# 229

CICS update

*December 2004*

## 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

Bob Thomas
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.*

# A simple CICS program to open files

## INTRODUCTION

You can use this transaction to open files in a CICS region. It can be scheduled to run during CICS start-up. The basic idea behind this transaction is to avoid individually opening all the files in a CICS region and also to avoid the time delay in cases where the files have been migrated.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   OPENFILE.
INSTALLATION. NBIC.
*****************************************************************
*                     PROGRAM PROLOGUE                          *
*****************************************************************
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
       01  FILLER                      PIC X(13)  VALUE 'OSPPENFXX W/S'.
       01  WORK-FIELDS.
             05  WS-FILE-NAME                PIC X(8).
             05  WS-RESP2                    PIC S9(4) COMP.
             05  WS-RESP1                    PIC S9(4) COMP.
             05  WS-OPEN-STATUS              PIC S9(8) COMP.
             05  WS-ENA-STATUS               PIC S9(8) COMP.
             05  WS-DSNAME                   PIC X(44).
             05  WS-COMM-AREA                PIC X.
LINKAGE SECTION.
       01  DFHCOMMAREA              PIC X.
       EJECT
PROCEDURE DIVISION.
 000-MAINLINE.
      EXEC CICS INQUIRE FILE START END-EXEC.
      MOVE 0 TO WS-RESP2.

      PERFORM UNTIL WS-RESP2 = DFHRESP(END)
              EXEC CICS INQUIRE FILE(WS-FILE-NAME) NEXT
                              DSNAME(WS-DSNAME)
                        OPENSTATUS(WS-OPEN-STATUS)
                        ENABLESTATUS(WS-ENA-STATUS)
                        RESP(WS-RESP2)
              END-EXEC

              IF WS-FILE-NAME(1:3) = 'OSF'
```

```
                        IF WS-OPEN-STATUS = DFHVALUE(CLOSED)
                                  OR WS-ENA-STATUS = DFHVALUE(UNENABLED)
                          EXEC CICS
                              SET FILE(WS-FILE-NAME) OPEN
                                    NOHANDLE
                          END-EXEC
                        END-IF
                END-IF
        END-PERFORM

        EXEC CICS INQUIRE FILE
                  END
        END-EXEC.

000-MAINLINE-X.
        EXEC CICS
RETURN
        END-EXEC.
        GOBACK.
        EJECT
```

The system commands used in this sample code are described below.

## INQUIRE FILE START

The system programming commands allow you to inquire about the definition and status of most of the resources defined to CICS, and about many elements of the CICS system as well. INQUIRE FILE in this example is used to browse a resource defined to CICS. In this case a FILE.

### Starting the browse

The general syntax to start a browse using INQUIRE is:

```
INQUIRE <resource-type> START
```

### Accessing the next resource

In the second step of a browse, you issue the INQUIRE command repetitively with another new option, NEXT. CICS returns one resource definition for each INQUIRE NEXT.

The general syntax is:

```
INQUIRE <resource-type (data-area)> NEXT
```

### Ending the browse

Stopping the browse is the final step. To do this you issue an INQUIRE for the resource type with just the END option; thus the browse ends:

