230

CICS update

January 2005

In this issue

A xephon PUBLICATION

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

Colin Smith
E-mail: info@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.

Printed in England.

Identifying DB2 in-doubt units of work at CICS shutdown

The DB2UOWS REXX EXEC is designed to identify any DB2 in-doubt units of work at CICS shutdown. This program is intended to be added to the end of the CICS PROC as a step after DFHSIP. The program reads the CICS SYSIN and scrapes out the INITPARM for DB2 and the APPLID to format a valid DB2 command to display all in-doubt units of work for that particular CICS region. If any in-doubts are found, the messages are written to the console so that automation can pick them up. This can help avoid a CICS INIT start if any in-doubts are found. A CICS INIT start while in-doubt units of work exist can cause database corruption. Sample JCL is included in the program comments.

DB2UOWS REXX EXEC

```
/*********************************************************************/
/*                              REXX                                 */
/*********************************************************************/
/* Purpose: Check for DB2 in-doubt units of work at CICS shutdown    */
/*-------------------------------------------------------------------*/
/* Syntax:  db2uows                                                  */
/*-------------------------------------------------------------------*/
/* Parms: N/A        - N/A                                           */
/*                                                                   */
/* Notes: Add as a step at the end of the CICS PROC                  */
/*        Depends on finding the CICS INITPARM DFHD2INI= value in    */
/*        SYSIN parms.  The CICSPARM DD must point to the correct    */
/*        CICS SYSIN.                                                */
/*                                                                   */
/*        Sample execution JCL:                                      */
/*                                                                   */
/*        //DB2UOWS  EXEC PGM=IKJEFT01,PARM='DB2UOWS'               */
/*        //STEPLIB  DD   DSN=db2.dsnload,DISP=SHR                  */
/*        //SYSEXEC  DD   DSN=rexx.exec.pds,DISP=SHR                */
/*        //CICSPARM DD   DSN=cics.sysin.pds,DISP=SHR               */
/*        //SYSTSPRT DD   SYSOUT=*                                  */
/*        //SYSTSIN  DD   DUMMY                                     */
/*                                                                   */
/*********************************************************************/
```

```
/*                       Change Log                                    */
/*********************************************************************/
/* Ensure all required DDs are present                               */
/*********************************************************************/
EXITRC = listdsi('STEPLIB' 'FILE')
if EXITRC <> Ø then
    do
     say 'STEPLIB is missing RC='EXITRC
     signal shutdown
    end
EXITRC = listdsi('SYSEXEC' 'FILE')
if EXITRC <> Ø then
    do
     say 'SYSEXEC is missing RC='EXITRC
     signal shutdown
    end
EXITRC = listdsi('CICSPARM' 'FILE')
if EXITRC <> Ø then
    do
     say 'CICSPARM is missing RC='EXITRC
     signal shutdown
    end
/*********************************************************************/
/* Read the CICS SYSIN dataset                                       */
/*********************************************************************/
"EXECIO * DISKR CICSPARM (STEM CICSPARM. FINIS"
EXITRC = RC
if EXITRC <> Ø then
    do
     say 'EXECIO error on CICSPARM RC='EXITRC
     signal shutdown
    end
/*********************************************************************/
/* Parse the CICS SYSIN looking for the db2ssid                      */
/*********************************************************************/
do i=1 to cicsparm.Ø
    select
      when pos("DFHD2INI='",cicsparm.i) <> Ø then
           parse var cicsparm.i . "DFHD2INI='" db2ssid "'" .
      when pos("APPLID=",cicsparm.i) <> Ø then
           parse var cicsparm.i "APPLID=" applid .
      otherwise nop
    end
end
/*********************************************************************/
/* Build the DB2 DISPLAY THREAD(applid) TYPE(INDOUBT) command        */
/*********************************************************************/
db2cmd = "-DISPLAY THREAD("applid") TYPE(INDOUBT)"
/*********************************************************************/
```

```
/* Queue the DB2 command                                              */
/*********************************************************************/
queue db2cmd
queue "END"
/*********************************************************************/
/* Report what the EXEC will do                                      */
/*********************************************************************/
say 'Checking' db2ssid 'for indoubts from' applid
say
say 'Issuing DB2 command:' db2cmd
say
/*********************************************************************/
/* Outtrap and execute the command                                   */
/*********************************************************************/
call outtrap indoubts.
"DSN SYSTEM("db2ssid")"
EXITRC = RC
if EXITRC <> 0 then say 'DB2 command error RC='EXITRC
/*********************************************************************/
/* Parse DB2 command output                                          */
/*********************************************************************/
do o=1 to indoubts.0
    select
/*********************************************************************/
/* Exclude all extraneous lines                                      */
/*********************************************************************/
      when word(indoubts.o,1) = 'DSNV401I'  then iterate
      when word(indoubts.o,1) = 'DSN9022I'  then iterate
/*********************************************************************/
/* Print the expected lines                                          */
/*********************************************************************/
/* GOOD response RC=0                                                */
/*********************************************************************/
      when word(indoubts.o,1) = 'DSNV412I' then
           do
           "SEND" "'"strip(indoubts.o,'T')"'"
            say strip(indoubts.o,'T')
            EXITRC = 0
           end
/*********************************************************************/
/* "BAD" response RC=20                                              */
/*********************************************************************/
      when word(indoubts.o,1) = 'DSNV406I' then
           do
           "SEND" "'"strip(indoubts.o,'T')"'"
            say strip(indoubts.o,'T')
            EXITRC = 20
           end
/*********************************************************************/
/* Print anything unexpected                                         */
```

```
/*********************************************************************/
      otherwise say strip(indoubts.o,'T')
    end
end
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
shutdown: nop
          exit(EXITRC)
```

Robert Zenuk
Systems Programmer (USA)

Get administrative and operational CICS parameters

When you have a large number of CICS regions to manage, and especially long-running regions, you must make sure that the CICS parameters are correct for a restart. So, you need to select, one-by-one, every CICS region and check the parameters that come from two sources – values (administrative) that are coded in DFHSIT and any overrides, and dynamic (current) values that might have been updated since the last start-up by any operator, or even by yourself a few weeks previously.

CICSSNAP finds out every CICS region running in your z/OS image (no CICS definitions required, only cross-memory) and, for each CICS region, it retrieves for each parameter its administrative value and specifies whether it comes from the SIT or overrides, and the current value in memory.

An example of the type of report it produces is:

```
CICS    SitParm    Admin   Over   Current
=========================================
CICSP7  MXT              40 No           52
CICSP7  DSALIM      8388608 No      8388608
CICSP7  EDSALIM  188743680 No    188743680
CICSP7  ICV           1000 No         1000
```

```
CICSP7  ICVTSD         500 No          500
CICSP7  PRTYAGE      32768 No        32768
CICSDX  MXT             32 No           32
CICSDX  DSALIM     8388608 YES     8388608
CICSDX  EDSALIM   89128960 YES    89128960
CICSDX  ICV           1000 No         1500
CICSDX  ICVTSD         500 No          500
```

Once CICSSNAP finds the CICS regions running in z/OS, it gets, in AR MODE, the anchor block address of each CICS domain related to the parameter value required (PA domain for SIT, SM domain for DSALIM, XM, etc). Then it retrieves the parameter value from these domains with a particular offset. Offsets to the current parameter value can easily be found with an SDUMP and a good IPCS VERBX. DFHSIT (by that I mean SIT merged with overrides) is simply pointed to by a PA domain anchor as well as the overrides table.

CICSSNAP shows only a few parameters (see TABPARM in the code), but you can update the code to re-generate the entire DFHSIT input stream with the current values in memory, and, for example, get rid of the overrides, rebuilding a complete SIT repository for all CICS regions.

