS is set up for debugging. The IBM Debug Tool (or equivalent vendor product) needs to have been installed and some new CICS datasets need to have been defined. The datasets are used to store debugging profiles, explained below. There is sample JCL provided in DFHDEFDS to define the associated VSAM datasets. These may be defined as RLS, non-RLS, or remote. CICS also needs to have been started with a new SIT parameter, DEBUGTOOL=YES, specified. This parameter defaults to NO. The setting can be changed via CEMT using the SET SYSTEM DEBUG command.

CREATE DEBUGGING PROFILES

The first step in debugging is to decide what needs to be debugged and under what circumstances. For example, it might be necessary to debug only a transaction if it is started at a particular terminal or by a particular userid. Debugging profiles are used to specify the conditions under which a debug tool needs to be invoked. Debugging profiles may be defined either by using the new CADP transaction from a 3270 terminal or by using the Web interface. The CADP transaction is intended as a complete replacement for and improvement on DTCN. Details of the new interfaces to debugging profiles may be found easily in the CICS Infocenter, and so will not be described here.

Whichever interface is used, the same information needs to be specified. For a COBOL program, the debugging profile can specify the transaction ID, the program name, and the compile unit name. Together with this information, details of the user of the program need to be specified; in particular the applid, the userid, the termid, and the netname can be set. These four pieces of information are defaulted to the current user's details. Any or all of the parameters may be left blank, in which case the parameter is always considered to be a match during the checking of profiles at run time. Any of the parameters can also end with a wildcard character of asterisk, '*', which makes the profiles more or less generic.

For Java programs, there are different parameters that may be specified. The class and method may be set, whereas the terminal and netname cannot, since these are meaningless for a Java program.

One very useful thing that can be specified for a Java program, an Enterprise JavaBean, or a CORBA stateless object is the JVM profile to use. Prior to CICS TS Version 2.3, in order to debug these entities it was necessary to install alternative definitions for the program and transaction so that the program would run in a debuggable JVM. There was no mechanism for dynamically switching to a different JVM profile at run-time. By using the JVM profile input parameter when defining a debugging profile, it is possible for CICS dynamically to switch to a different JVM profile for debugging. This has two advantages over previous CICS releases. Firstly, it requires much less set-up in CICS than used to be required, and, secondly, the debugging activity is more tightly scoped to the user who wants to perform it.

Debugging profiles should be defined for each instance where a debug tool needs to be invoked. For example, if an application consists of a Java front end to a COBOL program, then typically two debugging profiles will be required – one for the Java program and one for the COBOL program. If the COBOL program is large, it might be useful to have one debugging profile for each compile unit of the COBOL program.

ACTIVATION AND USE OF DEBUGGING PROFILES

Debugging profiles exist in one of two states – they are either ‘inactive’ or ‘active’. A newly-created profile is always inactive. This means that it will not cause a debug tool to be triggered. In order to be used, profiles must be activated. This can be achieved by using either CADP or the Web interface.

When activating a profile, or set of profiles, the option to set an output device for debugging of Language Environment-supported language programs is presented. This is where the choice of sending the debugging session to a 3270 or to a workstation is made. If CADP is being used, the default is to send the output to the terminal where CADP is being run. If the Web interface is being used, the default is to use TCP/IP to send to the address of the workstation with a port number of 8001. For Java programs, the equivalent definitions are in the JVM profile.

Another important choice to be made here, if TCP/IP is to be used, is whether to use single or multiple sockets for communication. The default is single and this is the preferred setting for WebSphere Studio products. Multiple must be used if the workstation is running a VisualAge debug tool. Note that some firewalls prevent multiple socket communications from working. If the wrong option is chosen, the transaction being debugged is likely to abend with a 4038 abend code.

Once profiles have been created and activated, debugging can occur. When a transaction is starting in a CICS region with DEBUGTOOL set to YES, a process of 'pattern matching' is implemented. This is the process of checking whether there are any active profiles that match the instance of the starting transaction.

To understand this, consider the sample profiles below, which are all assumed to be active:

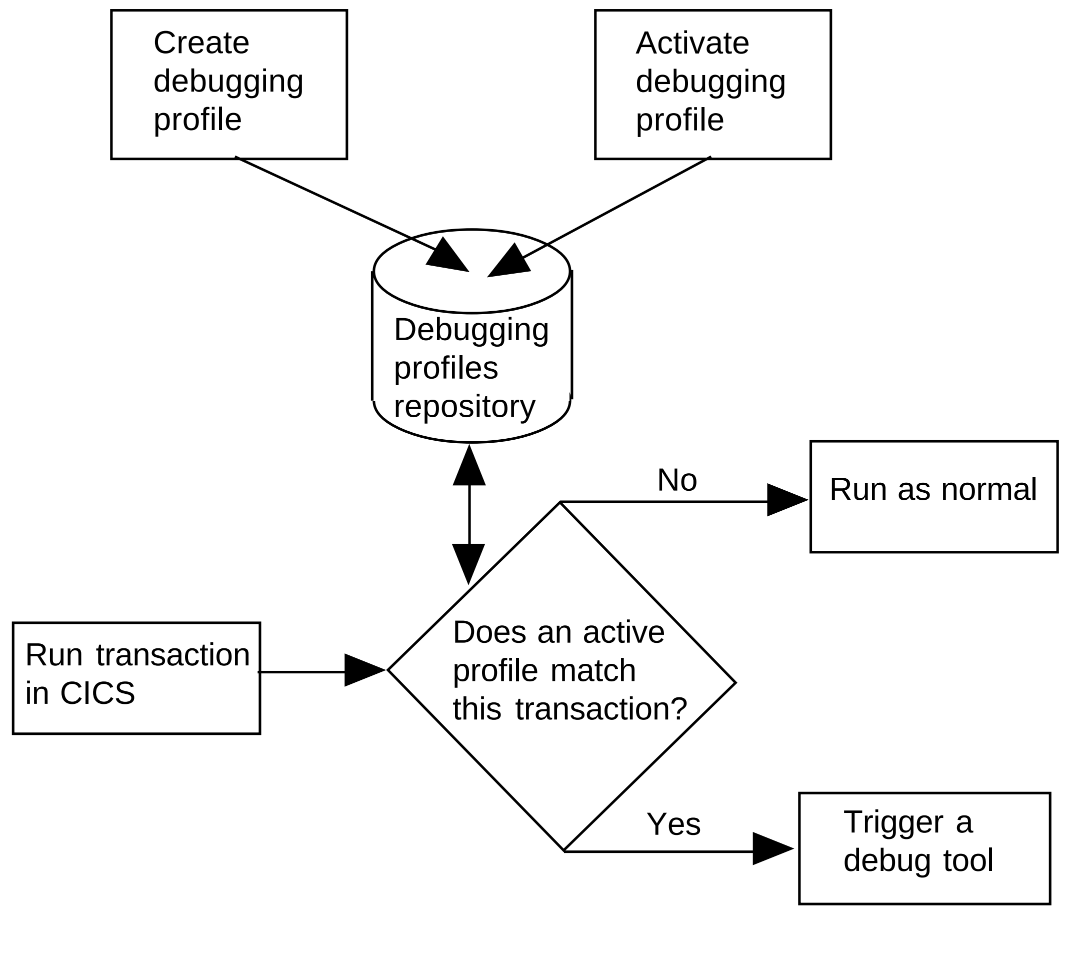
Create debugging profile
Activate debugging profile
Debugging profiles repository
No
Run as normal
Run transaction in CICS
Does an active profile match this transaction?
Yes
Trigger a debug tool

Figure 1: Debugging steps

Debugging profile	*TranID*	*TermID*	*Program*	*UserID*	*Applid*
Profile1	PAY1	TER1	PAYROLL1	TESTER1	CICSTST1
Profile2	PAY*	*	PYRL02	*	*
Profile3	PAY2	*	*	*	CICSTST2

Profile1 is the most specific. Unless all the conditions are matched exactly by a starting transaction, no debugging activity will be triggered. Profile2 will trigger a debug tool for any transaction starting with the characters PAY, which executes program PYRL02 when it is run by any user on a CICS region with any applid. Profile3 is limited to one CICS region (by specifying the applid) and a specific transaction, but whatever program this executes and whichever terminal and user executes the transaction, this profile will match the starting task.

The basic steps required for debugging are shown schematically in Figure 1. It is assumed that CICS already has the files defined for the debugging profiles and that DEBUGTOOL is set to yes.

DEBUGGING PROFILES HINTS AND TIPS

On CICS TS Version 2.3, CADP or the Web interface should be used for defining debugging profiles instead of DTCN. There are several important differences with profiles defined using these new methods. One is that the profiles survive across a CICS restart. This includes preservation of their state – an active profile will remain active. It is best not to have large numbers of active profiles on the repository and to delete profiles completely if they are no longer required.

If it is required to deactivate all profiles, another new CICS-supplied transaction, CIDP, can be used. The transaction can be scheduled to run as part of start-up or shut-down second stage PLT processing. Note that this deactivates the profiles on the repository file, so the change will affect all CICS regions sharing the file.

As a general principle, eschew generic profiles. Having a very generic profile could be troublesome and result in a debug tool being triggered accidentally. For example, if a profile was set with transaction of ‘*’, then every transaction entered by the user will result in a debug tool being triggered. Clearly, this would not be required in most situations.

The output device set when a profile was previously activated is another property that is preserved across a CICS restart. This may cause problems if termids are assigned dynamically by an autoinstall exit or TCP/IP addresses are dynamically allocated for example. A debug session might be started at a terminal or workstation where this is not intended or desired. To reset the output device, a profile needs to be deactivated and reactivated.

There is a performance overhead when CICS needs to check a new transaction instance against the current set of active debugging profiles. In a production CICS region, it is best to run with DEBUGTOOL=NO. If it is necessary to perform debugging activity in a production region, CEMT can be used to switch debugging on temporarily and then switch it off again when the work has been completed.

CONCLUSIONS

Application debugging in CICS has been enhanced by making the initial set up for debugging and the triggering of debug tools simpler. Debugging profiles for LE-supported languages, Java programs, EJBs, and CORBA stateless objects can be defined, making it easier to debug applications in mixed languages. For example, a Java servlet linking to a COBOL program can be handled by using two or more debugging profiles. It is also possible to constrain debugging activity more closely to the user who actually needs to do the debugging. The new debugging profiles, which are defined using the CADP transaction or the Web interface, can be shared across multiple CICS regions by multiple users. The profiles survive a CICS restart. Use of the CIDP transaction may be useful for global inactivation of profiles should this be required.