```
INQUIRE <resource-type>END
```

### OPENSTATUS

An option on INQUIRE FILE, OPENSTATUS returns a CVDA value identifying the status of the active CICS dump dataset.

CVDA values are:

- CLOSED – the active CICS dump dataset is closed.
- OPEN – the active CICS dump dataset is open.

### ENABLESTATUS

An option on INQUIRE FILE, ENABLESTATUS returns a CVDA value identifying whether application programs can access the file.

CVDA values are:

- DISABLED – the file is unavailable for access by application programs because it has been explicitly disabled. It must be explicitly enabled by a SET FILE ENABLED command or its CEMT equivalent before it can be accessed by application programs.
- DISABLING – a request to disable the file has been received, but tasks are executing that had previously accessed the file. These tasks are allowed to complete their use of the file, but new tasks are not allowed access.
- ENABLED – the file is available for access by application programs.
- UNENABLED – the file is unavailable for access by application programs because it is closed. It must be explicitly enabled by a SET FILE OPEN command or its

CEMT equivalent before it can be accessed by application programs.

- UNENABLING – a request to close the file has been received, but tasks are executing that had previously accessed the file. These tasks are allowed to complete their use of the file, but new tasks are not allowed access.
- RESP – CICS sets a primary and sometimes a secondary response code when it completes a command, and provides options for you to inspect them. The primary code returned by the RESP option is the general result – either NORMAL, meaning that the command executed successfully, or the name of an exception condition such as NOTAUTH (not authorized) or INVREQ (invalid request). The secondary code, returned by RESP2, provides a finer level of detail.
- DFHVALUE – you also use DFHVALUE when your program needs to interpret a value returned as a CVDA.
- SET FILE – change the attributes of a VSAM or BDAM file, including files that refer to CICS shared data tables and coupling facility data tables.

*Akila Balaji*
*Programmer/Analyst*
*Cognizant Technology Solutions (India)*


## CICS Business Event Publisher

According to IBM estimates, today, CICS handles more than 300 billion transactions and almost 90% of zSeries customers use CICS for their core applications. The issue is how we can continue to exploit CICS while allowing the enterprises to meet the current demands for Web interfacing and integrating legacy with other applications in a distributed environment.

CICS Business Event Publisher (BEP) for MQ Series from IBM is one of the ways by which this can be achieved – in a non-intrusive fashion. This article aims to explain what CICS BEP offers and how it can be used to realize the business needs of an organization.

## CONCEPTS OF BEP

BEP works on the principle that by monitoring the business event in the source applications (back-end transactions), you can enhance and extend its capabilities and drive new business processes that utilize new technology.

To be able to monitor the business event in the legacy application you should be able to identify:

- Information processing activities associated with a business event.
- Data and its format associated with the activities.

BEP enables users to define rules to monitor these business events and potentially generate messages to be put into the queue based on the events and the associated data.

The key point to note is that BEP performs this event monitoring and message publishing transparently to the CICS application – in the sense that the existing application code doesn't have to change to use BEP.

## BEP COMPONENTS

The BEP architecture consists of four major components – data space server, message server, event source connector, and rules database.

### BEP data space server

The BEP data space server is a long-running MVS started task that maintains the state of BEP message servers and event source connectors across warm starts of the servers and

connectors. Because the data space server maintains the state information (in a semi-persistent manner), the event source regions can be shut down without loss of data.

### BEP message server

The BEP message server is another long-running started task or batch job, which accepts event messages from event source connectors and writes the messages to the appropriate MQSeries queues.

The message server architecture consists of four components:

1 Main component – consists of a message server task that performs initiation, termination, and monitoring of the message server. The operator interfaces execute commands from the console.

2 Set component – consists of one or more subtasks defined to manage event sources. Set subtasks – one each for each set defined – run in the message server address space. These subtasks allocate storage areas for holding messages generated by the event source connectors, dispatch the queue tasks that write the messages to MQ queues, and monitor the event sources. Note that more than one event source can belong to a set – appropriate when they are low-volume event sources.

3 QTASK component – consists of MVS subtasks attached by each set subtask and is responsible for writing the event messages to the appropriate message queues.

4 QMGR component – consists of one task for each MQSeries queue manager accessed by the message server. The QMGR tasks monitor the query manager and maintain the inventory of queues to which the messages are being written.

A single message server can support CICS, DB2, and IMS event source connectors simultaneously. The option of running multiple message servers is also available and can be used for workload balancing or ease of management.

Parameters controlling the message server are specified in the CBMPARM library:

- MMAIN corresponds to the main component.
- MSETS corresponds to the set component – defining the sets.
- For each set task, the parameters are stored in an optional member (MSET followed by the 4-character set name – eg MSETS).
- MQMGR corresponds to the QMGR, where the queue managers can be predefined.
- For each QMGR task, the parameters are stored in an optional member (QUES followed by the queue manager subsystem id).
- For each of the queue subtasks, the parameters are stored in an optional member (MXTK followed by the event source set name).