CICSSNAP

```
*============================================================*
*  CICSSNAP :                                                *
*  Requirements : APF                                        *
*  Function     : Find out every CICS regions running within*
*                 z/OS, Establish XM mode and retrieve PARMs*
*                 Current Value (from CICS domain) and      *
*                 administrative values (from SIT and overri*
*                 de)                                       *
*  Status       : Test ok CTS 2.2                            *
*  Note         : Recovery routine can be coded              *
*============================================================*
CICSSNAP TITLE 'XM CICS SNAP Memory'
CICSSNAP CSECT
CICSSNAP AMODE 31
CICSSNAP RMODE 24
NBKXMAX   EQU   64                      Maximum number of CICSs
NBMAXPARM EQU   20                      Max Parms
NBREPORTL EQU   80                      Output RecLength
         BAKR  R14,0                    Save
```

```
         LR    R12,R15                    using r12
         USING CICSSNAP,R12               Addressability
*=============================================================*
         BAS  R14,INIT00                  Initialization routine
         BAS  R14,FINDKIX                 Scan ASCB/TCB/RB in AR mode
         BAS  R14,FORMAT                  Format CICS parms routine
         CLOSE OUT                        Close Output and
         PR                               go.
*=============================================================*
* INIT00 : Open output and Getmain TABCICS  (CICSLIST)
*=============================================================*
INIT00   EQU  *
         BAKR 14,0                        Save
         OPEN (OUT,OUTPUT)                Open Sysout
         GETMAIN RU,LV=24*NBKXMAX         Getmain for TABCICS
         ST   R1,TBKX                     Store TABCICS Pointer
         ST   R1,TBKXNEXT                 Store Next pointer to top
         PR                               Return
*=============================================================*
* FORMAT : Process each CICS in TABCICS
*        : Call the Parm format routine
*=============================================================*
FORMAT   EQU  *
         BAKR  14,0                       Save
         L    R1,TBKX                     ReLoad TABCICS addr
         LR    R11,R1                     CICSLIST in R11
         USING KXLIST,R11                 Addressability
         GETMAIN RU,LV=NBREPORTL          GeTmain output storage
         LR    R10,R1                     In R10
         PUT   OUT,LINE00                 Write Header
         PUT   OUT,LINE01                 Write Interline
FMT00    EQU   *
         MVC   ALET(4),KXALET             Get alert from CICSLIST
         MODESET KEY=ZERO,MODE=SUP        MODESET
         LAM   8,8,ALET                   Load alert for R8
         LAM   7,7,ALET                   for R7
         LAM   6,6,ALET                   for R7
         L     R8,KXDFHPA                 Reload PA domain anchor addr
         SAC   512                        Switch in AR mode
         L     R8,X'2C'(,R8)              To DFHSIT
         LR    R7,R8                      Same for R7
FMT01    EQU   *
         MVC   0(8,R10),KXNAME            CICS Name in output zone
         LR    R0,R11                     R0=>CICSLIST
         LR    R1,R10                     R1=>Output storage
         L     R15,=A(FMTPARM)            Load Parm format routine
         BASR  R14,R15                    Call
         SAC    0                         Back to primary mode
         LA     R11,36(R11)               Next CICS to process
         CLC   0(4,R11),=F'0'             End of TABCICS reached ?
```

```
         BNE    FMTØØ                      NO=> Go for it
FMT99    EQU   *
         PR                                Return
*=============================================================*
* FINKIX
*       1 Process STC and BATCH only
*       2 Chain TCB and RB till I find CDE=DFHSIP
*       3 Get KE anchor via TCBCAUF
*       3 Save in a CICS TABLE row
*=============================================================*
FINDKIX  EQU  *
         BAKR 14,Ø                        Save
         L     R2,CVTPTR                  To CVT
         L     R2,CVTASVT-CVT(,R2)        TO ASVT
         LH    R4,ASVTMAXU-ASVT+2(,R2)    R4=> number of ASVT entries
         LA    R8,ASVTENTY-ASVT(,R2)      R8=> First asvt  ent.
ASVTLOOP EQU *
         TM    Ø(R8),ASVTAVAI             Asid available
         BO    ASVTNEXT                   Get nextone
         L     R7,Ø(,R8)                  Get ASCB ADDR
         LA    R7,Ø(R7)                   ADDR
         LTR   R7,R7                      ASCB NULL
         BZ    ASVTNEXT                   Check next
         L     R2,ASCBOUCB-ASCB(R7)       Get OUCB addr
         LA    R2,Ø(R2)                   Clear HOB
         LTR   R2,R2                      any OUCB ?
         BZ    ASVTNEXT                   Next ASVT if no OUCB
         TM    OUCBSFL-OUCB(R2),OUCBINV   Is this OUCB valid ?
         BO    ASVTNEXT                   No, skip it too.
         L     R3,ASCBJBNI-ASCB(R7)       Jobname addr
         LTR   R3,R3                      If not a JOB => STC orTSU
         BZ    NOJOB                      Find out which
         L     R3,CHCSCBP-CHNAME(R3)      To cSCB
         MVC   JOBNAME(8),CHKEY-CHAIN(R3) Get Jobname
         B     IDENTIFY                   It that a CICS region ?
NOJOB  EQU *
         L     R3,ASCBCSCB-ASCB(R7)        CSCB ADDRESS
         LA    R3,Ø(R3)                    CLEAR TOP BYTE
         LTR   R3,R3                       ANY CSCB ?
         BZ    ASVTNEXT                    If no CSCB
         TM    OUCBYFL-OUCB(R2),OUCBSTT+OUCBMNT        START OR MOUNT ?
         BZ    ASVTNEXT                   Supposed to be TSU
         MVC   JOBNAME(8),CHCLS-CHAIN(R3)  Get Jobname
         B     IDENTIFY                   it that a CICS region
ASVTNEXT EQU *
         LA    R8,4(R8)                   Next entry
         BCT   R4,ASVTLOOP                Loop
         PR                               Exit
*=============================================================*
* Identify CICS regions                                       *
```

```
*=============================================================*
IDENTIFY EQU   *
         ST    R8,SAVASVT                    Save ASVT
         CLC   ASCBASID-ASCB(2,R7),=H'16'  No need to process
         BNH   ASVTNEXT                     system asids
         L     R1,ASCBASXB-ASCB(R7)         To ASXB
         L     R2,ASXBFTCB-ASXB(R1)         To first TCB
         LH    R3,ASXBTCBS-ASXB(R1)         Number of TCBs
         L     R1,(ASCBASSB-ASCB)(R7)       To ASSB
         MVC   STKN(8),(ASSBSTKN-ASSB)(R1) Get Stoken
         ST    R7,WASCB                      Store ASCB addr
         MODESET KEY=ZERO,MODE=SUP          Key0 adn get alet
         ALESERV ADD,ALET=ALET,STOKEN=STKN,CHKEAX=NO,                  *
               AL=WORKUNIT,ACCESS=PUBLIC
         LAM   6,6,ALET                     Load alert for r6
         LAM   7,7,ALET                     idem for r7
         LAM   8,8,ALET                     and r8
         LR    R8,R2                        Load first TCB
         SAC   512                          to  AR MODE
TCB00    EQU   *
         L     R7,0(0,R8)                   Top Rb
         CLI   RBSTAB1-RBBASIC(R7),RBFTPRB is it a PRB
         BNE   NEXTRB                       No, getnext
RB00     EQU   *
         L     R6,RBEPA-RBBASIC(0,R7)       To RBEPA
         CLM   R6,B'0111',=XL3'000000'      Is valid ?
         BE    NEXTRB                       No, go for next RB
         CLC   CDNAME-CDENTRY(8,R6),=CL8'DFHSIP'  is DFHSIP RB
         BNE   NEXTRB                       No, go for next RB
         B     CICS                         Yes, DFHSIP RB
NEXTRB   EQU   *
         L     R7,RBLINK-RBBASIC(0,R7)      Get next RB
         SLL   R7,8                         Clear top byte
         SRL   R7,8                         ND COMPARE WITH
         CR    R7,R8                        TCB ADDR
         BNE   RB00                                 CHECK NEXT RB
         L     R8,TCBTCB-TCB(,R8)           Next TCB
         LTR   R8,R8                        IS IT THE LAST  ?
         BZ    NOCICS                       YES, we are done
         B     TCB00                        Process next TCB
NOCICS   EQU   *
         SAC    0                           To primary mode
         ALESERV DELETE,ALET=ALET           Delete alert
         MODESET KEY=NZERO,MODE=PROB        Back to problem
         L     R8,SAVASVT                   restore ASVT
         B     ASVTNEXT                     Go for next asvt
CICS     EQU   *
         L     R8,TCBEXT2-TCB(,R8)          To TCBEXT2
         L     R8,TCBCAUF-TCBXTNT2(,R8)     To CAUF
         AH    R8,X'06'(,R8)                Skip vector
```

```
         LA    R8,X'1Ø'(,R8)               Skip the prefix
         L     R8,X'Ø4'(,R8)               to AFCS
         L     R6,X'Ø8'(,R8)               DFHKEKCB in R6
         L     R8,X'Ø8'(,R8)               to DFHKEKCB
         LA    R8,X'198'(,R8)              to DM PA
         L     R8,X'ØØ'(,R8)               to DFHPA
         L     R8,X'1Ø'(,R8)               to DFHPA anchor
         L     R1,TBKXNEXT                 Available row in TAB
         MVC   KXNAME-KXNAME(8,R1),JOBNAME Move Jobname in TABCICS
         MVC   KXASCB-KXNAME(4,R1),WASCB   Move ASCB addr
         MVC   KXALET-KXNAME(4,R1),ALET    Move ALET
         MVC   KXSTKN-KXNAME(4,R1),STKN    Move token
         ST    R8,KXDFHPA-KXNAME(,R1)      Move PA domain anchor addr
         LA    R8,X'1CC'(,R6)              to XM domain
         L     R8,X'ØØ'(,R8)               to DFHXM
         L     R8,X'1Ø'(,R8)               to DFHXMA anchor
         ST    R8,KXDFHXM-KXNAME(,R1)      Move XM domain anchor addr
         LA    R8,X'18Ø'(,R6)              to SM domain
         L     R8,X'ØØ'(,R8)               to DFHSM
         L     R8,X'1Ø'(,R8)               to DFHSMA anchor
         ST    R8,KXDFHSM-KXNAME(,R1)      Move SM domain anchor addr
         LA    R8,X'184'(,R6)              to DS domain
         L     R8,X'ØØ'(,R8)               to DFHDS
         L     R8,X'1Ø'(,R8)               To DFHDS anchor
         ST    R8,KXDFHDS-KXNAME(,R1)      Move DS domain anchor addr
         LA    R1,36(R1)                   To next CICS
         ST    R1,TBKXNEXT                 Store this "next cics" addr
         SAC    Ø                          Back to primary
         L     R8,SAVASVT                  restore ASVT
         B     ASVTNEXT                    Next ASVT entry
*============================================================*
* Format PARM   : RØ: KIXTAB
*               : R1: R1Ø Output Zone
*============================================================*
FMTPARM  EQU  *
         BAKR 14,Ø Save
         LA    R2,TABPARM                  Load tab parms addr
         LR    R11,RØ                      Use R11 for
         USING KXLIST,R11                  KXLIST addressability
         LR    R1Ø,R1                      Output zone in R1Ø
FMTPØØ   EQU   *
         L     R7,KXDFHPA                  To PA domain anchor
         L     R8,X'2C'(,R7)               To DFHSIT
         CLC   Ø(8,R2),=D'Ø'               Last parm to porcess
         BE    FMTP99                      Yes, return
FMTP1Ø   EQU   *
         MVC   8(8,R1Ø),Ø(R2)              PARM name to output
         A     R8,12(R2)                   To PARM value in DFHSIT
         LA    R4,4                        Max length
         SH    R4,8(R2)                    Max length - length in SIT
```

```
         XC    TEMP(4),TEMP              Clear TEMP
         LA    R5,TEMP                   Load TEMP addr
         AR    R5,R4                     To where we move
         LH    R1,8(R2)                  Reload PARM length in SIT
         BCTR  R1,Ø                      For execute
         EX    R1,MVCF1                  Move
         ICM   R7,B'1111',TEMP           Load R7 with the previous
         MVC   MASK(16),MASKØØ           Reload MASK for ED
         CVD   R7,PL8                    Pack value
         ED    MASK(16),PL8              Ed
         MVC   16(1Ø,R1Ø),MASK+6         move  Parm value to output
*
         LA    R1,8(R1Ø)                 PARM NAme addr
         L     RØ,KXDFHPA                Reload PA domain anchor
         L     R15,=V(CICSOSIT)          Call search-for-overrides
         BALR  R14,R15                   Routine
         MVC   27(4,R1Ø),Ø(R1)           Get either YES or No
         MVC   MASK(16),MASKØØ           Reload MASK for ED
*
         L     R4,2Ø(R2)                 Load displac. within KXLIST
         L     R8,Ø(R4,R11)              Load specific domain that
         AH    R8,16(R2)                 Correponds + Displ in domain
         LA    R4,4                      Max length
         SH    R4,1Ø(R2)                 Max length - length in Dom.
         XC    TEMP(4),TEMP              Clear TEMP
         LA    R5,TEMP                   Load TEMP addr
         AR    R5,R4                     To where we move
         LH    R1,1Ø(R2)                 Reload PARM length in Dom
         BCTR  R1,Ø                      For execute
         EX    R1,MVCF1                  Move
         ICM   R7,B'1111',TEMP           load R7 with the previous
         CVD   R7,PL8                    Pack
         ED    MASK(16),PL8              Edit
         MVC   32(1Ø,R1Ø),MASK+6         move  parm value
*
         SAC    Ø                        Back in primary
         PUT    OUT,Ø(R1Ø)               Write output
         SAC    512                      In AR
         LA     R2,24(R2)                Next PARM to process
         B      FMTPØØ                   process it.
FMTP99   EQU    *
         PR
         CNOP  Ø,4
MVCF1    MVC   Ø(Ø,R5),Ø(R8)             Move parm value
TEMP     DC   F'Ø'
*=============================================================*
* TABPARM : Name (8), length in SIT, length in domain
*         : Displacement in SIT (administrative value)
*         : Displacement in DOM (current value)
*         : Displacement in KXLIST where to obtain the
```

```
*         : right domain anchor pointer (PA,DS,XM,,)
*=============================================================*
TABPARM  DS   ØF
         DC   CL8'MXT',H'2',H'4',A(SITMXTSK-DFHSITBA),XL2'ØØC8'
         DC   A(KXDFHXM-KXNAME)
         DC   CL8'DSALIM',H'4',H'4',A(SITDSA-DFHSITBA),XL2'Ø118'
         DC   A(KXDFHSM-KXNAME)
         DC   CL8'EDSALIM',H'4',H'4',A(SITEDSA-DFHSITBA),XL2'Ø11C'
         DC   A(KXDFHSM-KXNAME)
         DC   CL8'ICV    ',H'4',H'4',A(SITICVAL-DFHSITBA),XL2'ØØ84'
         DC   A(KXDFHDS-KXNAME)
         DC   CL8'ICVTSD ',H'2',H'2',A(SITTSDTI-DFHSITBA),XL2'ØØ68'
         DC   A(KXDFHDS-KXNAME)
         DC   CL8'PRTYAGE',H'2',H'4',A(SITPMULT-DFHSITBA),XL2'ØØ1Ø'
         DC   A(KXDFHDS-KXNAME)
         DC   D'Ø'
*=============================================================*
* Working
*=============================================================*
         CNOP Ø,4
         DC   CL8'Working'
PL8      DC   PL8'Ø'
MASKØØ   DC   XL16'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø'
MASK     DC   XL16'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø'
TBKX     DC   F'Ø'
TBKXNEXT DC   F'Ø'
STKN     DC   D'Ø'
ALET     DC   F'Ø'
SAVASVT  DC   F'Ø'
WASCB    DC   F'Ø'
OUT      DCB  MACRF=PM,DSORG=PS,RECFM=FB,LRECL=8Ø,DDNAME=OUT
LINEØØ    DC   CL8Ø'CICS    SitParm    Admin   Over   Current '
LINEØ1    DC   CL8Ø'========================================== '
JOBNAME   DC   CL16' '
          LTORG
          CNOP  Ø,4
KXLIST    DSECT
KXNAME    DS   CL8                              CICS Job or Stcname
KXASCB    DS   F                                CICS ASCB addr
KXALET    DS   F                                Alet for specific CICS
KXSTKN    DS   F                                TOKEN for XM
KXDFHPA   DS   F                                PA Domain Anchor Addr
KXDFHSM   DS   F                                SM Domain Anchor Addr
KXDFHXM   DS   F                                XM Domain Anchor Addr
KXDFHDS   DS   F                                DS Domain Anchor Addr
         IHARB LIST=YES
         IHACDE
         DFHSIT TYPE=DSECT
         CVT DSECT=YES
         IEECHAIN
```

```
         IHAASCB
         IHAASSB
         IAZJSAB
         IHAASXB
         IKJTCB
         IRAOUCB
         IHAASVT LIST=YES
         COPY  EQU
         END
*============================================================*
* CICSOSIT : Search overides for SIT PARM                    *
* Input    : RØ : PARM DOMAIN ANCHOR ADDR                    *
*          : R1 : Addr to PARM NAME                          *
*============================================================*
CICSOSIT CSECT
CICSOSIT CSECT
         COPY  EQU
         BAKR  R14,Ø                      Save
         LR    R12,R15                    Use R12
         USING CICSOSIT,R12
         LR    R6,RØ                      Get PA domain anchor addr
OSITØØ   EQU   *
         L     R7,X'1C'(,R6)              To override PARMS
         LA    R5,X'1Ø'(,R7)              To override length
         AH    R5,X'1Ø'(,R7)              Compute to end override
         LA    R7,X'1Ø'(,R7)              To override length
OSIT1Ø   EQU   *
         LR    R3,R1                      R3<= PARM name Addr
         TRT   Ø(16,R3),CHARTAB           First blank
         SR    R1,R3                      Compute length
         BCTR  R1,Ø                       -1 for execute CLC
OSIT2Ø   EQU   *
         EX    R1,CLC                     Found it
         BE    OSIT5Ø                     Yes,
         LA    R7,1(,R7)                  No,+1 to search
         CR    R7,R5                      If you reached the end
         BH    OSIT9Ø                     then this PARM is not found
         B     OSIT2Ø                     Loop
OSIT5Ø   EQU   *
         MVC   CL16(16),=CL16'YES'        PARM found in overrides
         B     OSITEND
OSIT9Ø   EQU   *
         MVC   CL16(8),=CL8'No '          PARM not found in overrides
OSITEND  EQU   *
         LA    R1,CL16                    R1 for return
         PR
         CNOP  Ø,4
CLC      CLC   Ø(Ø,R7),Ø(R3)
CHARTAB  DS   ØXL256
         DC   64X'ØØ',X'4Ø',191X'ØØ'
```

```
CL16     DC   CL16' '
         END
```