Darren Beard (darren_beard@uk.ibm.com)
CICS Development
IBM (UK)

Investigating storage violations – part 2

This month we conclude the code that helps to identify the cause of storage violations.

```
*=====================================================================*
*  LINK_REQUEST: COLLECT LINK REQUEST
*   A DATA ENTRY IST ALWAYS FOLLOWED BY A WRAP LINE ENTRY
*=====================================================================*
         SPACE
LINK_REQUEST  DS  ØH
         SPACE
         ST    R14,SAVELIRE
         SPACE
*        CHECK THAT THE COMMAND GROUP CODE CORRESPONDS TO A PC REQUEST
*         FETCH THE PARAMETERS OF THE PC REQUEST
         L     R8,UEPCLPS              FETCH ADDRESS OF COMMAND PLIST
         USING PC_ADDR_LIST,R8
         SPACE
         L     R14,PC_ADDRØ            ADDRESS THE EID..
         USING PC_EID,R14
         SPACE
         MVC   PCGROUP,PC_GROUP        STORE GROUP COMMAND
         MVC   PCFUNCT,PC_FUNCT        STORE FUNCT COMMAND
         SPACE
         CLI   PC_GROUP,PC_PROGRAM_GRP    IS THIS A PC REQUEST?
         BNE   ERROR6                      NO, BRANCH TO ERROR ROUTINE
         SPACE
         CLI   PC_FUNCT,PC_LINK        LINK
         BNE   ERROR7                   NO
         SPACE
         L     R14,PC_ADDR1            ADDRESS THE LINK TO PROGRAM NAME
         MVC   PROGLINK,Ø(R14)         GET THE LINK TO PROGRAMNAME
         MVC   LINKCURR,Ø(R14)         CURRENT LINK TO PROGRAM
         SPACE
         DROP  R14                      DROP ADDRESSABILITY TO EID
         DROP  R8                       DROP ADDRESSABILITY TO CLPS
         SPACE
* ASK FOR CICSKEY
         SPACE
         MVC   RØ74PROK,LINKCURR        CURRENT LINK TO PROGRAM
         BAS   R14,INKUPROK             ASK FOR CICS-KEY
         CLI   RØ74KEYY,PGIS_CICS       ASK FOR CICS-KEY
         BNE   PCCEØ995                  NO CICS-KEY, NO COLLECTION
*        CLC   LINKCURR(L'KDFH),KDFH    ASK FOR CICS-KEY
*        BE    PCCEØ995                  CICS-KEY + DFH, NO COLLECTION
         SPACE
* COLLECT THE PROGRAM LINK REQUEST
         SPACE
         BAS   R14,INQUPROG             GET PROGRAMNAMES BY XPI CALL
         MVC   INVOCURR,ZSPROGIN        INVOKING PROGRAM NAME
         SPACE
PCCEØØ28 DS    ØH
         CLC   LASTPOIN,=CL4'POIN'      LASTPOIN SCHON GESETZT
         BNE   PCCEØØ29                  JA
         LA    R9,TABTCOMA              START OF TABLE
         B     PCCEØØ31                  JA
         SPACE
PCCEØØ29 DS    ØH
         L     R9,LASTPOIN              LAST USED ENTRY
         SPACE
PCCEØØ3Ø DS    ØH
         DS    ØH
         AH    R9,=AL2(L'TABTCOMA)      ADDRESS NEXT ENTRY
PCCEØØ31 DS    ØH
         DS    ØH                       ASK FOR END OF TABLE
         SPACE
* TEST   MVC   R9PRMES4,=CL2'31'        * TEST AID * LOGICAL POINT
* TEST   BAS   R14,R9PRPRIN             * TEST AID * R9 VALUE
         SPACE
         LA    R14,TABTCOMA             START OF TABLE
         A     R14,=AL4(TABTCOME-TABTCOMA) END OF TABLE
         CLC   &A.(L'TABTCOMA,R9),Ø(R14) ADDRESS END OF TABLE
         BNE   PCCEØØ6Ø                  NO END
         SPACE
         MVI   SWTABFUL,1               TABLE IS FULL
         AP    RFTABFU,=P'1'            TABLE IS FULL
         AP    RFERROR,=P'1'            COUNT ERRORS
         MVC   MSGEISØ5(L'MSGTEØ2),MSGTEØ2  TELL ABOUT..
         UNPK  MSGEISØ5+L'MSGTEØ2+2(5),RFTABFU+5(3)  RFTABFU ZEIGEN
         OI    MSGEISØ5+L'MSGTEØ2+2+4,X'FØ'
         CP    RFTABFU,=P'1Ø'           LIMIT MESSAGE
         BH    PCCEØØ6Ø                  ..NO MESSAGE
         BAS   R14,MSG                  COLLECT PC REQUEST IN LAST ENTRY
         B     PCCEØØ6Ø                 COLLEXT PC REQUEST IN LAST ENTRY
         DS    ØH                       IT IS A WRAP LINE ENTRY
         SPACE
PCCEØØ6Ø DS    ØH                      NEW ENTRY
         MVC   DSCALLED,LINKCURR       NEW CALLED PROGRAM
         MVC   DSCALLER,ZSPROGIN       INVOKING PROGRAM NAME
         ZAP   DSTASKN,EIBTASKN        TASKNUMBER
         MVC   DSDATE,EIBDATE          DATE
         MVC   DSTIME,EIBTIME          TIME
         AP    RFENTRY,=P'1'           COUNT USED ENTRIES
         SPACE
PCCEØØ65 DS    ØH                      EXISTING QUEUE
         SPACE
         DS    ØH                      REMEMBER LAST VALUES
         DS    ØH                      ASK FOR END OF TABLE
         CLI   SWTABFUL,1              ASK FOR TABLE FULL
         BNE   PCCEØØ68                 NO END
         LA    R9,TABTCOMA             START OF TABLE
         SPACE
* TEST   MVC   R9PRMES4,=CL2'65'       * TEST AID * LOGICAL POINT
* TEST   BAS   R14,R9PRPRIN            * TEST AID * R9 VALUE
         SPACE
         B     PCCEØØ69
         SPACE
PCCEØØ68 DS    ØH
         DS    ØH
         AH    R9,=AL2(L'TABTCOMA)     ADDRESS NEXT ENTRY FOR WRAP DATA
PCCEØØ69 DS    ØH
         SPACE
* TEST   MVC   R9PRMES4,=CL2'69'       * TEST AID * LOGICAL POINT
* TEST   BAS   R14,R9PRPRIN            * TEST AID * R9 VALUE
         SPACE
         MVC   DSCALLED,KWRAP8         NEW CALLED PROGRAM, SAY WRAPWRAP
         MVC   DSCALLER,KWRAP8         INVOKING PROGRAM NAME, SAY WRAP.
         ZAP   DSTASKN,EIBTASKN        TASKNUMBER
         MVC   DSDATE,EIBDATE          DATE
         MVC   DSTIME,EIBTIME          TIME
         SPACE
         SH    R9,=AL2(L'TABTCOMA)     ADDRESS PREV ENTRY,DATA,NOT WRAP
         DS    ØH                      REMEMBER LAST
PCCEØØ7Ø DS    ØH
         ST    R9,LASTPOIN              TABLE ADDRESS
         SPACE
* TEST   MVC   R9PRMES4,=CL2'7Ø'       * TEST AID * LOGICAL POINT
* TEST   BAS   R14,R9PRPRIN            * TEST AID * R9 VALUE
         SPACE
         MVC   LASTSYSI,SYSICURR        SYSID
         MVC   LASTCALD,LINKCURR        CALLED PROGRAM NAME
         MVC   LASTCALR,INVOCURR        CALLER PROGRAM NAME
         SPACE
PCCEØ15Ø DS    ØH
```

```
         SPACE
* COLLECT THE PROGRAM LINK REQUEST  AS PROGRAM USAGE COUNT
         SPACE
         DS    ØH                       MAINTAIN COLLECT-SUMM-TABLE
         DS    ØH
         LA    R1,TABOCOMA              ADDRESS OF COLLECT-TABLE
         USING OFTLTABL,R1
         SPACE
PCEEØ7ØØ DS    ØH                       INIT COLLECT-TABLE LOOP
         SPACE
         CLC   OSCALLED,LINKCURR        CALLED PROGRAM FOUND
         BE    PCEEØ71Ø                  YES
         CLC   OSCALLED,KEMPTY          EMPTY ENTRY
         BE    PCEEØ7Ø5                  YES
         SPACE
         AH    R1,=AL2(L'TABOCOMA)      ADDRESS NEXT ENTRY
         LA    R14,TABOCOMA             START OF TABLE
         A     R14,=AL4(TABOCOME-TABOCOMA) ADDRESS TABLES END
         CLC   Ø(L'TABOCOMA,R1),Ø(R14)  END OF TABLE
         BNE   PCEEØ7ØØ                  NO
         SPACE
         AP    RFTABVO,=P'1'           TABLE IS FULL
         AP    RFERROR,=P'1'           COUNT ERRORS
         MVC   MSGEISØ5(L'MSGTEØ2O),MSGTEØ2O TELL ABOUT..
         UNPK  MSGEISØ5+L'MSGTEØ2O+2(5),RFTABVO+5(3)  RFTABVO ZEIGEN
         OI    MSGEISØ5+L'MSGTEØ2O+2+4,X'FØ'
         CP    RFTABVO,=P'1Ø'          LIMIT MESSAGE
         BH    PCEEØ79Ø                 ..NO MESSAGE
         BAS   R14,MSG                 COLLECT PC REQUEST IN LAST ENTRY
         B     PCEEØ79Ø                COLLEXT PC REQUEST IN LAST ENTRY
         DS    ØH                       IT IS A WRAP LINE ENTRY
         SPACE
PCEEØ7Ø5 DS    ØH
         MVC   OSCALLED,LINKCURR        CALLING PROGRAM
         SPACE
PCEEØ71Ø DS    ØH
         MVC   OSCALLER,ZSPROGIN        CALLIED PROGRAM
         AP    OSCOUNT,=P'1'            USAGE COUNT
         ZAP   OSTASKN,EIBTASKN         TASKNUMBER
         ZAP   OSTIME,EIBTIME           EIBTIME
PCEEØ79Ø DS    ØH
         DROP  R1
         SPACE
*  END OF  PC_REQUEST_COMPLETE
         SPACE
PCCEØ995 DS    ØH
         SPACE
         L     R14,SAVELIRE
         BR    R14                      RETURN TO CALLER
         EJECT
         SPACE
*======================================================================*
*  XPI CALL
*    INQUIRE CURRENT PROGRAM
*======================================================================*
         SPACE
INQUPROG DS    ØH
         ST    R14,SAVEINQU
         L     R5,UEPXSTOR             ADDRESS XPI PARAMETER LIST
         USING DFHPGIS_ARG,R5
         L     R13,UEPSTACK            ADDRESS KERNEL STACK
         DFHPGISX CALL,                                                X
               CLEAR,                                                  X
               IN,                     IN                              X
               FUNCTION(INQUIRE_CURRENT_PROGRAM),                      X
               OUT,                    OUT                             X
               INVOKING_PROGRAM_NAME(ZSPROGIN),                        X
               RESPONSE(*),                                            X
               REASON(*)
         SPACE
         L     R13,UEPEPSA             ADDRESS ORIGINAL SAVE AREA
         CLI   PGIS_RESPONSE,PGIS_OK   CHECK RESPONSE
         BE    INQUPRO9
* ***    CLI   PGIS_REASON,PGIS_.....  CHECK RESPONSE REASON
* ***    DS    ØH                       REASON IS NOT CHECKED
* ***    BE    *
         DROP  R5
         B     ERROR2Ø                 SEND ERROR MESSAGE
         SPACE
INQUPRO9 DS    ØH
         MVC   PROGINVO,ZSPROGIN       INVOKING PROGRAM
         L     R14,SAVEINQU
         BR    R14
         EJECT
         SPACE
*======================================================================*
*  XPI CALL
*    INQUIRE PROGRAM
*    ASK FOR CICS-KEY PROGRAM
*======================================================================*
         SPACE
INKUPROK DS    ØH
         ST    R14,SAVEINKU
         L     R5,UEPXSTOR             ADDRESS XPI PARAMETER LIST
         USING DFHPGIS_ARG,R5
         L     R13,UEPSTACK            ADDRESS KERNEL STACK
         DFHPGISX CALL,                                                X
               CLEAR,                                                  X
               IN,                     IN                              X
               FUNCTION(INQUIRE_PROGRAM),                              X
               PROGRAM_NAME(RØ74PROK),                                 X
               OUT,                    OUT                             X
               EXECUTION_KEY(RØ74KEYY),                                X
               RESPONSE(*),                                            X
               REASON(*)
         SPACE
         L     R13,UEPEPSA             ADDRESS ORIGINAL SAVE AREA
         CLI   PGIS_RESPONSE,PGIS_OK   CHECK RESPONSE
         BE    INKUPRO9
* ***    CLI   PGIS_REASON,PGIS_.....  CHECK RESPONSE REASON
* ***    DS    ØH                       REASON IS NOT CHECKED
* ***    BE    *
         DROP  R5
         XC    RØ74KEYY,RØ74KEYY       CLEAR CICS KEY SIGNAL
         MVC   PROKLINK,RØ74PROK       REMEMBER PROGRAM NAME
         AP    RFPROKLI,=P'1'          COUNT ERRORS
         DS    ØH                      ..THESE ARE PROG AUTOINSTALL
         DS    ØH                         NOT FOUND PROBLEMS
         DS    ØH                         PROGRAMS IN CICSKEY ARE
         DS    ØH                         RDO DEFINED, NOT
         DS    ØH                         AUTOINSTALLED
         DS    ØH                         THESE ARE PROGRAMS NOT
         DS    ØH                         IN CICSKEY
*        B     ERROR2ØK                SEND ERROR MESSAGE
         SPACE
INKUPRO9 DS    ØH
         L     R14,SAVEINKU
         BR    R14
         EJECT
         SPACE
*=====================================================================*
*  MESSAGE OUTPUT
*=====================================================================*
         SPACE
MSG      DS    ØH
         ST    R14,SAVEMSG
         ST    R1,SAVEMSG1
         CLC   MSGEISØ1,=C'GLUEXPCC'   MESSAGE BOX INITIALISED
         BNE   MSGBOXN1                 NO, EARLY STATE
         CICWTO MSGEISA
         MVC   MSGEISØ5,BLANK
MSGEND   DS    ØH
         L     R14,SAVEMSG             RELOAD R14
         L     R1,SAVEMSG1
         BR    R14                     RETURN TO CALLER
         SPACE
MSGBOXN1 DS    ØH
         CICWTO MSGBOXN2               EARLY STATE MESSAGE
         B     MSGEND