### BEP event source connectors

The BEP event source connector monitors events in CICS, DB2, and IMS event sources and applies the pre-defined rules to determine whether they have to be published. The event monitors, when required, create the message from the associated data and pass it to the message server.

Each of CICS, IMS, and DB2 has its own event source connectors aimed at exploiting the native features appropriately. For example, BEP uses CICS global user exits for event monitoring, DB2 system log records for determining whether an event has occurred, and IMS logger exit routines to capture log images written by IMS.

### BEP rules database

The BEP rules database maintains and administers the rules defined to select and process the business events. It consists of three components:

1 Rules database component – a standard VSAM dataset that is a permanent repository of the rules and associated objects (rule groups).

2 Workstation administration client – Windows-based tool that allows the user to maintain the rules to be used by the message servers and event source connectors. Arranging rules as rule groups simplifies administration and the implementation of the rules.

3 BEP TCP/IP Listener – allows for objects to be uploaded to/downloaded from the mainframe message server and the workstation client. While doing this, the security requirements for the client requests are enforced.

### How BEP components work together

- The rules are pre-defined by the user and stored in the BEP rules database.
- During initialization, the BEP message server builds a copy of this rules database (based on the definition in the MMAIN member of CBMPARMS) in virtual storage.
- This virtual storage is located in the data space owned by the BEP data space server that is connected to the BEP message server.
- The event source connectors associated with this message server instance perform the event selection and monitoring based on the rules in virtual storage.

## USING BEP

CICS BEP Version 1.2 allows for multiple application-related events to be monitored and detected. The event processor determines whether the event is eligible to be published based on the rules defined. Once an event is found to be eligible to be published, it builds the message and passes the message to the BEP message server. The BEP message server writes the message to the appropriate queue with options specified in the rule.

The IMS and DB2 event processor allows the monitoring of a record insert/update/delete request on IMS and DB2 databases respectively.

The CICS event processor can monitor the following events:

- Read/update/write/delete/unlock requests on a VSAM file.
- Program control requests (LINK).
- Read/write/delete requests on a Temporary Storage Queue (TSQ) or a Transient Data Queue (TDQ).
- Interval Control Requests (eg START).

The eligibility of the event can be further controlled by specifying additional criteria in the rules database. The samples of such criteria include:

- Origin of the event – transaction ID, terminal ID, CICS APPLID, User ID, etc.
- Data content – the actual value of the data associated with the event.
- Event response code – eg CICS response code.

The published message can contain either all of the data associated with the event (eg the entire VSAM record that is added) or up to 128 specific message fields.

The destination queue manager and queue can either be statically defined in the rule, or dynamically created based on user-specified criteria when the message is created.

Other factors worth noting in the above-mentioned process are:

- In response to an event that it found eligible, the event processor can generate multiple messages (for example the event matches more than one rule) directed to multiple destinations.
- The MQPUT options that are to be used on the MQPUT

request can be specified (if not specified, the default values will be used).

- The sending of a message can be delayed until a syncpoint is reached.
- Also multiple messages within a single unit of work can be accumulated to reduce the network overhead.
- The data-related selection criteria and the message creation fields can be specified as offset/length pairs or selected from a list of COBOL copybook variables. Additionally, the message content can include specific literal values.
- User-written exit programs can be used either to exclude a specific event or to alter the message created before it is published.
- BEP internal trace entries can be created to aid with problem determination.

## POTENTIAL USES OF BEP

Uses can now integrate their existing legacy-based CICS applications into newer applications based on the latest technology. Enterprise Application Integration, Business-to-Business Integration, and Web enabling of legacy applications are the potential uses of BEP.

Message brokers that offer sophisticated message alteration and distribution depend on message creation that is external to them. BEP handles this gap and can act as the bridge between a legacy application and a message broker.

Also BEP can be used as an effective means of logging and monitoring CICS applications from an environment external to the mainframe (say as a gateway to an application monitoring solution running on a different platform).

The event monitoring capabilities of BEP can also help in the automation of business processes and error handling, improving the availability of applications.

The push message technology option provided by BEP can be exploited by eBusiness applications aiming at near real-time updates.

## CONCLUSION

BEP provides a much-needed non-intrusive way to extend legacy applications (without requiring changes to application code) that is fast and safe to use. As a concept, it is a simple one, which monitors events and can publish messages based on the associated data. How these messages are further exploited is up to the user, and the ways of making business use of it are unlimited.

Currently it supports only outbound messages – enabling the other applications to integrate well with legacy applications. There is a possibility that inbound messages will be supported in the future, enabling seamless integration of applications in both directions.

*Sasirekha Cota*
*Tata Consultancy Services (India)*


# EXCI batch client control program – part 2

*This month we conclude the code for the batch client control program.*