David Harou
Systems Programmer (France)

CICSDUMP routine – part 2

This month we conclude the code that allows you to invoke IPCS for CICS SVC dump datasets from the ISPF 3.4 DSLIST screen as a line command.

```
/*********************************************************************/
/* Generate the TEMPDSN                                              */
/*********************************************************************/
          tempdsn = uniqdsn()'('tempmem')'
/*********************************************************************/
/* ALLOCATE the TEMP DD and member                                   */
/*********************************************************************/
          call tsotrap "ALLOC F("tempdd") DA("qdsn(tempdsn)") NEW",
                       "LRECL(8Ø) BLKS(Ø) DIR(1) SPACE(1) CATALOG",
                       "UNIT("@sysda") RECFM(F B)"
/*********************************************************************/
/* Write the STEM to the TEMP DD                                     */
/*********************************************************************/
          call tsotrap 'EXECIO * DISKW' tempdd '(STEM' tempmem'. FINIS'
/*********************************************************************/
/* DROP the stem variable                                            */
/*********************************************************************/
          interpret 'drop' tempmem'.'
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return tempdd
/*********** @REFRESH END   TEMPMEM  2ØØ4/Ø9/Ø1 17:2Ø:19 *************/
/*********** @REFRESH BEGIN ISPWRAP  2ØØ2/Ø9/11 Ø1:11:43 *************/
/* ISPWRAP  - Wrapper for ISPF commands                              */
/*-------------------------------------------------------------------*/
/* VALIDRC  - Optional valid RC from the ISPF command, defaults to Ø */
/* ISPPARM  - Valid ISPF command                                     */
/*********************************************************************/
 ispwrap: module = 'ISPWRAP'
          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 ispparm
          zerrlm = 'NO ZERRLM'
/*********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume Ø  */
/*********************************************************************/
          parse var ispparm valid_rc isp_cmd
          if datatype(valid_rc,'W') = Ø then
             do
              valid_rc = Ø
              isp_cmd = ispparm
             end
          address ISPEXEC isp_cmd
          IRC = RC
/*********************************************************************/
/* If RC = Ø then return                                             */
/*********************************************************************/
          if IRC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return IRC
             end
          else
             do
              perrmsg = errmsg(sigl 'ISPF Command:')
              call rcexit IRC perrmsg isp_cmd strip(zerrlm)
             end
/*********** @REFRESH END   ISPWRAP  2ØØ2/Ø9/11 Ø1:11:43 *************/
/*********** @REFRESH BEGIN ISRWRAP  2ØØ2/Ø9/11 Ø1:12:34 *************/
/* ISRWRAP  - Wrapper for ISPF Edit commands                         */
/*-------------------------------------------------------------------*/
/* VALIDRC  - Optional valid RC from the ISPF command, defaults to Ø */
/* ISRPARM  - Valid ISPF Edit command                                */
/*********************************************************************/
 isrwrap: module = 'ISRWRAP'
          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 isrparm
/*********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume Ø  */
/*********************************************************************/
          parse var isrparm valid_rc isr_cmd
          if datatype(valid_rc,'W') = Ø then
             do
              valid_rc = Ø
```

```
             isr_cmd = isrparm
            end
          parse var isr_cmd isr_verb .
          address ISREDIT isr_cmd
          ERC = RC
/*********************************************************************/
/* If RC = 0 then return                                             */
/*********************************************************************/
          if ERC <= valid_rc then
             do
              pull tracelvl . module . sigl . sparms
              call modtrace 'STOP' sigl
              interpret 'trace' tracelvl
              return ERC
             end
          else
             do
              if isr_verb = 'MACRO' & ERC = 20 then
                 do
                  call rcexit ERC 'is an Edit Macro and not valid to',
                              'run outside of ISPF Edit'
                 end
              else
                 do
                  rerrmsg = errmsg(sigl 'ISPF Edit Command:')
                  call rcexit ERC rerrmsg isr_cmd
                 end
             end
/*********** @REFRESH END   ISRWRAP  2002/09/11 01:12:34 *************/
/*********** @REFRESH BEGIN TSOTRAP  2002/12/15 05:18:45 *************/
/* 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) <> 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 tsoparm
/*********************************************************************/
/* If the optional valid_rc parm is present use it, if not assume 0  */
/*********************************************************************/
          parse var tsoparm valid_rc tso_cmd
          if datatype(valid_rc,'W') = 0 then
             do
              valid_rc = 0
              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(ISRZØØØ)"
                      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.Ø
                      say tsoout.c
                   end
                   say trapmsg
                   call rcexit CRC terrmsg tso_cmd
                  end
               end
            end
/*********** @REFRESH END   TSOTRAP  2ØØ2/12/15 Ø5:18:45 *************/
/*********** @REFRESH BEGIN TSOQUIET 2ØØ3/Ø5/26 1Ø:22:58 *************/
/* TSOQUIET - Trap all output from a TSO command and ignore failures */
/*-------------------------------------------------------------------*/
/* TSOCMD   - TSO command to execute                                 */
/*********************************************************************/
 tsoquiet: module = 'TSOQUIET'
           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
/*********************************************************************/
/* Accept command to execute and throw away results                  */
/*********************************************************************/
           arg tsocmd
           call outtrap 'garbage.' Ø
           address TSO tsocmd
           call outtrap 'off'
/*********************************************************************/
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return
/*********** @REFRESH END   TSOQUIET 2ØØ3/Ø5/26 1Ø:22:58 *************/
/*********** @REFRESH BEGIN WAIT     2ØØ3/Ø5/18 Ø4:Ø3:43 *************/
/* 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) <> Ø 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
/*********** @REFRESH END   WAIT     2003/05/18 04:03:43 *************/
/*********** @REFRESH BEGIN SETBORD  2002/09/11 01:16:41 *************/
/* SETBORD  - Set the ISPF Popup active frame border color           */
/*-------------------------------------------------------------------*/
/* COLOR    - Colour for the Active Frame Border                     */
/*********************************************************************/
 setbord: module = 'SETBORD'
          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 color
/*********************************************************************/
/* Parse and validate colour                                         */
/*********************************************************************/
          if color = '' then color = 'YELLOW'
/*********************************************************************/
/* Build a temporary panel                                           */
/*********************************************************************/
          ispopt11.1=")BODY                          "
          ispopt11.2="%Command ===>_ZCMD        + "
          ispopt11.3=")INIT                          "
          ispopt11.4="&ZCMD = ' '                    "
```

```
          ispopt11.5="VGET (COLOR) SHARED         "
          ispopt11.6="&ZCOLOR = &COLOR            "
          ispopt11.7=".RESP = END                 "
          ispopt11.8=")END                        "
/*********************************************************************/
/* Allocate and load the Dynamic Panel                               */
/*********************************************************************/
          setdd = tempmem('ISPOPT11')
/*********************************************************************/
/* LIBDEF the DSN, VPUT @TFCOLOR and run CUAATTR                     */
/*********************************************************************/
          call ispwrap "LIBDEF ISPPLIB LIBRARY ID("setdd") STACK"
          call ispwrap "VPUT (COLOR) SHARED"
          call ispwrap "SELECT PGM(ISPOPT) PARM(ISPOPT11)"
          call ispwrap "LIBDEF ISPPLIB"
          call tsotrap "FREE F("setdd") DELETE"
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return
/*********** @REFRESH END   SETBORD  2002/09/11 01:16:41 *************/
/*********** @REFRESH BEGIN LOCK     2004/09/01 18:00:03 *************/
/* LOCK     - Put up a pop-up under foreground ISPF during long waits*/
/*-------------------------------------------------------------------*/
/* LOCKMSG  - Message for the pop-up screen                          */
/*********************************************************************/
 lock: module = 'LOCK'
       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 lockmsg
       if lockmsg = '' then lockmsg = 'Please be patient'
       if tsoenv = 'FORE' then
          do
/*********************************************************************/
/* Use the length of the lockmsg to determine the pop-up size        */
/*********************************************************************/
           if length(lockmsg) < 76 then
              locklen = length(lockmsg) + 2
           else
              locklen = 77
           if locklen <= 10 then locklen = 10
/*********************************************************************/
/* Build a temporary panel                                           */
/*********************************************************************/
           lock.1 = ")BODY EXPAND(//) WINDOW("locklen",1)"
           lock.2 = "%&LOCKMSG                    "
           lock.3 = ")END                         "
/*********************************************************************/
```

```
/* Lock the screen and put up a pop-up                               */
/*********************************************************************/
           call ispwrap "CONTROL DISPLAY LOCK"
           call popdyn 'LOCK' 8 execname 'Please be patient'
           lockpop = 'YES'
          end
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return
/*********** @REFRESH END   LOCK     2004/09/01 18:00:03 *************/
/*********** @REFRESH BEGIN UNLOCK   2003/10/18 09:33:19 *************/
/* UNLOCK   - Unlock from a popup under foreground ISPF              */
/*-------------------------------------------------------------------*/
/* PARM     - N/A                                                    */
/*********************************************************************/
 unlock: module = 'UNLOCK'
         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
         if tsoenv = 'FORE' then
             do
              if lockpop = 'YES' then
                 do
                  call ispwrap "REMPOP"
                  lockpop = 'NO'
                 end
              if popup = 'YES' then
                 do
                  call setbord 'BLUE'
                  call ispwrap "REMPOP"
                  popup = 'NO'
                 end
             end
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return
/*********** @REFRESH END   UNLOCK   2003/10/18 09:33:19 *************/
/*********** @REFRESH BEGIN PANDSN   2004/04/28 00:46:04 *************/
/* PANDSN   - Create a unique PDS(MEM) name for a dynamic panel      */
/*-------------------------------------------------------------------*/
/* PANEL    - Dynamic panel name                                     */
/*********************************************************************/
 pandsn: module = 'PANDSN'
         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 panel
         pandsn = uniqdsn()'('panel')'
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return pandsn
/*********** @REFRESH END   PANDSN   2004/04/28 00:46:04 *************/
/*********** @REFRESH BEGIN POPDYN   2002/09/11 01:15:11 *************/
/* POPDYN   - Addpop a Dynamic Panel                                 */
/*-------------------------------------------------------------------*/
/* DYN      - Dynamic panel name                                     */
/* DYNROW   - Default row for ADDPOP, defaults to 1                  */
/* DYNMSG   - ADDPOP Window title                                    */
/*********************************************************************/
 popdyn: module = 'POPDYN'
         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 dyn dynrow dynmsg
/*********************************************************************/
/* Set the default ADDPOP row location                               */
/*********************************************************************/
         if dynrow = '' then dynrow = 1
/*********************************************************************/
/* Set the default ADDPOP window title to the current exec name      */