MSGBOXN2 DFHMSG (MSGBOXN3,C'GLUEXPCC MSGEIS NOT INITIALIZED')
```

```
         EJECT
         SPACE
*=====================================================================*
*  ERROR HANDLING
*    ERROR HAS OCCURRED DURING PROCESSING
*=====================================================================*
         SPACE
ERROR1   DS    ØH
         MVC   MSGEISØ5(L'ERRTEØ1),ERRTEØ1
         B     ERRORMSG
ERRTEØ1  DC    C'INVALID EXIT ID ENTERED'
         SPACE
ERROR5   DS    ØH
         MVC   MSGEISØ5(L'ERRTEØ5),ERRTEØ5
         B     ERRORMSG
ERRTEØ5  DC    C'EXIT RECURSIVE ENTERED'
         SPACE
ERROR6   DS    ØH
         MVC   MSGEISØ5(L'ERRTEØ6),ERRTEØ6
         B     ERRORMSG
ERRTEØ6  DC    C'NO PC REQUEST ENTERED'
         SPACE
ERROR7   DS    ØH
         MVC   MSGEISØ5(L'ERRTEØ7),ERRTEØ7
         B     ERRORMSG
ERRTEØ7  DC    C'NO PC FUNCTION ENTERED'
         SPACE
ERROR1Ø  DS    ØH
         MVC   MSGEISØ5(L'ERRTE1Ø),ERRTE1Ø
         B     ERRORMSG
ERRTE1Ø  DC    C'ADDRESS CWA ERROR'
         SPACE
ERROR11  DS    ØH
         MVC   MSGEISØ5(L'ERRTE11),ERRTE11
         B     ERRORMSG
ERRTE11  DC    C'LAYOUT CWA ERROR 1'
         SPACE
ERROR12  DS    ØH
         MVC   MSGEISØ5(L'ERRTE12),ERRTE12
         B     ERRORMSG
ERRTE12  DC    C'LAYOUT CWA ERROR 2'
         SPACE
ERROR13  DS    ØH
         MVC   MSGEISØ5(L'ERRTE13),ERRTE13
         B     ERRORMSG
ERRTE13  DC    C'GETMAIN ERROR TABLE STORAGE'
         SPACE
ERROR14  DS    ØH
         MVC   MSGEISØ5(L'ERRTE14),ERRTE14
         B     ERRORMSG
```

```
ERRTE14  DC    C'LENGTH ERROR IN MSGBOX/MSGEIS'
         SPACE
         SPACE
ERROR19  DS    ØH
         MVC   MSGEISØ5(L'ERRTE19),ERRTE19
         B     ERRORMSG
ERRTE19  DC    C'FORCED TEST ERROR'
         SPACE
ERROR2Ø  DS    ØH
         MVC   MSGEISØ5(L'ERRTE2Ø),ERRTE2Ø
         B     ERRORMSG
ERRTE2Ø  DC    C'ERROR INQUIRE CURRENT PROGRAM'
         SPACE
ERROR2ØK DS    ØH
         MVC   MSGEISØ5(L'ERRTE2ØK),ERRTE2ØK
         B     ERRORMSG
ERRTE2ØK DC    C'ERROR INQUIRE PROGRAM'
         SPACE
ERROR21  DS    ØH
         MVC   MSGEISØ5(L'ERRTE21),ERRTE21
         B     ERRORMSG
ERRTE21  DC    C'ERROR INQUIRE TRANSACTION'
         SPACE
ERROR47  DS    ØH
         MVC   MSGEISØ5(L'ERRTE47),ERRTE47
         B     ERRORMSG
ERRTE47  DC    C'ERROR UEPGAL, NO LENGTH'
         SPACE
ERROR48  DS    ØH
         MVC   MSGEISØ5(L'ERRTE48),ERRTE48
         B     ERRORMSG
ERRTE48  DC    C'ERROR UEPGAL, TOO SMALL'
         SPACE
ERROR49  DS    ØH
         MVC   MSGEISØ5(L'ERRTE49),ERRTE49
         B     ERRORMSG
ERRTE49  DC    C'ERROR UEPGAA, NO ADDRESS'
         SPACE
ERRTEØØ  DC    C'*DUMMY MESSAGE*'
         SPACE
ERRORMSG DS    ØH
         OC    RFERROR,RFERROR         ERROR COUNT NOT INIT
         BZ    *+1Ø                     YES
         AP    RFERROR,=P'1'           COUNT ERRORS
         BAS   R14,MSG                 GIVE ERROR MESSAGE
         B     RETURN
         EJECT
         SPACE
***********************************************************************
* CONSTANTS
```

```
***********************************************************************
         SPACE
         DS    ØD
EYE_CATCHER   DC CL16'XPCREQ  STORAGE ' EYECATCHER
KTPXYCIC DC    CL8'TPXYCICS'           CICS JOB NAME PATTERN
BLANK    DC    CL8Ø' '                 BLANKS
         SPACE
KEMPTY   DC    CL8'**EMPTY*'           EMPTY ENTRY
KWRAP8   DC    CL8'WRAPWRAP'           WRAP LINE
         SPACE
MSGTEØ2  DC    C'COLLTABL IS FULL'
MSGTEØ2O DC    C'OFTLTABL IS FULL'
MSGTEØ2S DC    C'OFTLTABL TOO SMALL'
         SPACE
* STANDARD MESSAGE BOX, COPIED IN DFHEISTG AND FILLED THERE
*  KEEP IN STEP WITH MSGTEX
MSGXWAGMK   DFHMSG (MSGXWAØ1,CLØ9'GLUEXPCC ',                        *
               MSGXWAØ2,CL25'CWA ADDRESSED, GETMAIN OK')
         SPACE
         DS ØF
MSGBOXA  DFHMSG  (MSGBOXØ1,CLØ8'GLUEXPCC',    EXIT NAME              X
               MSGBOXØ2,CLØ1' ',              FILLER                 X
               MSGBOXØ3,CLØ4'SYSI',           SYSID                  X
               MSGBOXØ4,CLØ1' ',              FILLER                 X
               MSGBOXØ5,CL4Ø'MSGBOXMSGBOX')   TEXT
MSGBOXE  DS      ØF                           END OF MESSAGE BOX
         SPACE
***********************************************************************
* R9 TABLE-REGISTER DISPLAY AS TEST AID
***********************************************************************
         SPACE
R9PRPRIN DS    ØH                      R9 VALUE    DISPLAY
         ST    R14,R9PRSAVE            ST R14 UPRO SAVE
         MVC   R9PRMESZ,R9PRMES4       SAVE CALLERS POINT
         MVC   R9PRMESS,BLANK          BLANK MESSAGE AREA
         MVC   R9PRMES4,R9PRMESZ       RESTORE CALLERS POINT
         MVC   R9PRMESØ,=CL1Ø' GLUEXPCC ' EXIT PROGRAMNAME
         MVC   R9PRMES1,=CL4' R9='     HEADER
         ST    R9,R9PRR9R9             R9
         UNPK  R9PRMES2(L'R9PRMES2+L'R9PRMES3),R9PRR9R9(L'R9PRR9R9+1)
         MVI   R9PRMES3,C' '           UNPK ZONE
         NC    R9PRMES2,NCMASK         LEFT HALFBYTE GETS ZERO
         TR    R9PRMES2,TRTAB          TR TO CHARACTER
         MVC   R9PRLENG,=A(L'R9PRMESS) LENGTH FOR WTO
         EXEC CICS WRITE OPERATOR TEXT (R9PRMESS) TEXTLENGTH (R9PRLENG)X
               RESP (R9PRRESP)
         L     R14,R9PRSAVE            RELAOD R14
         BR    R14                     GO BACK
         SPACE
NCMASK   DC    X'ØFØFØFØFØFØFØFØF'     NC MASK OFF GETS OFF
```

```
TRTAB    DC    C'Ø123456789ABCDEF'     TRTAB
         SPACE
         SPACE
***********************************************************************
* NAMES OF THE EXPECTED CICS-KEY LINKED TO PROGRAMS
*  IN SEQUENCE OF HIGHEST USAGE
*  TABLE MUST HAVE LESS OR EQUAL ENTRIES TO TABLE TABOCOMA / OFTLTABL
***********************************************************************
TABONAMØ DC    ØCL8' '
         SPACE
* COLLECTION   27.3.2ØØ3      USAGE COUNT
*  NO USAGE MEANS USAGE FROM Ø TO 1Ø
CPMODUMB DC    CL8'CPMODUMB'  54156Ø   UMBRELLA CODE FOR OMEGAMON
DRSSINTC DC    CL8'DRSSINTC'  365796   LISTEN IN JES SPEICHERN
CPMISTØØ DC    CL8'CPMISTØØ'  269232   MIPS SPEICHER GETMAIN
SCRHHP23 DC    CL8'SCRHHP23 '   3672   SCRHHELP
CICSUAUP DC    CL8'CICSUAUP'     894   AUTOINSTALL PROGRAM
CICSTLOA DC    CL8'CICSTLOA'     747   LOAD SUPERVISOR
CPMODUC7 DC    CL8'CPMODUC7'     61Ø   CPMODUC7
EZACICME DC    CL8'EZACICME'     414   CSKL SOCKET INTERFACE
CICSSOCL DC    CL8'CICSSOCL'     272   LINKED FROM CICSSOCO
DFHEMTD  DC    CL8'DFHEMTD '      93   DFHEMTD CEMT
CICSCEML DC    CL8'CICSCEML'      93   LINKED FROM CICSCEMT
CICSUPEP DC    CL8'CICSUPEP'      45   PROGRAM ERROR PROGRAM LINKED
M4DMCPEP DC    CL8'M4DMCPEP'      45   DUMPMASTER PROGRAM ERROR PROGRAM
EZACIC12 DC    CL8'EZACIC12'           CSKL SOCKET INTERFACE
EZACIC2Ø DC    CL8'EZACIC2Ø'           CSKL SOCKET INTERFACE
EZACIC21 DC    CL8'EZACIC21'           CSKL SOCKET INTERFACE
EZACIC22 DC    CL8'EZACIC22'           CSKL SOCKET INTERFACE
DFHPEP   DC    CL8'DFHPEP  '       4   PROGRAM ERROR PROGRAM ORIGINAL
CICSUZNE DC    CL8'CICSUZNE'      78   DFHZNEP
         SPACE
SCREHPIF DC    CL8'SCREHPIF '          SCREHELP TPC1
SCREHFIO DC    CL8'SCREHFIO '          SCREHELP TPC1
SCRETBRO DC    CL8'SCRETBRO '          SCREHELP TPC1
         SPACE
CICSSTAR DC    CL8'CICSSTAR'           PLTPI
CICSSTBR DC    CL8'CICSSTBR'           PLTPI
CICSPLTC DC    CL8'CICSPLTC'           PLTPI PLTSD
CICSBKT2 DC    CL8'CICSBKT2'           PLTPI
CICSTABL DC    CL8'CICSTABL'           PLTPI
CPSTARAØ DC    CL8'CPSTARAØ'           PLTPI
DMCSTART DC    CL8'DMCSTART'           PLTPI PLTSD
DMCSRT31 DC    CL8'DMCSRT31'           PLTPI
KOCOMEØØ DC    CL8'KOCOMEØØ'           PLTPI PLTSD
CICSENQM DC    CL8'CICSENQM'           PLTPI
CODIPLTX DC    CL8'CODIPLTX'           PLTPI
CPMIIOLO DC    CL8'CPMIIOLO'           PLTPI
CTSCRHHØØ DC   CL8'CTSCRHHØØ'          PLTPI
SCRHHPØ5 DC    CL8'SCRHHPØ5 '          PLTPI TOR
```

```
SCRHHP25 DC    CL8'SCRHHP25 '           PLTPI AOR
         SPACE
CICSFINI DC    CL8'CICSFINI'            PLTSD
CICSDB2C DC    CL8'CICSDB2C'     1Ø5    PLTSD
CICSDB2S DC    CL8'CICSDB2S'       1    PLTSD
CICSSOCO DC    CL8'CICSSOCO'       1    PLTSD
         SPACE
CICSUAUT DC    CL8'CICSUAUT'            AUTOINSTALL TERMINAL ONLY TOR
         SPACE
CODITDAT DC    CL8'CODITDAT'            CORE TRANSACTION, DISPLAY
CODIMSGS DC    CL8'CODIMSGS'            CORE TRANSACTION   OF THE
CODIMSGP DC    CL8'CODIMSGP'            CORE TRANSACTION    TABLES
         SPACE
#PRDEEN# DC    CL8'#PRDEEN#'            PREDEFINED PROGRAMS END
         SPACE
TABONAMZ DC    XL8'FFFFFFFFFFFFFFFF'    ENDE DER NAMEN
         SPACE
TABONAMA DC    A((TABONAMZ-TABONAMØ)/8)  NUMBER OF PREDEFINED PROGRAMS
         SPACE
***********************************************************************
* LTORG
***********************************************************************
         LTORG ,
         SPACE
***********************************************************************
* ..DO NOT COMPILE THE FOLLOWING SAMPLE CODE..    AGO .END
***********************************************************************
         AGO   .END
***********************************************************************
*  -- CODE TO ENABLE / DISABLE THIS GLUE EXIT --
***********************************************************************
         SPACE
DFHEISTG DSECT
RESPEXIT DS    F                        E.C. RESPONSE
RESPEXI2 DS    F                        E.C. RESPONSE2
         SPACE
ENABLE   DS    ØH
         EXEC CICS ENABLE PROGRAM ('GLUEXPCC') EXIT ('XPCREQ')         X
               GALENGTH (2Ø48)                                         X
               START  RESP (RESPEXIT) RESP2 (RESPEXI2)
         OC    RESPEXIT,RESPEXIT
         BZ    EXIEOK
         SPACE
DISABLE  DS    ØH
         EXEC CICS DISABLE PROGRAM ('GLUEXPCC') EXIT ('XPCREQ')        X
               STOP   RESP (RESPEXIT) RESP2 (RESPEXI2)
         OC    RESPEXIT,RESPEXIT
         BZ    EXIDOK
***********************************************************************
*  -- CODE TO TAKE A TRANSACTION DUMP IN A PLTSD PROGRAM TO DUMP
*      4Ø96 BYTE OF THE COLLECTED TABLES, CONTAINING THE PROGRAM
*      SUMMARY COMPLETE --
**********************************************************************
DFHEISTG DSECT
EPCCRESP DS    F                        E.C. RESPONSE
EPCCFROM DS    F                        DUMP FROM ADDRESS
EPCCLENG DS    F                        DUMP LENGTH VALUE
         SPACE
         MVC   EPCCFROM,CWAEXIPL        DUMP FROM
         MVC   EPCCLENG,KLEXIPCC        DUMP LENGGTH TABOCOME-TABOCOMA
         SPACE
         EXEC CICS DUMP TRANSACTION DUMPCODE (KCEXIPCC) RESP (EPCCRESP)X
               SEGMENTLIST (EPCCFROM) LENGTHLIST (EPCCLENG)            X
               NUMSEGMENTS(KNEXIPCC)
         SPACE
         B     ........
         SPACE
KPEXIPCC DC    CL8'GLUEXPCC'            GLUEXPCC
KCEXIPCC DC    CL4'CKEY'                DUMP CODE CKEY WIE CICSKEY
KLEXIPCC DC    F'4Ø96'                  LENGTHLIST => TABTCOME-TABTCOMA
KNEXIPCC DC    F'1'                     NUMSEGMENTS
         SPACE
**********************************************************************
*   -- CODE FOR INSERTION IN CICS-KEY PROGRAMS (FOR WHICH YOU HAVE
*      THE SOURCE), WHICH ARE NOT LINKED (EG CALLED BY TRANSACTION)
*      TO HAVE (FORCE) AN ENTRY IN THE TABLES COLLECTED --
*      RDO DEFINE THE LINKED TO PROGRAM AS A CICS-KEY PROGRAM
**********************************************************************
         SPACE
* IF GLUEXPCC IS ACTIVE, FORCE AN ENTRY IN GLUEXPCC-S TABLE
         SPACE
         L     R14,CWAEXIPL             ASK FOR GLUEXPCC ACTIVE
         LTR   R14,R14                   NOT ACTIVE
         BZ    NOEXIPCC
         CLC   Ø(L'GLUEXPCC,R14),GLUEXPCC
         BNE   NOEXIPCC                  NOT ACTIVE
         EXEC CICS LINK PROGRAM (CICSUAUT) NOHANDLE
         B     NOEXIPCC
         SPACE
CICSUAUT DC    CL8'CICSUAUT'            LINKED TO PROGRAM NAME
GLUEXPCC DC    CL8'GLUEXPCC'            EXIT NAME IN TABLE
         SPACE
NOEXIPCC DS    ØH
         SPACE
.END      ANOP
**********************************************************************
* END
**********************************************************************
          END   GLUEXPCC
```