```
*--------------------------------------------------------------------*
*- Check the response code and produce diagnostics if required       -*
*--------------------------------------------------------------------*
A600CHCK CLC   EXCI_RESPONSE,=AL4(EXCI_NORMAL)  IF NORMAL RESPONSE
         BE    A600OK                            THEN OK MESSAGE
         BAL   R14,Z300_EXCI_DIAGNOSTICS         ELSE DIAGNOSTICS
         MVC   CURR_RC,EXCI_RESPONSE                   SET RC
         B     A600RET                                 AND RETURN.
A600OK   MVC   CM419COM_SYSPRINT_MSG,=C'CM420602I'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420602I),CM420602I  & TEXT
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
```

```
*--------------------------------------------------------------------*
*- Return to caller                                                 -*
*--------------------------------------------------------------------*
A600RET  BAL   R14,Z400_CHECK_RC       CHECK HIGHEST RC
         L     R14,A600SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
**********************************************************************
*- A 7 0 0 _ D E A L L O C _ P I P E  :  EXCI Deallocate_Pipe        -*
**********************************************************************
*- R2   DSECT - CM420COM                                            -*
*- R3   BASE                                                        -*
*- R8   DSECT - CM419COM                                            -*
*- R9   DSECT - EXCI_RETURN_CODE                                    -*
*- R13  DSECT - DFHEISTG                                            -*
*- R14  Linkage                                                     -*
**********************************************************************
A700_DEALLOC_PIPE  DS  0H
         ST    R14,A700SR14            SAVE REGISTER 14
         XC    CURR_RC,CURR_RC         CLEAR CURRENT RC
         BAL   R14,Z200_WRITE_SYSPRINT BLANK LINE
*--------------------------------------------------------------------*
*- Write message for Deallocate_Pipe call.                          -*
*--------------------------------------------------------------------*
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420701I'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420701I),CM420701I  & TEXT
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*--------------------------------------------------------------------*
*- Call Deallocate_Pipe                                             -*
*--------------------------------------------------------------------*
         MVC   CM420COM_CALL_TYPE,=AL4(DEALLOCATE_PIPE)  CALL TYPE
A700CALL CALL  DFHXCIS,                                             X
               (CM420COM_VERSION_NUM,                               X
               CM420COM_RETURN_AREA,                                X
               CM420COM_USER_TOKEN,                                 X
               CM420COM_CALL_TYPE,                                  X
               CM420COM_PIPE_TOKEN),                                X
               VL,                                                  X
               MF=(E,CM420COM_PL)
*--------------------------------------------------------------------*
*- Check the response code and produce diagnostics if required      -*
*--------------------------------------------------------------------*
A700CHCK CLC   EXCI_RESPONSE,=AL4(EXCI_NORMAL)  IF NORMAL RESPONSE
         BE    A700OK                           THEN OK MESSAGE
         BAL   R14,Z300_EXCI_DIAGNOSTICS        ELSE DIAGNOSTICS
         MVC   CURR_RC,EXCI_RESPONSE                 SET RC
         B     A700RET                               AND RETURN.
A700OK   MVC   CM419COM_SYSPRINT_MSG,=C'CM420702I'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420702I),CM420702I  & TEXT
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
```

```
*-------------------------------------------------------------------*
*- Return to caller                                                 -*
*-------------------------------------------------------------------*
A700RET  BAL   R14,Z400_CHECK_RC       CHECK HIGHEST RC
         L     R14,A700SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*********************************************************************
*- A 9 0 0 _ T E R M I N A T I O N  :  Perform Termination          -*
*********************************************************************
*- R3   BASE                                                        -*
*- R4   Work Register                                               -*
*- R8   DSECT - CM419COM                                            -*
*- R13  DSECT - DFHEISTG                                            -*
*- R14  Linkage                                                     -*
*- R15  Linkage and CM419 return code                               -*
*********************************************************************
A900_TERMINATION  DS  0H
         ST    R14,A900SR14            SAVE REGISTER 14
         XC    CURR_RC,CURR_RC         CLEAR CURRENT RC
         BAL   R14,Z200_WRITE_SYSPRINT BLANK LINE
*-------------------------------------------------------------------*
*- Convert RC to displayable decimal characters and issue message.  -*
*-------------------------------------------------------------------*
         L     R4,HIGH_RC              LOAD RC
         CVD   R4,WORKDW_1             CONVERT TO DECIMAL
         UNPK  WORKDW_2,WORKDW_1       CONVERT TO ...
         OI    WORKDW_2+7,X'F0'        ... DISPLAYABLE DECIMAL
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420901I'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420901I),CM420901I    & TEXT
         MVC   CM419COM_SYSPRINT_DATA+40(L'WORKDW_2),WORKDW_2   & RC
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*-------------------------------------------------------------------*
*- LINK to CM419 passing CM419COM for a CLOSE of SYSPRINT.          -*
*-------------------------------------------------------------------*
         MVI   CM419COM_FUNCTION,CM419COM_PUT_CLOSE  SET FUNCTION
A900LINK LINK  EP=CM419,PARAM=(CM419ST),MF=(E,LINK_PL)
         ST    R15,CURR_RC             SAVE RETURN CODE
*-------------------------------------------------------------------*
*- Return to caller                                                 -*
*-------------------------------------------------------------------*
A900RET  BAL   R14,Z400_CHECK_RC       CHECK HIGHEST RC
         L     R14,A900SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*********************************************************************
*- Z 1 0 0 _ L O A D _ P R O G  :  Load a sub-program               -*
*********************************************************************
*- R0   Sub-program EPA                                             -*
*- R3   BASE                                                        -*
```