/*********************************************************************/
         if dynmsg = '' then dynmsg = execname
/*********************************************************************/
/* Check whether the RETURN option is specified in the DYNMSG        */
/*********************************************************************/
         dynreturn = 'NO'
         if word(dynmsg,1) = 'RETURN' then
            parse var dynmsg dynreturn dynmsg
/*********************************************************************/
/* Allocate and load the Dynamic Panel                               */
/*********************************************************************/
         dyndd = tempmem(dyn)
/*********************************************************************/
/* LIBDEF the POPDYN panel                                           */
/*********************************************************************/
         call ispwrap "LIBDEF ISPPLIB LIBRARY ID("dyndd") STACK"
/*********************************************************************/
/* Change the Active Frame Colour                                    */
/*********************************************************************/
         call setbord 'YELLOW'
/*********************************************************************/
/* set the POPUP variable if this is not a LOCK request              */
/*********************************************************************/
         if dyn = 'LOCK' & popup = 'NO' then
```

```
            popup = 'NO'
         else
            popup = 'YES'
/*********************************************************************/
/* Put up the pop-up                                                 */
/*********************************************************************/
         zwinttl = dynmsg
         call ispwrap "ADDPOP ROW("dynrow")"
         DRC = ispwrap(8 "DISPLAY PANEL("dyn")")
         call ispwrap "LIBDEF ISPPLIB"
         call tsotrap "FREE F("dyndd") DELETE"
/*********************************************************************/
/* Change the Active Frame Colour                                    */
/*********************************************************************/
         call setbord 'BLUE'
/*********************************************************************/
/* Determine how to return                                           */
/*********************************************************************/
         if dynreturn = 'NO' then
            call rcexit DRC 'terminated by user'
         if dynreturn = 'RETURN' & DRC = 8 then
            do
             call ispwrap "REMPOP"
             popup = 'NO'
            end
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return DRC
/*********** @REFRESH END   POPDYN   2002/09/11 01:15:11 *************/
/*********** @REFRESH BEGIN SAYDD    2004/03/29 23:48:37 *************/
/* 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
/*********************************************************************/
/* Build print lines. Default strips and prefixes date and timestamp */
/* @BLANK - Blank line, no date and timestamp                        */
/* @      - No stripping, retains leading blanks                     */
/* @@     - No stripping, No date and timestamp                      */
/*********************************************************************/
```

```
           do
            select
               when message = '@BLANK@' then msgline.msgm = ' '
               when word(message,1) = '@' then
                    do
                     message = substr(message,2,length(message)-1)
                     msgline.msgm = date() time() message
                    end
               when substr(message,1,2) = '@@' then
                    do
                     message = substr(message,3,length(message)-2)
                     msgline.msgm = message
                    end
               otherwise msgline.msgm = date() time() strip(message)
            end
           end
/*********************************************************************/
/* If a number is provided, add that number of blank lines after     */
/* the message                                                       */
/*********************************************************************/
        if msglines > 0 then
           do msgt=1 to msglines
              msge = msgt + msgm
              msgline.msge = ' '
           end
/*********************************************************************/
/* Write the contents of the MSGLINE stem to the MSGDD               */
/*********************************************************************/
        call tsotrap "EXECIO * DISKW" msgdd "(STEM MSGLINE. FINIS"
        drop msgline. msgb msgt msge
        pull tracelvl . module . sigl . sparms
        call modtrace 'STOP' sigl
        interpret 'trace' tracelvl
        return
/*********** @REFRESH END   SAYDD    2004/03/29 23:48:37 *************/
/*********** @REFRESH BEGIN BRWSDD   2002/09/11 01:05:08 *************/
/* BRWSDD   - Invoke ISPF Browse on any DD                           */
/*-------------------------------------------------------------------*/
/* BRWSDD   - Any DD to browse                                       */
/*********************************************************************/
 brwsdd: module = 'BRWSDD'
         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 brwsdd
         if brwsdd = '' then call rcexit 90 'Browse DD missing'
         call ispwrap "LMINIT DATAID(DATAID) DDNAME("brwsdd")"
/*********************************************************************/
/* Browse the VIO dataset                                            */
```

```
/*********************************************************************/
         call ispwrap "BROWSE DATAID("dataid")"
/*********************************************************************/
/* FREE and DELETE the VIO dataset                                   */
/*********************************************************************/
         call ispwrap "LMFREE DATAID("dataid")"
         call tsotrap "FREE F("brwsdd")"
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return
/*********** @REFRESH END   BRWSDD   2002/09/11 01:05:08 *************/
/*********** @REFRESH BEGIN JOBINFO  2002/09/11 01:12:59 *************/
/* JOBINFO  - Get job related data from control blocks               */
/*-------------------------------------------------------------------*/
/* ITEM     - Optional item number desired, default is all           */
/*********************************************************************/
 jobinfo: module = 'JOBINFO'
          if wordpos(module,probe) <> 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',0)
          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
/*********** @REFRESH END   JOBINFO  2002/09/11 01:12:59 *************/
/*********** @REFRESH BEGIN STACK    2004/05/18 09:25:09 *************/
/* STACK    - UNLOAD, RELOAD or LIST the Stack                       */
/*-------------------------------------------------------------------*/
/* OPTION   - UNLOAD, RELOAD or LIST                                 */
/*********************************************************************/
 stack: arg stackopt
/*********************************************************************/
/* Unload the parentage stack to avoid display problems              */
/*********************************************************************/
        if stackopt = 'UNLOAD' | stackopt = 'LIST' then
           do deq=1 to queued()
              pull stackinfo
              tempq.deq = stackinfo
           end
/*********************************************************************/
/* List the stack                                                    */
/*********************************************************************/
        if stackopt = 'LIST' then
           do req=deq-1 to 1 by -1
              say tempq.req
           end
/*********************************************************************/
/* Reload the parentage stack                                        */
/*********************************************************************/
        if stackopt = 'RELOAD' | stackopt = 'LIST' then
           do req=deq-1 to 1 by -1
              push tempq.req
           end
        return
/*********** @REFRESH END   STACK    2004/05/18 09:25:09 *************/
/*********** @REFRESH BEGIN PTR      2002/07/13 15:45:36 *************/
/* PTR      - Pointer to a storage location                          */
/*-------------------------------------------------------------------*/
/* ARG(1)   - Storage Address                                        */
/*********************************************************************/
 ptr: return c2d(storage(d2x(arg(1)),4))
/*********** @REFRESH END   PTR      2002/07/13 15:45:36 *************/
/*********** @REFRESH BEGIN STG      2002/07/13 15:49:12 *************/
/* STG      - Return the data from a storage location                */
/*-------------------------------------------------------------------*/
/* ARG(1)   - Location                                               */
/* ARG(2)   - Length                                                 */
/*********************************************************************/
 stg: return storage(d2x(arg(1)),arg(2))
```

```
/*********** @REFRESH END   STG      2002/07/13 15:49:12 *************/
/*********** @REFRESH BEGIN DEBUG    2004/08/28 21:13:30 *************/
/* DEBUG    - Use debugging services                                  */
/*------------------------------------------------------------------*/
/* PARM     - PEEK, TRACE, PROBE, TIMER                               */
/********************************************************************/
 debug: parse arg debugopt debugstring
        debugline   = sigl-1
        debugsource = strip(sourceline(debugline))
        if translate(debugopt) = 'PEEK' then
           do
            say 'Peek at line:' debugline':' debugsource
            say 'Peek at line:' debugline':' strip(debugstring)
            return
           end
        if translate(debugopt) = 'TRACE' then
           do
            say 'Trace started at line:' debugline':' debugsource
            interpret 'trace' debugstring
            return
           end
        if translate(debugopt) = 'PROBE' then
           do
            say 'Probe started at line:' debugline':' debugsource
            say 'Probing modules:' debugstring
            probe = translate(debugstring)
            return
           end
        if translate(debugopt) = 'TIMER' then
           do
            debugline   = sigl+1
            debugsource = strip(sourceline(debugline))
            if symbol('dbtime') = 'LIT' then
               do
                olddbtime = 0
                oldsyscpu = 0
                oldsyssrv = 0
               end
            dbtime  = time('e')
            syscpu  = strip(sysvar('SYSCPU'))
            syssrv  = strip(sysvar('SYSSRV'))
            idbtime = dbtime - olddbtime
            isyscpu = syscpu - oldsyscpu
            isyssrv = right((syssrv - oldsyssrv),5)
            timestats = idbtime 'CPU:' isyscpu 'SRV:' isyssrv
            say 'Interval:' timestats', at' debugline':' debugsource
            olddbtime = dbtime
            oldsyscpu = syscpu
            oldsyssrv = syssrv
            return
```

```
          end
/*********** @REFRESH END   DEBUG    2004/08/28 21:13:30 *************/
/*********** @REFRESH BEGIN MODTRACE 2003/12/31 21:56:54 *************/
/* MODTRACE - Module Trace                                           */
/*-------------------------------------------------------------------*/
/* TRACETYP - Type of trace entry                                    */
/* SIGLINE  - The line number called from                            */
/*********************************************************************/
 modtrace: if modtrace = 'NO' then return
           arg tracetyp sigline
           tracetyp = left(tracetyp,5)
           sigline = left(sigline,5)
/*********************************************************************/
/* Adjust MODSPACE for START                                         */
/*********************************************************************/
           if tracetyp = 'START' then
              modspace = substr(modspace,1,length(modspace)+1)
/*********************************************************************/
/* Set the trace entry                                               */
/*********************************************************************/
           traceline = modspace time('L') tracetyp module sigline sparms
/*********************************************************************/
/* Adjust MODSPACE for STOP                                          */
/*********************************************************************/
           if tracetyp = 'STOP' then
              modspace = substr(modspace,1,length(modspace)-1)
/*********************************************************************/
/* Determine where to write the traceline                            */
/*********************************************************************/
           if ispfenv = 'YES' & tsoenv = 'FORE' then
/*********************************************************************/
/* Write to the ISPF Log, do not use ISPWRAP here                    */
/*********************************************************************/
              do
               zedlmsg = traceline
               address ISPEXEC "LOG MSG(ISRZ000)"
              end
           else
              say traceline
/*********************************************************************/
/* SAY to SYSTSPRT                                                   */
/*********************************************************************/
           return
/*********** @REFRESH END   MODTRACE 2003/12/31 21:56:54 *************/
```