Hannes Bojan
CICS Systems Programmer (Germany) © Xephon 2004

CICS on Windows?

If, like Bertelsmann AG, you were running CICS under VM/VSE, what would you do next? Bertelsmann felt that their mainframe environment did not allow for critical applications to meet future requirements. It did not provide flexibility, scalability, and connectivity in terms of integration with outside solutions. They identified cost pressures on this system, but they also had a desire to reuse existing business processes developed over 19 years.

So what was their solution? To move to an open environment based on Windows, and transform 2,900 programs and 2.5 million lines of COBOL code. And this, they claimed, gave them enhanced performance and halved their costs.

Similar moves to client/server operating environments in the past have rarely matched the stability and performance of the mainframe world. Plus, the cost of new developments has nullified any financial savings, without achieving the functional depth of the old programs.

Their main business-critical application was an in-house enterprise resource planning package developed by ICS. The application covers the central business processes and manages approximately 3.5 million customers of book clubs in Austria, Switzerland, Italy, French Canada, and Poland. The book club software had to deal with different products and prices, and numerous special conditions such as free products, packages, combination products, and the like. In addition, it had to allow for the ‘goods returned’ policies for each country.

Development of this application started in the 1980s. It ran under VSE and CICS and was based on a data model that was structured 100% in a DL/I database*.* The application was developed in approximately 1,600 man-days, completely in COBOL, and contained approximately one million lines of code.

The original application permitted regular updating to be made, and several times it was revised and adapted. By the mid-1990s the ‘flexibility buffer’ originally built into the application was exhausted, and new demands on the system could be achieved only by implementing newer technologies. The developed product structure could no longer be displayed within the DL/I database and it was changed to DB2.

By 1998, Bertelsmann had moved the application’s merchandise management system across to DB2. When ICS wanted to make the corresponding adjustments for the customer administration, they found true limitations in the existing systems. The products of the book clubs are complex in structure, but their number is limited. In turn, customers have become strongly differentiated in interest areas. In addition, the data volumes also have to be considered: the 3.5 million active customers serviced by this application generated more than 100 million account transactions per year. Tests showed that not only did the required storage capacity need to be substantially increased, but the computer run-times needed improving by a factor of four. The company felt that implementing these measures on an IBM mainframe gave no real benefits. More money would be spent without achieving any improvements or increasing market share and penetration.

It was essential to transfer the existing COBOL business logic directly across to the new platform, and do it quickly. Other considerations included the introduction of a graphical user interface, the need to use the Web as a sales channel outlet, and the need to allow a range of business processes from the supplier to the customer via the Internet.

What was the appropriate platform for the future? A change to the AS/400 would have meant a completely differently transaction system with consequently large changes to the application software. MVS was ruled out because of cost considerations alone. So they chose Windows servers.

Micro Focus offered a way to transfer the programs completely and without any changes of code on to the Windows platform.

This meant that their mainframe applications would operate on Windows servers without needing to be changed! Micro Focus products were used to manage the COBOL and CICS run-time platforms.

However, during the migration project, ICS deviated slightly from a strict one-to-one mainframe to Windows conversion for pragmatic reasons. First, the DL/I to DB2 database transition required some program adjustments. Secondly, they used this opportunity to change the EBCDIC mainframe data to ASCII.

All the book clubs run by the ICS system were migrated to the Windows platform in six-month stages from 2002, starting with the original ICS kernel application. The migration was not only accomplished problem-free, but it was successful.

The improvements in throughput are largely owing to performance gains from the new server hardware, tuning access to UDB during the transformation of the application, and overall improvements in batch processing.

The performance comparison between Windows servers and the mainframe varied. On line transactions were better on the PC (70% of the mainframe time) and the batch jobs were, although slightly slower on some occasions, 10% better overall. For example, sorting 1.35 million members needs just 7 seconds under COBOL on the Windows PC.

So, did they use specially configured PCs? Surprisingly, no. Each location contains a Compaq ML-530 as a production system with only a single processor, 2 gigabytes of RAM, and a disk capacity of 800GB in the active enterprise.

The company says that the new systems were cheaper initially and are much cheaper to operate and maintain; it claims to have achieved concrete savings of 50,000 euros per month.

Bertelsmann is now moving to Enterprise Server because it provides a strategic platform to deploy its CICS applications, and as a way to use existing business logic as Web services.

While moving off the mainframe is probably not a realistic idea for MVS CICS sites, VM/VSE CICS users may find the experience makes interesting reading.

Editor's note: we would like to hear what other CICS users think about migrating off the mainframe to Windows. E-mail your comments to Trevor Eddolls at trevore@xephon.com.

Execute native CEMT commands from batch

This is a REXX EXEC I wrote to execute native CEMT commands from batch using the CPSM API. The goal was to allow jobs to manage their own CICS resources and get accurate return codes back in the JCL.

Sample JCL and usage instructions are in the comments of the program.

The following CEMT commands are supported:

- CEMT SET PROG – ENABLE, DISABLE, NEWCOPY, PHASEIN.
- CEMT SET TRAN – ENABLE, DISABLE.
- CEMT SET FILE – CLOSE, OPEN, DISABLE, ENABLE.
- CEMT SET DB2CONN – CONNECTED, NOTCONNECT, SECURITY, FORCE.
- CEMT SET CONN – ACQUIRE, BACKOUT, CANCEL, COMMIT, ENDAFFINITY, FORCE, INSERVICE, NORECOVDATA, NOTPENDING, OUTSERVICE, PURGE, RELEASE, RESYNC.

```
/*******************************************************************/
/*                              REXX                               */
/* Purpose: Parse CEMT commands and use CPSM to run equivalents     */
/* Syntax:  cemtset context scope cmas simulate                    */
/* Parms: context    - The CPSM context value (required)           */
/*        scope      - The CPSM scope value (required)             */
/*        cmas       - CMAS name (required)                        */
/*        simulate   - Show what would happen, but don't do it     */
/*                   - optional (any value except NO)              */
/* Notes: All CEMT commands are read from the CEMTCMDS DD statement. */
/*                                                                 */
/* Sample JCL for CEMTSET:                                         */
/* //jobcard  JOB  ....                                            */
/* //CEMTSET  EXEC PGM=IKJEFTØ1,PARM='CEMTSET context scope cmas'   */
/* //STEPLIB  DD   DSN=your.cics.seyuauth.pds,DISP=SHR             */
/* //SYSEXEC  DD   DSN=your.exec.pds,DISP=SHR                      */
/* //SYSTSPRT DD   SYSOUT=*                                        */
/* //DIAGMSGS DD   SYSOUT=*                                        */
/* //SYSTSIN  DD   DUMMY                                           */
/* //CEMTCMDS DD   DSN=your.source(member),DISP=SHR                */
/* Optionally, you can add a CMASMAP DD to look up the CMAS based on */
/* the LPAR name.  The CMAS name will be first taken from the EXEC  */
/* card PARM.  If not found in the PARM, then a CMASMAP DD is used.  */
/* If the CMASMAP DD is missing, the internal name pattern is used.  */
/* //CMASMAP  DD   DSN=your.source(member),DISP=SHR                */
/*                                                                 */
/* Sample CEMTCMDS input member:                                   */
/*   ----+----1----+----2----+----3----+----4----+----5            */
/*   * Comment lines have an asterisk in column one                */
/*   * Although CEMT S TRAN(TR1 TR2 TR3 TR4) DIS is supported      */
/*   * it is very slow.  It is much faster to use 4 commands       */
/*   * CEMT S TRAN(TR1) DIS                                        */
/*   * CEMT S TRAN(TR2) DIS                                        */
/*   * CEMT S TRAN(TR3) DIS                                        */
/*   * CEMT S TRAN(TR4) DIS                                        */
/*   * Currently CONN, DB2CONN, FILE, PROG and TRAN are supported   */
/*   CEMT SET TRAN(AA*) DIS                                        */
/*   * Upper, lower or mixed case is supported                     */
/*   cemt set tran(aa*) ena                                        */
/*                                                                 */
/* Any CEMT command could potentially return a non-zero return code. */
/* If you must ensure that a command executed successfully, use only */
/* one command per step.                                           */
/*                                                                 */
/* Sample CMASMAP input member:                                    */
/*   ----+----1----+----2----+----3----+----4----+----5            */
/*   * Comment lines have an asterisk in column one                */
/*   * Free form starting column position                          */
/*   * LPAR CMASNAME                                               */
/*   sys1 cmas1                                                    */
/*   sys2 cmas2     Everything after 2 words is ignored              */
/*   fred yrb49a                                                     */
/* Standard housekeeping activities                                 */
 call time 'r'
 parse arg parms
 signal on syntax name trap
 signal on failure name trap
 signal on novalue name trap
 probe = 'NONE'
 modtrace = 'NO'
 modspace = ''
 call stdentry 'DIAGMSGS'
 module = 'MAINLINE'
 push trace() time('L') module 'From:' Ø 'Parms:' parms
 if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
 call modtrace 'START' Ø
/* Set local esoteric names                                          */
 @vio   = 'VIO'
 @sysda = 'SYSDA'
/* Validate parms                                                    */
 arg context scope cmas simulate
 if context  = '' then call rcexit 8 'Missing CPSM Context'
 if scope    = '' then call rcexit 8 'Missing CPSM Scope'
 if cmas     = '' then cmas = cpsmcmas()
 if simulate = '' then simulate = 'NO'
/* Identify CONTEXT and SCOPE being used                             */
 if tsoenv = 'BACK' then say
 call msg 'CPSM Context:' context 'specified'
 call msg 'CPSM Scope:' scope 'specified'
/* Determine if the CEMTCMDS DD is allocated                         */
 if tsoenv = 'BACK' then say
 call ddcheck 'CEMTCMDS'
 call msg 'CEMT commands are provided in DD CEMTCMDS DSN' sysdsname
 if tsoenv = 'BACK' then say
/* Load all the input commands into a stem                           */
 call tsotrap "EXECIO * DISKR CEMTCMDS (STEM CEMTCMD. FINIS"
/* Connect to the local CMAS                                         */
 cpsm_thread = cpsminit(cmas)
/* Get the number of regions in the scope                            */
 regcount = cpsmget(cpsm_thread context scope 'CICSRGN')
 parse var regcount . regcount
 call msg scope 'in' context 'contains' regcount 'regions'
 if tsoenv = 'BACK' then say
/* Print the command and call the CEMT parser. Ignore any lines in   */
/* the input file with a '*' in column 1.                            */
 do i=1 to cemtcmd.Ø
    if substr(cemtcmd.i,1,1) = '*' then iterate
/* Allows a WAIT nn statement to cause an 'n' second pause between   */
    if word(cemtcmd.i,1) = 'WAIT' then
       do
```

```
        call wait word(cemtcmd.i,2)
        iterate
       end
    call msg cemtcmd.i
    if tsoenv = 'BACK' then say
    call cemtpar cemtcmd.i
 end
/* Terminate the CPSM connections                                    */
 call cpsmterm
/* Shutdown                                                          */
 shutdown: nop
/* Put unique shutdown logic before the call to stdexit              */
/* Shutdown message and terminate                                    */
          call stdexit time('e')
/* All internal subroutines specific to CEMTSET (not refreshable)    */
/* CEMTPAR  - CEMT Parser                                            */
/* CEMTCON  - Parse and format CPSM Object/Actions for Connections   */
/* CEMTDB2C - Parse and format CPSM Object/Actions for DB2 Attach    */
/* CEMTFI   - Parse and format CPSM Object/Actions for Files         */
/* CEMTPROG - Parse and format CPSM Object/Actions for Programs      */
/* CEMTTRA  - Parse and format CPSM Object/Actions for Transactions  */
/* CMDPRINT - Execute the command and print what happened            */
/* Parse the incoming CEMT command                                   */
 cemtpar: arg cemtcmd
         if pos('(',cemtcmd) = 0 then
            parse var cemtcmd cemt cemtf cemtt cemto
         else
            parse var cemtcmd cemt cemtf cemtt '(' cemtr ') ' cemto
         cemto = strip(cemto)
/* Confirm this is a CEMT command                                    */
         if cemt = 'CEMT' then
            do
/* Confirm this is a CEMT SET command                                */
             if substr(cemtf,1,1) = 'S' then
                do
/* Determine which type of CICS resource this is                     */
                 select
                   when substr(cemtt,1,3) = 'CON' then call cemtcon
                   when substr(cemtt,1,4) = 'DB2C' then call cemtdb2c
                   when substr(cemtt,1,2) = 'FI' then call cemtfi
                   when substr(cemtt,1,3) = 'TRA' then call cemttra
                   when substr(cemtt,1,4) = 'PROG' then call cemtprog
                   otherwise
                        do
                         call rcexit 20 'CEMT Resource Type:' cemtt,
                                        'not supported'
                        end
                 end
                 return
                end
```

```
/* Everything that isn't a CEMT SET command                          */
              else
                 do
                call rcexit 20 cemtf 'is an unsupported CEMT function'
                 end
             end
/* Everything that isn't a CEMT command                              */
          else
              do
               call rcexit 20 cemtcmd 'is an unsupported command'
              end
/* Process Connections                                               */
 cemtcon: object = 'CONNECT'
          poparm = 'NONE'
/* Build the filter                                                  */
          filter = ''
          do r=1 to words(cemtr)
             filter = filter || ' NAME =' word(cemtr,r)
             if r < words(cemtr) then filter = filter || ' OR'
          end
          do l=1 to words(cemto)
/* Map CEMT command options to supported CPSM Resource Actions       */
             select
               when substr(word(cemto,l),1,2) = 'AC' then
                    action = 'ACQUIRE'
               when substr(word(cemto,l),1,1) = 'B' then
                    action = 'BACKOUT'
               when substr(word(cemto,l),1,2) = 'CA' then
                    action = 'CANCEL'
               when substr(word(cemto,l),1,2) = 'CO' then
                    action = 'COMMIT'
               when substr(word(cemto,l),1,2) = 'EN' then
                    action = 'ENDAFFINITY'
               when substr(word(cemto,l),1,6) = 'FORCEU' then
                    action = 'FORCE'
               when substr(word(cemto,l),1,2) = 'IN' then
                    action = 'INSERVICE'
               when substr(word(cemto,l),1,3) = 'NOR' then
                    action = 'NORECOVDATA'
               when substr(word(cemto,l),1,3) = 'NOT' then
                    action = 'NOTPENDING'
               when substr(word(cemto,l),1,2) = 'OU' then
                    action = 'OUTSERVICE'
               when substr(word(cemto,l),1,2) = 'PU' then
                    action = 'PURGE'
               when substr(word(cemto,l),1,3) = 'REL' then
                    action = 'RELEASE'
               when substr(word(cemto,l),1,3) = 'RES' then
                    action = 'RESYNC'
               otherwise
```

```
                     do
                      call msg 'CEMT Option:' word(cemto,l) 'is not',
                               'supported for CPSM object' object,
                               '- NO ACTION TAKEN'
                      MAXRC = 4
                      leave
                     end
              end
              call cmdprint object action poparm filter
           end
           if tsoenv = 'BACK' then say
           return
/* Process DB2 Connections                                            */
 cemtdb2c: object = 'DB2CONN'
           poparm = 'BUSY(WAIT)'
/* Build the filter                                                   */
           filter = 'NAME = *'
           do l=1 to words(cemto)
/* Map CEMT command options to supported CPSM Resource Actions        */
              select
                when substr(word(cemto,l),1,8) = 'SECURITY' then
                     do
                      action = 'REBUILD'
                      poparm = 'NONE'
                     end
                when substr(word(cemto,l),1,9) = 'CONNECTED' then
                     do
                      action = 'CONNECT'
                      poparm = 'BUSY(WAIT)'
                     end
                when substr(word(cemto,l),1,12) = 'NOTCONNECTED' then
                     do
                      action = 'DISCONNECT'
                      poparm = 'BUSY(WAIT)'
                     end
                when substr(word(cemto,l),1,5) = 'FORCE' then
                     do
                      action = 'FORCE'
                      poparm = 'NONE'
                     end
                otherwise
                    do
                     call msg 'CEMT Option:' word(cemto,l) 'is not',
                              'supported for CPSM object' object,
                              '- NO ACTION TAKEN'
                     MAXRC = 4
                     leave
                    end
              end
              call cmdprint object action poparm filter
```

```
          end
          if tsoenv = 'BACK' then say
          return
/* Process Files                                                   */
 cemtfi: object = 'LOCFILE'
/* Build the filter                                                */
        filter = ''
        do r=1 to words(cemtr)
           filter = filter || ' FILE =' word(cemtr,r)
           if r < words(cemtr) then filter = filter || ' OR'
        end
        do l=1 to words(cemto)
/* Map CEMT command options to supported CPSM Resource Actions       */
           select
             when substr(word(cemto,l),1,2) = 'DI' then
                  do
                   action = 'DISABLE'
                   poparm = 'NONE'