```
*- R8   DSECT - CM419COM                                              -*
*- R13  DSECT - DFHEISTG                                              -*
*- R14  Linkage                                                       -*
************************************************************************
Z100_LOAD_PROG  DS  0H
         ST    R14,Z100SR14            SAVE REGISTER 14
*---------------------------------------------------------------------*
*- LOAD Macro - EPLOC contains Entry Point Name. If the load fails   -*
*- the program will abend.                                           -*
*---------------------------------------------------------------------*
Z100LOAD LOAD EPLOC=EPLOC
         ST    R0,LOADPT               SAVE LOAD POINT
*---------------------------------------------------------------------*
*- Issue program loaded message. Convert the LOADPT address to       -*
*- displayable characters.                                           -*
*---------------------------------------------------------------------*
Z100MSG  DS    0H                      ISSUE MESSAGE
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420002I'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420002I),CM420002I  & TEXT
         MVC   CM419COM_SYSPRINT_DATA+8(L'EPLOC),EPLOC OVERLAY PGM
         MVC   WORK05(L'LOADPT),LOADPT ADDITIONAL BYTE AVOIDS SWAP
         UNPK  WORK09,WORK05           UNPACK ADDRESS
         MVC   WORKDW_1(L'WORKDW_1),WORK09      MOVE REQ. BYTES (8)
         TR    WORKDW_1,TRANTAB0-240   TRANSLATE REQ. BYTES
         MVC   CM419COM_SYSPRINT_DATA+37(L'WORKDW_1),WORKDW_1  MOVE
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*---------------------------------------------------------------------*
*- Return to caller                                                  -*
*---------------------------------------------------------------------*
Z100RET  L     R14,Z100SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
************************************************************************
*- Z 2 0 0 _ W R I T E _ S Y S P R I N T :  Write message            -*
************************************************************************
*- R3   BASE                                                         -*
*- R8   DSECT - CM419COM                                             -*
*- R13  DSECT - DFHEISTG                                             -*
*- R14  Linkage                                                      -*
************************************************************************
Z200_WRITE_SYSPRINT  DS  0H
         ST    R14,Z200SR14            SAVE REGISTER 14
         MVI   CM419COM_FUNCTION,CM419COM_PUT    SET FUNCTION
*---------------------------------------------------------------------*
*- LINK to CM419 passing CM419COM. If the LINK doesn't work the      -*
*- program will abend.                                               -*
*---------------------------------------------------------------------*
Z200LINK LINK EP=CM419,PARAM=(CM419ST),MF=(E,LINK_PL)
*---------------------------------------------------------------------*
*- The return code from CM419 is not saved as a current return       -*
```