Robert Zenuk
Systems Programmer (USA)

A simple CICS 3270 sniffer and player

PROBLEM

Sometimes it can be difficult to figure out what's going on with a CICS end user. Sometimes you have to deal with people connected from foreign countries who don't speak your language. Sometimes they are not aware of which transactions they run.

Recently, in a particular situation, I really needed to see what the user was seeing, and none of our usual tools (like VTAM GTF, IP traces, CEDx/f trap, or even VTAM hook) were appropriate to provide an actual replay function.

SOLUTION

PSNIFF uses ZCOUT1 and ZCIN CICS GLUEs to capture TIOAs for a specified termid. These GLUEs allocate a shared storage area to store them (XPI getmain) for up to a hard-coded number of screens (see code). Then PCSNIFF is able to replay every screen captured.

USAGE

There are three commands:

- To activate TIOA capture:

  ```
  SNIF TERM=xxx =>
  ```

- To logically deactivate capture (but not disable the exit):

  ```
  SNIF PLAY =>
  ```

  A play flag is tested in the GLUEs (see code). It replays every screen with a one-second interval.

- Free shared storage:

  ```
  SNIF FREE =>
  ```

This will disable the GLUEs (ZCIN ZCOUT1).

EXTENSION

From these simple programs you could add various selection criteria like userid, some particular string within a BMS map, a transaction id, etc.