                  end
             when substr(word(cemto,l),1,2) = 'CL' then
                  do
                   action = 'CLOSE'
                   poparm = 'BUSY(WAIT)'
                  end
             when substr(word(cemto,l),1,2) = 'OP' then
                  do
                   action = 'OPEN'
                   poparm = 'NONE'
                  end
             when substr(word(cemto,l),1,2) = 'EN' then
                  do
                   action = 'ENABLE'
                   poparm = 'NONE'
                  end
             when substr(word(cemto,l),1,1) = 'F'  then
                  poparm = 'BUSY(FORCE)'
             otherwise
                  do
                   call msg 'CEMT Option:' word(cemto,l) 'is not',
                            'supported for CPSM object' object,
                            '- NO ACTION TAKEN'
                   MAXRC = 4
                   leave
                  end
            end
            call cmdprint object action poparm filter
        end
        if tsoenv = 'BACK' then say
        return
/* Process Programs                                                */
```

```
 cemtprog: object = 'PROGRAM'
           poparm = 'NONE'
/* Build the filter                                                   */
           filter = ''
           do r=1 to words(cemtr)
              filter = filter || ' PROGRAM =' word(cemtr,r)
              if r < words(cemtr) then filter = filter || ' OR'
           end
           do l=1 to words(cemto)
/* Map CEMT command options to supported CPSM Resource Actions        */
              select
                when substr(word(cemto,l),1,2) = 'DI' then
                     action = 'DISABLE'
                when substr(word(cemto,l),1,2) = 'EN' then
                     action = 'ENABLE'
                when substr(word(cemto,l),1,2) = 'NE' then
                     action = 'NEWCOPY'
                when substr(word(cemto,l),1,2) = 'PH' then
                     action = 'PHASEIN'
                otherwise
                    do
                     call msg 'CEMT Option:' word(cemto,l) 'is not',
                              'supported for CPSM object' object,
                              '- NO ACTION TAKEN'
                     MAXRC = 4
                     leave
                    end
              end
              call cmdprint object action poparm filter
           end
           if tsoenv = 'BACK' then say
           return
/* Process Transactions                                               */
 cemttra: object = 'LOCTRAN'
          poparm = 'NONE'
/* Build the filter                                                   */
          filter = ''
          do r=1 to words(cemtr)
             filter = filter || ' TRANID =' word(cemtr,r)
             if r < words(cemtr) then filter = filter || ' OR'
          end
          do l=1 to words(cemto)
/* Map CEMT command options to supported CPSM Resource Actions        */
             select
               when substr(word(cemto,l),1,2) = 'DI' then
                    action = 'DISABLE'
               when substr(word(cemto,l),1,2) = 'EN' then
                    action = 'ENABLE'
               otherwise
                   do
```

```
                     call msg 'CEMT Option:' word(cemto,1) 'is not',
                              'supported for CPSM object' object,
                              '- NO ACTION TAKEN'
                     MAXRC = 4
                     leave
                    end
             end
/* Call once for REMTRAN and again for LOCTRAN                        */
             call cmdprint 'REMTRAN' action poparm filter
             call cmdprint 'LOCTRAN' action poparm filter
             if tsoenv = 'BACK' then say
           end
          return
/* Print the command request                                          */
 cmdprint: arg object action poparm filter
           cpsm_parms = cpsm_thread context scope
/* Get the number of qualifying resources                             */
           resnum = cpsmget(cpsm_parms object '#' filter)
           parse var resnum . resnum
/* Perform the action and return the number of affected resources    */
           updnum = Ø
           if simulate = 'NO' then
              do
              updnum = cpsmpobj(cpsm_parms object action poparm filter)
               parse var updnum . updnum
/* Set a non-zero return code if resources found not equal to updated*/
               if resnum <> updnum then
                  do
                   MAXRC = 4
                   call msg execname 'ERROR: Not all CPSM resources',
                                     'found were updated, RC='MAXRC
                   if tsoenv = 'BACK' then say
                  end
              end
           else
              call msg 'Running in SIMULATE mode'
           call msg 'CPSM Object:' object 'CPSM Action:' action',',
                    'CPSM Resources Found:' resnum', Updated:' updnum
           return
/* 27 Internal Subroutines provided in CEMTSET                        */
/* Last Subroutine REFRESH was 29 Apr 2ØØ4 23:53:11                   */
/* RCEXIT   - Exit on non-zero return codes                           */
/* TRAP     - Issue a common trap error message using rcexit          */
/* ERRMSG   - Build common error message with failing line number    */
/* STDENTRY - Standard Entry logic                                    */
/* STDEXIT  - Standard Exit logic                                     */
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/* DDCHECK  - Determine if a required DD is allocated                 */
/* DDLIST   - Returns number of DDs and populates DDLIST variable    */
/* DDDSNS   - Returns number of DSNs in a DD and populates DDDSNS    */
/* TSOTRAP  - Capture the output from a TSO command in a stem        */
/* WAIT     - Wait for a specified number of seconds                  */
/* SAYDD    - Print messages to the requested DD                      */
/* JOBINFO  - Get job related data from control blocks                */
/* ISITUP   - Check to see if an address space is active              */
/* PTR      - Pointer to a storage location                           */
/* STG      - Return the data from a storage location                 */
/* CPSMERR  - Format a CPSM error message for RCEXIT                  */
/* CPSMFDBK - CPSM Feedback command used to collect CPSM error data  */
/* CPSMCMAS - Get CMAS name                                           */
/* CPSMINIT - Initialize a CPSM session                               */
/* CPSMGET  - Get a CPSM Result Set                                   */
/* CPSMPOBJ - Perform an action on a CPSM object                      */
/* CPSMTERM - Terminate a CPSM session                                */
/* MODTRACE - Module Trace                                            */
/* RCEXIT   - Exit on non-zero return codes                           */
/* EXITRC   - Return code to exit with (if non-zero)                  */
/* ZEDLMSG  - Message text for it with for non-zero EXITRCs          */
 rcexit: parse arg EXITRC zedlmsg
         if EXITRC <> Ø then
            do
             trace 'o'
/* If execution environment is ISPF then VPUT ZISPFRC                 */
             if execenv = 'TSO' | execenv = 'ISPF' then
                do
                 if ispfenv = 'YES' then
                    do
                     zispfrc = EXITRC
/* Does not call ISPWRAP to avoid obscuring error message modules    */
                     address ISPEXEC "VPUT (ZISPFRC)"
                    end
                end
/* If a message is provided, wrap it in date, time and EXITRC        */
             if zedlmsg <> '' then
                do
                 zedlmsg = time('L') execname zedlmsg 'RC='EXITRC
                 call msg zedlmsg
                end
/* Write the contents of the Parentage Stack                          */
             stacktitle = 'Parentage Stack Trace ('queued()' entries):'
/* Write to MSGDD if background and MSGDD exists                      */
             if tsoenv = 'BACK' then
                do
                 if subword(zedlmsg,9,1) = msgdd then
                    do
                     say zedlmsg
                     signal shutdown
                    end
                 else
                    do
```

```
                     call saydd msgdd 1 zedlmsg
                     call saydd msgdd 1 stacktitle
                    end
                end
             else
/* Write to the ISPF Log if foreground                              */
                do
                 zerrlm = zedlmsg
                 address ISPEXEC "LOG MSG(ISRZØØ3)"
                 zerrlm = center(' 'stacktitle' ',78,'-')
                 address ISPEXEC "LOG MSG(ISRZØØ3)"
                end
/* Unload the Parentage Stack                                       */
             do queued()
                pull stackinfo
                if tsoenv = 'BACK' then
                   do
                    call saydd msgdd Ø stackinfo
                   end
                else
                   do
                    zerrlm = stackinfo
                    address ISPEXEC "LOG MSG(ISRZØØ3)"
                   end
             end
/* Put a terminator in the ISPF Log for the Parentage Stack         */
             if tsoenv = 'FORE' then
                do
                 zerrlm = center(' 'stacktitle' ',78,'-')
                 address ISPEXEC "LOG MSG(ISRZØØ3)"
                end
/* Signal SHUTDOWN.  SHUTDOWN label MUST exist in the program       */
             signal shutdown
            end
         else
            return
/* TRAP     - Issue a common trap error message using rcexit        */
/* PARM     - N/A                                                   */
 trap: traptype = condition('C')
       if traptype = 'SYNTAX' then
          msg = errortext(RC)
       else
          msg = condition('D')
       trapline = strip(sourceline(sigl))
       msg = traptype 'TRAP:' msg', Line:' sigl '"'trapline'"'
       call rcexit 666 msg
/* ERRMSG   - Build common error message with failing line number   */
/* ERRLINE  - The failing line number passed by caller from SIGL    */
/* TEXT     - Error message text passed by caller                   */
 errmsg: nop
```

```
         parse arg errline text
         return 'Error on statement' errline',' text
/* STDENTRY - Standard Entry logic                                  */
/* MSGDD    - Optional MSGDD used only in background               */
 stdentry: module = 'STDENTRY'
           if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           arg msgdd
           parse upper source . . execname . execdsn . . execenv .
/* Start up values                                                  */
           EXITRC = Ø
           MAXRC = Ø
           ispfenv = 'NO'
           popup = 'NO'
           lockpop = 'NO'
           headoff = 'NO'
           hcreator = 'NO'
           keepstack = 'NO'
           lpar = mvsvar('SYSNAME')
           zedlmsg = 'Default shutdown message'
/* Determine environment                                            */
           if substr(execenv,1,3) <> 'TSO' & execenv <> 'ISPF' then
              tsoenv = 'NONE'
           else
              do
               tsoenv = sysvar('SYSENV')
               signal off failure
              "ISPQRY"
               ISPRC = RC
               if ISPRC = Ø then
                  do
                   ispfenv = 'YES'
/* Check if HEADING ISPF table exists already, if so set HEADOFF=YES */
                   call ispwrap "VGET (ZSCREEN)"
                   if tsoenv = 'BACK' then
                      htable = jobinfo(1)||jobinfo(2)
                   else
                      htable = userid()||zscreen
                   TBCRC = ispwrap(8 "TBCREATE" htable "KEYS(HEAD)")
                   if TBCRC = Ø then
                      do
                       headoff = 'NO'
                       hcreator = 'YES'
                      end
                   else
                      do
                       headoff = 'YES'
                      end
                  end
               signal on failure name trap
              end
/* MODTRACE must occur after the setting of ISPFENV                  */
           call modtrace 'START' sigl
/* Start-up message (if batch)                                       */
           startmsg = execname 'started' date() time() 'on' lpar
           if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
              headoff = 'NO' then
              do
               jobname = mvsvar('SYMDEF','JOBNAME')
               jobinfo = jobinfo()
               parse var jobinfo jobtype jobnum .
               say jobname center(' 'startmsg' ',61,'-') jobtype jobnum
               say
               if ISPRC = -3 then
                  do
                   call saydd msgdd 1 'ISPF ISPQRY module not found,',
                                 'ISPQRY is usually in the LINKLST'
                   call rcexit 20 'ISPF ISPQRY module is missing'
                  end
/* If MSGDD is provided, write the STARTMSG and SYSEXEC DSN to MSGDD */
               if msgdd <> '' then
                  do
                   call ddcheck msgdd
                   call saydd msgdd 1 startmsg
                   call ddcheck 'SYSEXEC'