```
*- code as this would have a negative effect on the current and      -*
*- high return codes logic. Any I/O errors will be covered by        -*
*- system abends.                                                    -*
*--------------------------------------------------------------------*
*        ST    R15,CURR_RC             SAVE RETURN CODE
*--------------------------------------------------------------------*
*- Return to caller                                                  -*
*--------------------------------------------------------------------*
Z200RET  L     R14,Z200SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
**********************************************************************
*- Z 3 0 0 _ E X C I _ D I A G N O S T I C S  :  Diagnostics         -*
**********************************************************************
*- R3   BASE                                                         -*
*- R4   Work Register                                                -*
*- R8   DSECT - CM419COM                                             -*
*- R9   DSECT - EXCI_RETURN_CODE                                     -*
*- R13  DSECT - DFHEISTG                                             -*
*- R14  Linkage                                                      -*
**********************************************************************
Z300_EXCI_DIAGNOSTICS  DS  0H
         ST    R14,Z300SR14            SAVE REGISTER 14
*--------------------------------------------------------------------*
*- Write EXCI diagnostics follow message.                            -*
*--------------------------------------------------------------------*
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420010E'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA(L'CM420010E),CM420010E  & TEXT
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*--------------------------------------------------------------------*
*- Convert EXCI_RESPONSE to displayable decimal characters           -*
*- and issue message.                                                -*
*--------------------------------------------------------------------*
         L     R4,EXCI_RESPONSE        LOAD RESPONSE
         CVD   R4,WORKDW_1             CONVERT TO DECIMAL
         UNPK  WORKDW_2,WORKDW_1       CONVERT TO ...
         OI    WORKDW_2+7,X'F0'        ... DISPLAYABLE DECIMAL
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420010E'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA+6(L'CM420010A),CM420010A  & TEXT
         MVC   CM419COM_SYSPRINT_DATA+28(L'WORKDW_2),WORKDW_2   & CODE
         BAL   R14,Z200_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*--------------------------------------------------------------------*
*- Convert EXCI_REASON to displayable decimal characters             -*
*- and issue message.                                                -*
*--------------------------------------------------------------------*
         L     R4,EXCI_REASON          LOAD REASON
         CVD   R4,WORKDW_1             CONVERT TO DECIMAL
         UNPK  WORKDW_2,WORKDW_1       CONVERT TO ...
         OI    WORKDW_2+7,X'F0'        ... DISPLAYABLE DECIMAL
         MVC   CM419COM_SYSPRINT_MSG,=C'CM420010E'     MOVE MSG. NO.
```

```
         MVC   CM419COM_SYSPRINT_DATA+6(L'CM42ØØ1ØB),CM42ØØ1ØB  & TEXT
         MVC   CM419COM_SYSPRINT_DATA+28(L'WORKDW_2),WORKDW_2   & CODE
         BAL   R14,Z2ØØ_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*-------------------------------------------------------------------*
*- Convert EXCI_SUB_REASON1 to displayable decimal characters       -*
*- and issue message.                                               -*
*-------------------------------------------------------------------*
         L     R4,EXCI_SUB_REASON1     LOAD SUB-REASON 1
         CVD   R4,WORKDW_1             CONVERT TO DECIMAL
         UNPK  WORKDW_2,WORKDW_1       CONVERT TO ...
         OI    WORKDW_2+7,X'FØ'        ... DISPLAYABLE DECIMAL
         MVC   CM419COM_SYSPRINT_MSG,=C'CM42ØØ1ØE'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA+6(L'CM42ØØ1ØC),CM42ØØ1ØC  & TEXT
         MVC   CM419COM_SYSPRINT_DATA+28(L'WORKDW_2),WORKDW_2   & CODE
         BAL   R14,Z2ØØ_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*-------------------------------------------------------------------*
*- Convert EXCI_SUB_REASON2 to displayable decimal characters       -*
*- and issue message.                                               -*
*-------------------------------------------------------------------*
         L     R4,EXCI_SUB_REASON2     LOAD SUB-REASON 2
         CVD   R4,WORKDW_1             CONVERT TO DECIMAL
         UNPK  WORKDW_2,WORKDW_1       CONVERT TO ...
         OI    WORKDW_2+7,X'FØ'        ... DISPLAYABLE DECIMAL
         MVC   CM419COM_SYSPRINT_MSG,=C'CM42ØØ1ØE'     MOVE MSG. NO.
         MVC   CM419COM_SYSPRINT_DATA+6(L'CM42ØØ1ØD),CM42ØØ1ØD  & TEXT
         MVC   CM419COM_SYSPRINT_DATA+28(L'WORKDW_2),WORKDW_2   & CODE
         BAL   R14,Z2ØØ_WRITE_SYSPRINT  WRITE MSG TO SYSPRINT
*-------------------------------------------------------------------*
*- Return to caller                                                 -*
*-------------------------------------------------------------------*
Z3ØØRET  L     R14,Z3ØØSR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*********************************************************************
*- Z 4 Ø Ø _ C H E C K _ R C  :  Check Return Codes                 -*
*********************************************************************
*- R3   BASE                                                       -*
*- R13  DSECT - DFHEISTG                                           -*
*- R14  Linkage                                                    -*
*********************************************************************
Z4ØØ_CHECK_RC  DS  ØH
         ST    R14,Z4ØØSR14            SAVE REGISTER 14
         CLC   CURR_RC,HIGH_RC         IF CURRENT RC > HIGHEST RC
         BH    Z4ØØHIGH                   THEN RESET HIGHEST RC
         B     Z4ØØRET                    ELSE RETURN
Z4ØØHIGH MVC   HIGH_RC,CURR_RC         MOVE CURRENT RC TO HIGHEST RC
*-------------------------------------------------------------------*
*- Return to caller                                                 -*
*-------------------------------------------------------------------*
Z4ØØRET  L     R14,Z4ØØSR14            RESTORE REGISTER 14
```