You can also code some synchronous XM (PC routine) to store the TIOA from the CICS regions and use recorded data as input to a replay scenario engine.

PZCOUT1

```
*================================================================*
* PZOUT1 :  ASM/No Translator (SDFHMAC in SYSLIB)                *
* PROCESS:  Check Termid ,                                       *
*           Check capture not logically deactivated,             *
*           Obtain GWA, Getmain Shared Storage, Capture TIOA,    *
*           store pointer, increment the number of screens,      *
*           check maximum screens                                *
*================================================================*
         TITLE 'PZCOUT1 - SIMPLE CICS 327Ø SNIFFER output data'
         DFHUEXIT TYPE=EP,ID=XZCOUT1        XZCOUT1 GLUE
         DFHUEXIT TYPE=XPIENV               XPI
         COPY DFHTIOA                       TIOA
         COPY DFHTCTTE                      TCTTE
         COPY DFHSMMCY                      for XPI calls
*
TCTTEAR  EQU  4                             use R4 for TCTTE
TIOABAR  EQU  5                             usr R5 for TIOA
PZCOUT1 CSECT
PZCOUT1 AMODE 31
PZCOUT1 RMODE ANY
         SAVE (14,12)
         LR    R11,R15
         USING PZCOUT1,R11
         LR    R2,R1                        DFHUEPAR plist provided
         USING DFHUEPAR,R2                  R2 for it
         LA    R15,UERCNORM                 Set response OK
         L     R1Ø,UEPGAA                   Get my (ZCOUT1) GWA
         CLI   Ø(R1Ø),X'FF'                 is sniffer in play mode
         BE    RETURN                       don't do anything just exit
         CLI   Ø(R1Ø),X'ØØ'                 has sniffer already init
         BE    RETURN                       just exit
         L     R4,UEPTCTTE                  Get TCTTE addr
```

```
         CLC   TCTTETI(4),8(R1Ø)           is termid requested ?
         BNE   RETURN                      just exit if no.
         L     R5,UEPTIOA                  Get TIOA addr
         L     R6,UEPXSTOR                 For XPI parmlist
         USING DFHSMMC_ARG,R6              usr R6 for it
         L     R13,UEPSTACK                make sure R13=> Stack
         LH    R8,TIOATDL                  Terminal data length (TIOA)
         AH    R8,=H'6'                    + prefix
         DFHSMMCX CALL,                                                X
               CLEAR,                                                  X
               IN,                                                     X
               FUNCTION(GETMAIN),                                      X
               STORAGE_CLASS(SHARED_USER),                             X
               SUSPEND(NO),                                            X
               INITIAL_IMAGE(X'ØØ'),                                   X
               GET_LENGTH((R8)),                                       X
               OUT,                                                    X
               ADDRESS((R4)),                                          X
               RESPONSE(*),                                            X
               REASON(*),
         CLI SMMC_RESPONSE,SMMC_OK         Getmain in SDSA ok ?
         BNE   STOP                        just exit and force X'ff'
         LH    R1,TIOATDL                  Terminal data length
         AH    R1,=H'6'                    Skip TIOA prefix.
         LR    RØ,R4                       Get receive addr
         LR    R15,R1                      Prepare to MVCL
         LA    R14,TIOATDL                 the TIOA into
         MVCL  RØ,R14                      previously getmained stor.
         L     R1,12(R1Ø)                  Load Counter
         LA    R1,1(R1)                    Add one
         C     R1,4(R1Ø)                   Have we reached Max screen
         BH    STOP                        If yes, STOP => GWA+Ø=>FF
         ST    R1,12(R1Ø)                  Store result in place
         SLL   R1,2                        multiply by 4
         ST    R4,12(R1,R1Ø)               Store buffer addr in GWA
         B     RETURN                      Return
STOP     EQU   *
         MVI   Ø(R1Ø),X'FF'                Force sniffer in PLAY MODE
RETURN   DS    ØH
         L     R13,UEPEPSA                 Reaload SA
         RETURN (14,12),RC=UERCNORM
         EJECT ,
         LTORG ,
                 DS ØD
         DFHEIEND
         DFHREGS
         END   PZCOUT1
```

PZCIN

```
*==================================================================*
* PZCIN  :  ASM/No Translator (SDFHMAC in SYSLIB)                  *
* PROCESS:  Check Termid                                           *
*           Check capture not logically deactivated                 *
*           Obtain ZCOUT1 GWA via its GWA, Getmain Shared Storage*
*           Capture TIOA,                                          *
*           store pointer, increment the number of screens,        *
*           check maximum screens                                  *
*==================================================================*
         TITLE 'PZCIN - SIMPLE CICS 327Ø SNIFFER Input data'
         DFHUEXIT TYPE=EP,ID=XZCIN          XZCIN GLUE
         DFHUEXIT TYPE=XPIENV               XPI
         COPY DFHTIOA                       TIOE
         COPY DFHTCTTE                      TCTTE
         COPY DFHSMMCY                      for XPI calls
TCTTEAR  EQU  9                             usr R9 for TCTTE
TIOABAR  EQU  5                             use R5 for TIOA
PZCIN  CSECT
PZCIN  AMODE 31
PZCIN  RMODE ANY
           SAVE (14,12)
           LR    R11,R15
           USING PZCIN,R11
           LR    R2,R1                      DFHUEPAR plist provided
           USING DFHUEPAR,R2                R2 for it
           LA    R15,UERCNORM               Set response OK
           L     R1Ø,UEPGAA                 Get My (ZCIN) GWA
           L     R1Ø,Ø(R1Ø)                 and GET ZCOUT1 GWA
           CLI   Ø(R1Ø),X'FF'               is sniffer in play mode
           BE    RETURN                     don't do anything just exit
           CLI   Ø(R1Ø),X'ØØ'               has sniffer already init
           BE    RETURN                     just exit
           L     R9,UEPTCTTE                Get TCTTE
           CLC   TCTTETI(4),8(R1Ø)          Is TERMID requested  .
           BNE   RETURN                     just exit if no.
           L     R5,UEPTIOA                 Get TIOA addr
           L     R6,UEPXSTOR                For XPI parmlist
           USING DFHSMMC_ARG,R6             use R6, for addr
           L     R13,UEPSTACK               make sure R13=>Stack
           LH    R8,TIOATDL                 Terminal data length (TIOA)
           AH    R8,=H'6'                   + prefix
           AH    R8,=H'4'                   + CL3'INP' + X'AID' = 4
           DFHSMMCX CALL,                                          X
                 CLEAR,                                            X
                 IN,                                               X
                 FUNCTION(GETMAIN),                                X
                 STORAGE_CLASS(SHARED_USER),                       X
                 SUSPEND(NO),                                      X
```

```
              INITIAL_IMAGE(X'ØØ'),                                    X
              GET_LENGTH((R8)),                                        X
              OUT,                                                     X
              ADDRESS((R4)),                                           X
              RESPONSE(*),                                             X
              REASON(*)
         CLI  SMMC_RESPONSE,SMMC_OK          Getmain in SDSA ok
         BNE  STOP                           just exit and force X'FF'
         MVC  Ø(3,R4),=CL3'INP'              Eye catcher for input data
         MVC  3(1,R4),TCTTEAID               Get Aid from TCTTE
         LA   R4,4(R4)                       +4 to skip prefix.
         LH   R1,TIOATDL                     Terminal data length
         AH   R1,=H'6'                       Skip TIOA prefix
         LR   RØ,R4                          Get this receive addr
         LR   R15,R1                         Prepare to MVCL
         LA   R14,TIOATDL                    the TIOA into
         MVCL RØ,R14                         the getmained storage
         L    R1,12(R1Ø)                     Load counter
         LA   R1,1(R1)                       Add one
         C    R1,4(R1Ø)                      Have We reached Max screens
         BH   STOP                           Is yes => STOP X'FF'
         ST   R1,12(R1Ø)                     Store result in place
         SLL  R1,2                           multiply by 4
         SH   R4,=H'4'                       -4
         ST   R4,12(R1,R1Ø)                  Store buffer addr in GWA
         B    RETURN                         return
STOP     EQU  *
         MVI  Ø(R1Ø),X'FF'                   Force sniffer in play mode
RETURN   DS   ØH
         L    R13,UEPEPSA                    Reaload SA
         RETURN (14,12),RC=UERCNORM
         EJECT ,
         LTORG ,
         DS ØD
         DFHEIEND
         DFHREGS
         END  PZCIN
```