                   call saydd msgdd 0 execname 'loaded from' sysdsname
/* If there are PARMS, write them to the MSGDD                       */
                   if parms <> '' then
                      call saydd msgdd 0 'Parms:' parms
/* If there is a STEPLIB, write the STEPLIB DSN MSGDD                */
                   if listdsi('STEPLIB' 'FILE') = 0 then
                      do
                       steplibs = dddsns('STEPLIB')
                       call saydd msgdd 0 'STEPLIB executables loaded',
                           'from' word(dddsns,1)
                       if dddsns('STEPLIB') > 1 then
                          do
                           do stl=2 to steplibs
                              call saydd msgdd 0 copies(' ',31),
                                   word(dddsns,stl)
                           end
                          end
                      end
                  end
              end
/* If foreground, save ZFKA and turn off the FKA display             */
           else
              do
               fkaset = 'OFF'
               call ispwrap "VGET (ZFKA) PROFILE"
               if zfka <> 'OFF' & tsoenv = 'FORE' then
                  do
                   fkaset = zfka
                   fkacmd = 'FKA OFF'
                   call ispwrap "CONTROL DISPLAY SAVE"
                 call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKACMD)"
                   call ispwrap "CONTROL DISPLAY RESTORE"
                  end
              end
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return
/* STDEXIT  - Standard Exit logic                                    */
/* ENDTIME  - Elapsed time                                           */
/* Note: Caller must set KEEPSTACK if the stack is valid             */
 stdexit: module = 'STDEXIT'
          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 endtime
          endmsg = execname 'ended' date() time() format(endtime,,1)
/* if MAXRC is greater then EXITRC then set EXITRC to MAXRC          */
          if MAXRC > EXITRC then EXITRC = MAXRC
          endmsg = endmsg 'on' lpar 'RC='EXITRC
          if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
             headoff = 'NO' then
             do
              say
              say jobname center(' 'endmsg' ',61,'-') jobtype jobnum
/* Make sure this isn't a MSGDD missing error then log to MSGDD      */
              if msgdd <> '' & subword(zedlmsg,9,1) <> msgdd then
                 do
                call saydd msgdd 1 execname 'ran in' endtime 'seconds'
                  call saydd msgdd Ø endmsg
                 end
             end
/* If foreground, reset the FKA if necessary                         */
          else
             do
              if fkaset <> 'OFF' then
                 do
                  fkafix = 'FKA'
                  call ispwrap "CONTROL DISPLAY SAVE"
                call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKAFIX)"
                  if fkaset = 'SHORT' then
                     call ispwrap "DISPLAY PANEL(ISPBLANK)",
                                  "COMMAND(FKAFIX)"
                 call ispwrap "CONTROL DISPLAY RESTORE"
                end
            end
/* Clean up the temporary HEADING table                              */
          if ispfenv = 'YES' & hcreator = 'YES' then
             call ispwrap "TBEND" htable
/* Remove STDEXIT and MAINLINE Parentage Stack entries, if there     */
          call modtrace 'STOP' sigl
          if queued() > Ø then pull . . module . sigl . sparms
          if queued() > Ø then pull . . module . sigl . sparms
          if tsoenv = 'FORE' & queued() > Ø & keepstack = 'NO' then
             pull . . module . sigl . sparms
/* if the Parentage Stack is not empty, display its contents         */
          if queued() > Ø & keepstack = 'NO' then
             do
              say queued() 'Leftover Parentage Stack Entries:'
              say
              do queued()
                 pull stackundo
                 say stackundo
              end
              EXITRC = 1
             end
/* Exit                                                              */
          exit(EXITRC)
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/* ZEDLMSG  - The long message variable                              */
 msg: module = 'MSG'
      parse arg zedlmsg
      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
/* If this is background or OMVS use SAY                             */
      if tsoenv = 'BACK' | execenv = 'OMVS' then
         say zedlmsg
      else
/* If this is foreground and ISPF is available, use SETMSG           */
         do
          if ispfenv = 'YES' then
/* Does not call ISPWRAP to avoid obscuring error message modules    */
             address ISPEXEC "SETMSG MSG(ISRZØØØ)"
          else
             say zedlmsg
         end
      pull tracelvl . module . sigl . sparms
      call modtrace 'STOP' sigl
      interpret 'trace' tracelvl
      return
/* DDCHECK  - Determine if a required DD is allocated                */
/* DD       - DDNAME to confirm                                      */
 ddcheck: module = 'DDCHECK'
          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 dd
          dderrmsg = 'OK'
          LRC = listdsi(dd "FILE")
/* Allow sysreason=3 to verify SYSOUT DD statements                  */
          if LRC <> Ø & strip(sysreason,'L',Ø) <> 3 then
             do
              dderrmsg = errmsg(sigl 'Required DD' dd 'is missing')
              call rcexit LRC dderrmsg sysmsglvl2
             end
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return
/* DDLIST   - Returns number of DDs and populates DDLIST variable    */
/* N/A      - None                                                   */
 ddlist: module = 'DDLIST'
         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
/* Trap the output from the LISTA STATUS command                     */
         call outtrap 'lines.'
         address TSO "LISTALC STATUS"
         call outtrap 'off'
         ddnum = Ø
/* Parse out the DDNAMEs and concatenate into a list                 */
         ddlist = ''
         do ddl=1 to lines.Ø
            if words(lines.ddl) = 2 then
               do
                parse upper var lines.ddl ddname .
                ddlist = ddlist ddname
                ddnum = ddnum + 1
               end
            else
               do
                iterate
               end
         end
/* Return the number of DDs                                          */
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return ddnum
/* DDDSNS   - Returns number of DSNs in a DD and populates DDDSNS    */
/* TARGDD   - DD to return DSNs for                                   */
 dddsns: module = 'DDDSNS'
         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 targdd
         if targdd = '' then call rcexit 77 'DD missing for DDDSNS'
/* Trap the output from the LISTA STATUS command                      */
         x = outtrap('lines.')
         address TSO "LISTALC STATUS"
         dsnnum = Ø
         ddname = '$DDNAME$'
/* Parse out the DDNAMEs, locate the target DD and concatentate DSNs */
         do ddd=1 to lines.Ø
            select
               when words(lines.ddd) = 1 & targdd = ddname &,
                    lines.ddd <> 'KEEP' then
                    dddsns = dddsns strip(lines.ddd)
               when words(lines.ddd) = 1 & strip(lines.ddd),
                    <> 'KEEP' then
                    dddsn.ddd = strip(lines.ddd)
               when words(lines.ddd) = 2 then
                    do
                     parse upper var lines.ddd ddname .
                     if targdd = ddname then
                        do
                         fdsn = ddd - 1
                         dddsns = lines.fdsn
                        end
                    end
               otherwise iterate
            end
         end
/* Get the last DD                                                    */
         ddnum = ddlist()
         lastdd = word(ddlist,ddnum)
/* Remove the last DSN from the list if not the last DD or SYSEXEC   */
         if targdd <> 'SYSEXEC' & targdd <> lastdd then
            do
             dsnnum = words(dddsns) - 1
             dddsns = subword(dddsns,1,dsnnum)
            end
/* Return the number of DSNs in the DD                                */
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return dsnnum
/* TSOTRAP  - Capture the output from a TSO command in a stem        */
/* VALIDRC  - Optional valid RC, defaults to zero                     */
/* TSOPARM  - Valid TSO command                                       */
```

```
 tsotrap: module = 'TSOTRAP'
          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 tsoparm
/* If the optional valid_rc parm is present use it, if not assume Ø  */
          parse var tsoparm valid_rc tso_cmd
          if datatype(valid_rc,'W') = Ø then
             do
              valid_rc = Ø
              tso_cmd = tsoparm
             end
          call outtrap 'tsoout.'
          tsoline = sigl
          address TSO tso_cmd
          CRC = RC
          call outtrap 'off'
/* If RC = Ø then return                                             */
          if CRC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return CRC
             end
          else
             do
              trapmsg = center(' TSO Command Error Trap ',78,'-')
              terrmsg = errmsg(sigl 'TSO Command:')
/* If RC <> Ø then format output depending on environment            */
              if tsoenv = 'BACK' | execenv = 'OMVS' then
                 do
                  say trapmsg
                  do c=1 to tsoout.Ø
                     say tsoout.c
                  end
                  say trapmsg
                  call rcexit CRC terrmsg tso_cmd
                 end
              else
/* If this is foreground and ISPF is available, use the ISPF LOG     */
                 do
                  if ispfenv = 'YES' then
                     do
                      zedlmsg = trapmsg
/* Does not call ISPWRAP to avoid obscuring error message modules    */
                      address ISPEXEC "LOG MSG(ISRZØØØ)"
                      do c=1 to tsoout.Ø
                         zedlmsg = tsoout.c
                         address ISPEXEC "LOG MSG(ISRZØØØ)"
                      end
                      zedlmsg = trapmsg
                      address ISPEXEC "LOG MSG(ISRZ000)"
                      call rcexit CRC terrmsg tso_cmd,
                          ' see the ISPF Log (Option 7.5) for details'
                     end
                 else
                     do
                      say trapmsg
                      do c=1 to tsoout.0
                         say tsoout.c
                      end
                      say trapmsg
                      call rcexit CRC terrmsg tso_cmd
                     end
                end
           end
/* WAIT     - Wait for a specified number of seconds                */
/* SECONDS  - Number of seconds to wait                             */
/* WMODE    - Use any value to stop printing batch wait messages    */
 wait: module = 'WAIT'
       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 seconds wmode
       if datatype(seconds,'W') = 0 then seconds = 10
       RC = syscalls('ON')
/* If foreground ISPF lock the screen                               */
       if tsoenv = 'FORE' & ispfenv = 'YES' then
          call lock seconds 'second wait was requested'
/* If background, report the wait time                              */
       if tsoenv = 'BACK' & wmode = '' then
          call saydd msgdd 0 seconds 'second wait was requested'
/* Call USS SLEEP                                                   */
       address SYSCALL "SLEEP" seconds
/* If foreground ISPF lock the screen                               */
       if tsoenv = 'FORE' & ispfenv = 'YES' then
          call unlock
       RC = syscalls('OFF')
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return
/* 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) <> 0 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    */
/* the message                                                       */
        msgb = 1
        if msglines > 0 then
           do msgb=1 to msglines
              msgline.msgb = ' '
           end
/* If the linesize is too long break it into multiple lines and      */
/* create continuation records                                       */
        msgm = msgb
        if length(message) > 60 & substr(message,1,2) <> '@@' then
           do
            messst = lastpos(' ',message,60)
            messseg = substr(message,1,messst)
            msgline.msgm = date() time() strip(messseg)
            message = strip(delstr(message,1,messst))
            do while length(message) > 0
               msgm = msgm + 1
               if length(message) > 55 then
                  messst = lastpos(' ',message,55)
               if messst > 0 then
                  messseg = substr(message,1,messst)
               else
                  messseg = substr(message,1,length(message))
               msgline.msgm = date() time() 'CONT:' strip(messseg)
               message = strip(delstr(message,1,length(messseg)))
            end
           end
        else
```