```
         BR    R14                     RETURN TO CALLER
         EJECT
*************************************************************************
*- M O D I F I A B L E   I N S T R U C T I O N S                      -*
*************************************************************************
* GETPROG and GETAPPL, both in A100.
         EJECT
*************************************************************************
*- C O N S T A N T S                                                  -*
*************************************************************************
* None!!
         EJECT
*************************************************************************
*- T A B L E S                                                        -*
*************************************************************************
TRANTAB0 DC    C'0123456789ABCDEF'     TRANSLATE HALF BYTES
         EJECT
*************************************************************************
*- M E S S A G E S                                                    -*
*************************************************************************
CM420001I  DC  C'Loading program          ....'
CM420002I  DC  C'Program          loaded at address X''        ''.'
CM420003I  DC  C'         storage address X''        '' length         X
                bytes.'
CM420010E  DC  C'Error processing EXCI call, diagnostics follow:-'
CM420010A  DC  C'EXCI Response . . . : '
CM420010B  DC  C'EXCI Reason . . . . : '
CM420010C  DC  C'EXCI Subreason-1  . : '
CM420010D  DC  C'EXCI Subreason-2  . : '
CM420101I  DC  C'PARM Field - Program=         Region=        .'
CM420102E  DC  C'PARM Field contains errors, program terminating....'
CM420201I  DC  C'INITIALIZE_USER processing....'
CM420202I  DC  C'INITIALIZE_USER successful.'
CM420301I  DC  C'ALLOCATE_PIPE processing....'
CM420302I  DC  C'ALLOCATE_PIPE successful.'
CM420401I  DC  C'OPEN_PIPE processing....'
CM420402I  DC  C'OPEN_PIPE successful.'
CM420501I  DC  C'LINKing to sub-program          ...'
CM420502I  DC  C'Control returned from sub-program          RC='
CM420601I  DC  C'CLOSE_PIPE processing....'
CM420602I  DC  C'CLOSE_PIPE successful.'
CM420701I  DC  C'DEALLOCATE_PIPE processing....'
CM420702I  DC  C'DEALLOCATE_PIPE successful.'
CM420901I  DC  C'Program Terminated. Highest Return Code '
         EJECT
*************************************************************************
*- E N D   CM420                                                      -*
*************************************************************************
         END   CM420
```

## CM420A01 – CM420COM

```
*********************************************************************
*                 C A R L   W A D E   M C B U R N I E               *
*                   -  I T   C O N S U L T A N T  -                 *
*                            www.cwmit.com                          *
*********************************************************************
* MODULE NAME = CM420A01                                            *
* MODULE TYPE = DSECT CM420COM                                      *
* DESCRIPTION = Communications Area for CM420 (Assembler)           *
*               External CICS Interface Batch Client Control.       *
*               This member contains all the definitions            *
*               required for the External CICS Interface            *
*               CALL interface, including the call parameter        *
*               list, return_area, and dpl_retarea.                 *
*               NB this member is for use only by EXCI client       *
*               programs and the client programs must also           *
*               include the CICS supplied copybooks                 *
*               DFHXCPLD and DFHXCRCD.                              *
*********************************************************************
         EJECT
*********************************************************************
* CHANGE HISTORY:                                                   *
*********************************************************************
         EJECT
*********************************************************************
* C M 4 2 0    -     Communications Area                            *
*********************************************************************
CM420COM              DSECT
CM420COM_ALIGN        DS    0D            ALIGNMENT
CM420COM_EYECATCH     DS    CL16          EYECATCHER
CM420COM_EXCI         DS    0F            EXCI CALL INTERFACE
CM420COM_VERSION_NUM  DS    F             EXCI VERSION_NUMBER
CM420COM_RETURN_AREA  DS    CL(EXCI_RETURN_CODE__LEN) EXCI RETURN_AREA
CM420COM_USER_TOKEN   DS    F             EXCI USER_TOKEN
CM420COM_CALL_TYPE    DS    F             EXCI CALL_TYPE
CM420COM_PIPE_TOKEN   DS    F             EXCI PIPE_TOKEN
CM420COM_PGMNAME      DS    CL8           EXCI PGMNAME
CM420COM_USER_NAME    DS    CL8           EXCI USER_NAME
CM420COM_CICS_APPL    DS    CL8           EXCI CICS APPLID
*------_COMMAREA                        * PROVIDED BY DPL CALLER
CM420COM_COMMAREA_LEN DS    F              EXCI COMMAREA LENGTH
CM420COM_DATA_LEN     DS    F              EXCI DATA LENGTH
CM420COM_TRANSID      DS    CL4            EXCI TRANSACTION ID.
*------_UOWID                           * NULL ADDRESS
*------_USERID                          * NULL ADDRESS = DEFAULT
CM420COM_DPL_RETAREA  DS    CL(EXCI_DPL_RETAREA__LEN) EXCI DPL_RETAREA
CM420COM_OPTIONS      DS    XL1            EXCI OPTIONS
CM420COM_EXCI_LEN     EQU   *-CM420COM_EXCI    LENGTH
CM420COM_PL  CALL ,                                                 X
```