PSNIFF2

```
*=================================================================*
* PSNIFF :  ASM/CICS/CICSKEY                                      *
* USAGE  :  SNIF TERM="Termid"  start capture process             *
*        :  SNIF PLAY           Stop capture and start replay     *
*        :  SNIF FREE           Stop GLUEs and Freemain capture   *
*                               Buffer                            *
*  TEST  : ok CTS 2.2
* LOGIC  :                                                        *
*   AT SNIF TERM="Termid"                                         *
```

```
*         PSNIFF Retrieve the termid you want to trace          *
*         Operational Flag  at GWA+X'ØØ' = ØØ (Not ready yet)   *
*         Enable ZCOUT1 and ZCIN                                *
*         Store ZCOUT1 gwa pointer in ZCIN GWA                  *
*         Set MaxScreen                                         *
*         Enable Capture  :GWA+X'ØØ' = X'Ø1'=>Go Capture        *
*   AT SNIF PLAY                                                *
*         Disable Capture  :GWA+X'ØØ' = X'FF' =>Play Mode       *
*         PSNIFF Retrieve ZCOUT1 GWA which contains at +X'16'   *
*         a Word Tab of every capture buffer address.           *
*         Analyse input or output and send those buffers with   *
*         a one second internval between the sends.             *
*   AT FREE                                                     *
*         Check  Capture  :GWA+X'ØØ' = X'FF' Play mode          *
*         PSNIFF Retrieve ZCOUT1 GWA to get the Word tab        *
*         Freemain one by one the capture buffers and disable   *
*         ZCIN and ZCOUT1.                                      *
*===============================================================*
*===============================================================*
* Working Storage                                               *
*===============================================================*
DFHEISTG DSECT ,
MSGAID   DS    CL8Ø                    Display Aid Entered Message
BUFIN    DS    CL8Ø                    Buffer for retrieve
TERMSEL  DS    CL4                     Select Termid
GWAL     DS    F                       GWA Length
RESPONSE DS    F                       Response from CICS API
LONG     DS   F                        Getmain Length 1
B3270L   DS   H                        Getmain Length 2
LEN      DS   H                        Work send Length
INTV     DS   F                        Interval (s) beetween play
WCC      DS   C                        WCC
B        DS    C                       EXIT RETURN CODE
         DFHEJECT
PSNIFF CSECT ,
PSNIFF RMODE ANY
         DFHREGS ,                     Equate registers
         MVC   INTV(4),=F'1'           Interval in PLay mode=>1s
         MVC   BUFIN(8Ø),=8ØC' '       Blank retrieve buffer
         EXEC CICS RECEIVE INTO(BUFIN) RESP(RESPONSE)
         CLC   BUFIN+5(3),=CL3'PLAY'   Is Play function requested?
         BE    PLAY                    then process.
         CLC   BUFIN+5(3),=CL3'FREE'   Is Free function requested?
         BE    FREE                    then  process
         CLC   BUFIN+5(5),=CL5'TERM='  Is Term function requested?
         BNE   BADCMD                  if not, issue a error msg.
         MVC   TERMSEL(4),BUFIN+1Ø     otherwise  get Termid
         B     ENABLE                  and process.
BADCMD   EQU  *
         EXEC CICS SEND TEXT FROM(MSGERØØ) TERMINAL ERASE LENGTH(43)
```

```
         B    RETURN                              Bad syntax.
*================================================================*
* Enable Exit :                                                  *
* 1 Enable ZCOUT1 and ZCIN                                       *
* 2 ZCOUT1 GWA layout X'ØØ' : Operational Flag X'ØØ',X'Ø1',X'FF' *
*                     X'Ø4' : Max screens to be captured         *
*                     X'Ø8' : Termid                             *
*                     X'ØC' : Current number of captured screens *
*   At X'1Ø' Start of Address TAB (each work points to captured  *
*   buffer getmained by ZCIN and ZCOUT) along the capture period.*
*================================================================*
ENABLE   EQU  *
*                                      Test GLUE already active
         EXEC CICS EXTRACT EXIT PROGRAM('PZCOUT1') GASET(R5)          *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BE    ENAØØ9Ø
         EXEC CICS EXTRACT EXIT PROGRAM('PZCIN') GASET(R8)            *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BE    ENAØØ9Ø
*
         EXEC CICS ENABLE PROGRAM('PZCOUT1') EXIT('XZCOUT1') START    *
               GALENGTH(32760) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BNE   ENAØØ1Ø
         EXEC CICS EXTRACT EXIT PROGRAM('PZCOUT1') GASET(R5)          *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BNE   ENAØØ1Ø
         EXEC CICS ENABLE PROGRAM('PZCIN') EXIT('XZCIN') START        *
               GALENGTH(2Ø)   RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BNE   ENAØØ1Ø
         EXEC CICS EXTRACT EXIT PROGRAM('PZCIN') GASET(R8)            *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
         BNE   ENAØØ1Ø
         ST    R5,Ø(R8)                    ZCOUT1 GWA addr in ZCIN GWA
         MVC   4(4,R5),MAXSCREEN           Max Screens Number
         MVC   8(4,R5),TERMSEL             Termid to capture
         MVC   12(4,R5),=F'Ø'              Current Captured screens
         MVI   Ø(R5),X'Ø1'                 Go capture
         EXEC CICS SEND TEXT FROM(MSGOKØ1) TERMINAL ERASE LENGTH(43)
         B    RETURN                       Return to CICS.
ENAØØ1Ø  EQU   *
         EXEC CICS SEND TEXT FROM(MSGERØ2) TERMINAL ERASE LENGTH(43)
         B    RETURN                       Return to CICS.
ENAØØ9Ø  EQU   *
         EXEC CICS SEND TEXT FROM(MSGERØ7) TERMINAL ERASE LENGTH(43)
```

```
         B    RETURN                          Return to CICS.
*================================================================*
* PLAY :  Get GWA addr                                          *
*          Mark GWA as inactive (X'FF') at offset X'ØØ'         *
*================================================================*
PLAY     EQU  *
         EXEC CICS EXTRACT EXIT PROGRAM('PZCOUT1') GASET(R5)          *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)     Failed to acces
         BNE   PLAY1Ø                       exit GWA, exit
         MVI  Ø(R5),X'FF'                   Mark capture in play mode
         L     R6,12(R5)                    Number of screens
         CLC   16(4,R5),=F'Ø'              If nothing has been captured
         BE    PLAY9Ø                       Exit.
         LA    R5,16(R5)                    To first Screen
         L     R8,Ø(R5)                     First TIOA captured buffer
         CLC   Ø(3,R8),=CL3'INP'            Is it inbound traffic.
         BNE   PLAYØØ                       No,
         LA    R5,4(R5)                     Yes, so skip it and get
         L     R8,Ø(R5)                     To the next Screen
         BCTR  R6,Ø                         -1 in number of capturedbf
         LTR   R6,R6                        was it the only buffer
         BZ    PLAY9Ø                       nothing left to process ex
PLAYØØ   EQU   *
         CLC   Ø(3,R8),=CL3'INP'            If input data
         BE    INPUTØØ                      go and process
         LH    R4,Ø(R8)                     Load buffer length
         LA    R4,2(R4)                     +2
         STH   R4,B327ØL                    Store lengtjh for send
         MVC   WCC(1),3(R8)                 retrieve original WCC
         CLI   2(R8),X'F5'                  Is it W/E 327Ø command  ?
         BNE   NOCLEAR                      so consider => noerase
         EXEC  CICS SEND FROM(4(R8)) LENGTH(B327ØL) CTLCHAR(WCC) ERASE
         B     PLAYNXT                       Next, screen
NOCLEAR EQU   *
         EXEC  CICS SEND  FROM(4(R8)) LENGTH(B327ØL) CTLCHAR(WCC)
PLAYNXT EQU   *
         EXEC CICS DELAY FOR SECONDS(INTV)
         LA    R5,4(R5)                     To Next Screen pointer
         L     R8,Ø(R5)                     To Next Screen
         BCT   R6,PLAYØØ                    Loop on replay.
PLAY1Ø  EQU   *
         EXEC CICS SEND TEXT FROM(MSGOKØ5) TERMINAL ERASE LENGTH(43)
         B     RETURN                       return to CICS.
*================================================================*
* PLAY INPUT                                                    *
*================================================================*
INPUTØØ  EQU   *
         LH    R4,4(R8)                     Get Length
         LTR   R4,R4                        Is it null
```

```
         BNZ   INPUT4Ø                      no,process length not null
         BAL   R9,GETAID                    otherwise GETAID
INPUT2Ø  EQU   *
         MVC   WCC(1),7(R8)                 And send which Key has
         EXEC CICS SEND FROM(MSGAID) LENGTH(9) CTLCHAR(WCC)
         B    PLAYNXT
INPUT4Ø  EQU   *
*        MVC   WCC,7(R8)                    Get original WCC
         MVC    LEN(2),4(R8)                Get TIOA length
INPUT4B  EQU   *
         EXEC  CICS SEND  FROM(8(R8)) LENGTH(LEN) CTLCHAR(WCC)
INPUT5Ø  EQU  *
         EXEC CICS DELAY FOR SECONDS(INTV)
         BAL   R9,GETAID                    Get AID ex: *PF1Ø*
INPUT7Ø  EQU   *
         MVC   WCC(1),7(R8)                 Get original WCC
         EXEC CICS SEND FROM(MSGAID) LENGTH(9) CTLCHAR(WCC)
         B    PLAYNXT                       Next Screen to play.
         CNOP  Ø,4
PLAY9Ø   EQU   *
         EXEC CICS SEND TEXT FROM(MSGOKØ4) TERMINAL ERASE LENGTH(43)
         B     RETURN
*=================================================================*
* Get Aid                                                        *
*=================================================================*
GETAID   EQU   *
         MVC   MSGAID(3),=XL3'114Ø4Ø'       SBA for col1-Row1 for
         MVI   MSGAID+3,C'*'                AID message
         MVI   MSGAID+8,C'*'
         LA    R2,AIDTABN                   AIDTAB number of rows
         LA    R1,AIDTAB                    AIDTAB addr
AIDØØ    EQU   *
         CLC   3(1,R8),Ø(R1)                Match TIOA/AIDTAB
         BE    AID2Ø                        we got it
         LA    R1,8(R1)                     Next on in AIDTAB
         BCT   R2,AIDØØ                     look throught ADITAB
         MVC   MSGAID+4(4),=CL4'????'       curious AID found.
         B     AID9Ø                        Return.
AID2Ø    EQU   *
         MVC   MSGAID+4(4),1(R1)            Get AID text : "PF1Ø"
AID9Ø    EQU   *
         BR    R9                           Return to caller
         CNOP  Ø,4
*=================================================================*
* FREE : disable GLUE  and freemain captured buffers.            *
*=================================================================*
FREE     EQU   *
         EXEC CICS EXTRACT EXIT PROGRAM('PZCOUT1') GASET(R5)          *
               GALENGTH(GWAL) RESP(RESPONSE)
         CLC   RESPONSE,DFHRESP(NORMAL)
```

```
         BNE   FREE9Ø
         CLI  Ø(R5),X'FF'                  If  Capture in Play Mode
         BNE  FREE99                      No, issue play to stop before
         L     R6,12(R5)                   Number of screens
         CLC   16(4,R5),=F'Ø'              Nothing captured
         BE    FREE8Ø                      Disable exits and return
         LA    R5,16(R5)                   To first Screen Addr
         L     R8,Ø(R5)                    To first Screen
FREEØØ   EQU   *
         EXEC  CICS FREEMAIN DATAPOINTER(R8)
         LA    R5,4(R5)                    To Next Screen addr
         L     R8,Ø(R5)                    To Next Screen
         BCT   R6,FREEØØ                   Till the end
FREE8Ø   EQU   *
         EXEC  CICS DISABLE PROGRAM('PZCOUT1') STOP EXITALL
         EXEC  CICS DISABLE PROGRAM('PZCIN') STOP EXITALL
FREE9Ø   EQU   *
         EXEC CICS SEND TEXT FROM(MSGOKØ3) TERMINAL ERASE LENGTH(43)
         B     RETURN
FREE99   EQU   *
         EXEC CICS SEND TEXT FROM(MSGERØ6) TERMINAL ERASE LENGTH(43)
         B     RETURN
RETURN   DS    ØH
         EXEC CICS RETURN
FIN      DS    ØH
         CNOP  Ø,4
*================================================================*
* AIDTAB                                                         *
*================================================================*
AIDTAB   DS    ØF
         DC    X'6Ø',CL7'No'
         DC    X'7D',CL7'Ent'
         DC    X'F1',CL7'PF1'
         DC    X'F2',CL7'PF2'
         DC    X'F3',CL7'PF3'
         DC    X'F4',CL7'PF4'
         DC    X'F5',CL7'PF5'
         DC    X'F6',CL7'PF6'
         DC    X'F7',CL7'PF7'
         DC    X'F8',CL7'PF8'
         DC    X'F9',CL7'PF9' '
         DC    X'7A',CL7'PF1Ø'
         DC    X'7B',CL7'PF11'
         DC    X'7C',CL7'PF12'
         DC    X'C1',CL7'PF13'
         DC    X'C2',CL7'PF14'
         DC    X'C3',CL7'PF15'
         DC    X'C4',CL7'PF16'
         DC    X'C5',CL7'PF17'
         DC    X'C6',CL7'PF18'
```

```
          DC    X'C7',CL7'PF19'
          DC    X'C8',CL7'PF20'
          DC    X'C9',CL7'PF21'
          DC    X'4A',CL7'PF22'
          DC    X'4B',CL7'PF23'
          DC    X'4C',CL7'PF24'
          DC    X'6C',CL7'PA1'
          DC    X'6E',CL7'PA2'
          DC    X'6B',CL7'PA3'
          DC    X'6D',CL7'Clr'
          DC    F'0'
AIDTABN   EQU   (*-AIDTAB)/8
*=================================================================*
* Constants                                                       *
*=================================================================*
MAXSCREEN DC    F'250'
IO        DC    X'114040'
          DC    CL32'Trigger reset ok..'
MSGER00   DC    X'114040',CL40'Invalid syntax: SNIF TERM=/PLAY/FREE'
MSGOK01   DC    X'114040',CL40'Capture started'
MSGER02   DC    X'114040',CL40'Failed to enable ZCIN/ZCOUT1'
MSGOK03   DC    X'114040',CL40'Free storage and disable ok '
MSGOK04   DC    X'114040',CL40'No more screens to be played'
MSGOK05   DC    X'114040',CL40'Replay ended'
MSGER06   DC    X'114040',CL40'PLAY before FREE'
MSGER07   DC    X'114040',CL40'Already active please FREE'
          END   PSNIFF
```