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

Robert Zenuk
Systems Programmer (USA)

CICS news

IBM has announced CICS Interdependency Analyzer for z/OS Version 1.3, which identifies resources used by CICS transactions and the relationship between them. The product also reports on DB2, IMS, and WebSphere MQ resources that are used by CICS. The main resources that are identified include those associated with transactions, programs, BMS maps, files, temporary storage queues, transient data queues, 3270 Bridge facility, Web Services, CorbaServer, and Enterprise JavaBeans.

The information can be used to improve the efficiency of CICS applications. It is also designed to help speed CICS application migration and reuse, and to increase CICS system availability.

Affinities data that is captured by the Transaction Affinities utility in CICS can be loaded into DB2 tables for analysis. The query interface has been updated for affinities. Sample SQL queries allow resource comparisons on data in DB2 tables. It also collects Task Control Block data, which is used in assessing threadsafe aspects of CICS-DB2 programs.

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

* * *

Fujitsu Software has announced NeoKicks Version 1.0, which allows CICS users to migrate to ASP .Net applications. This, they claim, delivers greatly increased agility to IT groups by lowering platform maintenance costs, giving interfaces new life as ASP.NET Web applications or Windows Forms client applications, and integrating with Visual Studio .NET for much higher developer productivity.

The product migrates the CICS code to the .NET environment, allowing it to benefit from tools such as Visual Studio .NET for maintenance and RAD, as well as placing the latest technologies fully within the reach of the migrated CICS applications. It also transforms the CICS BMS screens into ASP.NET Web pages (Web Forms) or optionally Windows Forms.

NeoKicks is available now in a limited membership Early Release Program.

For further information contact:
Fujitsu Software, 1250 E Arques Avenue, Sunnyvale, CA 94085, USA.
Tel: (408) 746 6300.
URL: http://www.adtools.com/products/windows/neokicks.html.

* * *

Acucorp has announced support for IBM TXSeries for Multiplatforms. This support, available as part of the *extend* suite of technologies, allows sites to use IBM's distributed environments and CICS.

Companies can build new applications on distributed platforms that work with IBM's TXSeries and Acucorp's ACUCOBOL-GT development system (part of *extend*). Sites can use IBM's and Acucorp's technologies for new CICS development. Organizations that wish to transition host systems can, they claim, simplify the process by utilizing the robust, secure, and scalable CICS facilities available in TXSeries – including familiar APIs, a common program structure, and interoperability with other CICS environments.

For further information contact:
Acucorp, 8515 Miralani Drive, San Diego, CA 92126-4352, USA.
Tel: (858) 689 4500.
URL: http://www.acucorp.com/company/press/releases/2004/2004_pr_10.php.