David Harou
Systems Programmer (France)

CICS and Service-Oriented Architecture

A Service-Oriented Architecture (SOA) is essentially a collection of services that communicate with each other. A service is a unit of work executed by a service provider. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. A means is needed of connecting services to each other. Web services essentially use XML to create a connection. Web services can be used to combine mainframe CICS applications with Web-based applications.

Web services simplify integration by reducing the number of APIs to one – SOAP, and the number of data formats to one – XML. It should be noted that there are variations in XML syntax, eg FinXML, FIXML, OFX, and IFX.

Some CICS transactions interact with an end user at a terminal – a terminal-oriented transaction. These use BMS to handle the presentation logic of the transaction. The other type of transaction doesn't interact with an end user – another program invokes the transactions (a COMMAREA transaction). These COMMAREA applications take a request from a requesting application and return data to it in a single step. This is very similar to the way Web services work. With Web services, a single SOAP request provides the host data. BMS expresses fields and data as name/value pairs, and this can be converted to XML for use by Web services.

There are two ways to integrate CICS applications as Web services, and both use adapters. There are connectors that run on the mainframe and can use native interfaces allowing seamless integration with the target application. Alternatively, there are gateways that run off the mainframe on middle-tier servers and might use screen-scraping.

The decision of whether to use connectors or gateways usually depends on the types of application to be integrated. CICS TS provides facilities to allow the use of connectors to access legacy applications. Remote applications can invoke just about every existing CICS application as a Web service.

With connectors, no additional hardware is required and no changes to the CICS application are needed.

Gateways, running on a physical or logical middle tier, usually involve some form of screen-scraping. There is very tight integration between the gateway and a specific application, which means that any changes to the application will break the integration.

When gateways communicate with terminal-oriented CICS applications they open or start a terminal session with the

mainframe application, send some sort of request to the application, receive a response in the form of a terminal datastream, capture the screen data using HLLAPI, process the screen data, convert the contents to XML, and ship the XML document to the requester. (Sometimes an FEPI (Front End Programming Interface) is used on the mainframe instead of a terminal emulation client running on a middle-tier. This maintains the gateway model; it just moves the middle tier onto the mainframe.)

Gateways are a quick way to get a Web service, but there is always the chance of a performance bottleneck with screen scraping, plus there are numerous potential points of failure between the mainframe application and the Web service. For this reason, gateways are best for short-term projects, either as a transition to using connectors or as a stop-gap measure during application reengineering or platform migration.

Products that implement the connector model for Web services come in two types – dynamic or using code generation.

Code generators require developers to stick to a fairly structured process. The process requires users to:

1 Download program source code, copybooks, or screen maps to a workstation running a proprietary tool.

2 Generate the Web services 'wrapper' code, which usually involves using a proprietary tool.

3 Upload the generated code back to the mainframe.

4 Compile, link-edit, install, and test the code that was generated in step 2.

5 Repeat steps 1–4 for each program and after a program changes.

Dynamic connectors have little or no configuration, and changes to legacy applications are automatically included in the SOAP/XML output. Often there is no configuration required, while some cases may require a single-step process to

specify Web service information for each application.

CICS TS includes facilities that allow third-party vendors to create connectors that can immediately enable mainframe applications as Web services. These facilities provide additional benefits over gateways, such as improved performance and better stability than with screen-scraping alternatives. Some connectors use the same industry-standard technologies as Web services, which makes it possible for applications to transparently make use of CICS transactions within a Web services architecture and receive the resulting data as well-formed XML.

So, basically, if you want a Service-Oriented Architecture, you want connectors rather than gateways, and preferably dynamic connectors.

Nick Nourse
Independent Consultant (UK)

Why not share your expertise and earn money at the same time? *CICS Update* is looking for program code, REXX EXECs, JavaScript, etc, that experienced users of CICS have written to make their life, or the lives of their users, easier. We are also looking for explanatory articles, and hints and tips, from experienced users. We would also like suggestions on how to improve CICS performance. Articles should be sent to the editor, Trevor Eddolls, at trevore@xephon.com.

Flashline has announced that its Flashline Registry integrates with WebSphere Studio Asset Analyzer (WSAA) to automatically harvest artefacts from mainframe systems, including CICS. This integration makes mainframe software assets more visible, understandable, and reusable.

WSAA scans System/390 and distributed assets, including COBOL, PL/I, JCL, CICS, and IMS. Flashline's WSAA Connector automatically populates the Flashline Registry with information about the mainframe assets, enabling them to be shared with development teams across the enterprise alongside Java and .NET artefacts. Teams can quickly review asset interdependencies, workflows, e-business ratings, and data dependencies.

For further information contact:
Flashline, 1300 East 9th Street, Suite 1600, Cleveland, OH 44114, USA.
Tel: 800 259 1961.
URL: www.flashline.com/Press/view.jsp?sid=1101122738149-3643529019-102&id=227.

***

Fair Isaac Corporation has announced the general availability of Fair Isaac Blaze Advisor for COBOL, which allows organizations to incorporate a rules decisioning system that co-exists with and upgrades existing mainframe applications, while reducing development loads on IT departments. It is designed for organization wanting to reduce the time and costs required to incorporate changes in business priorities into its decision systems.

When utilizing rule management techniques and business rules on a mainframe, clients usually have to add Java-based services that require complex interfaces across mainframe regions to their existing CICS and COBOL applications. Fair Isaac Blaze Advisor for COBOL enables users to develop rule services that can be compiled to IBM mainframe-standard COBOL, and by enabling these rule services to read and write data in their COBOL copybooks.

Blaze Advisor for COBOL enables users to develop rule services that can be compiled to ANSI-standard COBOL, and allows these rule services to read and write data in COBOL copybooks/CICS interfaces.

For further information contact:
Fair Isaac Corp, 901 Marquette Avenue, Suite 3200, Minneapolis, MN 55402, USA.
Tel: (612) 758 5200.
URL: www.fairisaac.com/Fairisaac/Solutions/Product+Index/Blaze+Advisor.

***

IBM has enhanced eight tools from its Application Integration and Middleware (AIM) portfolio for zSeries mainframes.

One of them, CICS Interdependency Analyzer (IA) Version 1.3, has been enhanced with the ability to give more information for customers to dynamically balance workloads for CICS applications.

For further information contact your local IBM representative
URL: www-306.ibm.com/software/htp/cics/ianaly/overview.html.

***