```
               (CM420COM_VERSION_NUM,                                  X
               CM420COM_RETURN_AREA,                                   X
               CM420COM_USER_TOKEN,                                    X
               CM420COM_CALL_TYPE,                                     X
               CM420COM_PIPE_TOKEN,                                    X
               CM420COM_PGMNAME,                                       X
               Ø,                                                      X
               CM420COM_COMMAREA_LEN,                                  X
               CM420COM_DATA_LEN,                                      X
               CM420COM_TRANSID,                                       X
               Ø,                                                      X
               Ø,                                                      X
               CM420COM_DPL_RETAREA,                                   X
               CM420COM_OPTIONS),                                      X
               VL,                                                     X
               MF=L
CM420COM_LENGTH       EQU   *-CM420COM    LENGTH OF CM420COM
*-----------------------------------------------------------------------*
*- E N D   CM420A01                                                    -*
*-----------------------------------------------------------------------*
         EJECT
```

## CM419

CM419 is a general I/O subprogram for reading input from SYSIN and writing output to SYSPRINT. It is used by CM420 for writing messages to SYSPRINT and can, but must not, be used by any EXCI client program. CM420 always loads CM419 and when CM420 links to an EXCI client program it passes both the CM420COM and CM419COM COMMAREAs to the client program.

```
CM419    TITLE 'CM419 : I/O SUB-PROGRAM'
************************************************************************
*                 C A R L   W A D E   M C B U R N I E                  *
*                   -  I T   C O N S U L T A N T  -                    *
*                            www.cwmit.com                             *
************************************************************************
* MODULE NAME = CM419                                                  *
* MODULE TYPE = CSECT  (sub-program)                                   *
* DESCRIPTION = Batch I/O sub-program.                                 *
*               This program must be passed a parameter list           *
*               in Register 1 :                                        *
*               Address 1 :  CM419COM                                  *
*               This program is more or less a "utility" for           *
*               reading input from SYSIN and writing output            *
*               to SYSPRINT.                                           *
*               The function required of each call to CM419 must       *
```

```
*               be specified in CM419COM_FUNCTION. The supported      *
*               functions are :                                        *
*               CM419COM_GET       - read record from SYSIN            *
*                                    (implicit open if required)       *
*               CM419COM_GET_CLOSE - read record and close SYSIN       *
*               CM419COM_PUT       - write record to SYSPRINT          *
*                                    (implicit open if required)       *
*               CM419COM_PUT_CLOSE - write record and close SYSPRINT   *
*************************************************************************
         EJECT
*************************************************************************
* CHANGE HISTORY:                                                       *
*************************************************************************
         EJECT
*************************************************************************
* REGISTER  EQUATES                       USAGE                         *
* --------  --------  ------------------------------------------------ *
* REG  Ø    RØ                                                          *
* REG  1    R1        Work Register                                     *
* REG  2    R2                                                          *
* REG  3    R3        Base Register for CSECT CM419                     *
* REG  4    R4        Work Register                                     *
* REG  5    R5                                                          *
* REG  6    R6                                                          *
* REG  7    R7                                                          *
* REG  8    R8        DSECT - CM419COM                                  *
* REG  9    R9                                                          *
* REG 1Ø    R1Ø       Work Register                                     *
* REG 11    R11                                                         *
* REG 12    R12                                                         *
* REG 13    R13       SAVE AREA                                         *
* REG 14    R14       Linkage                                           *
* REG 15    R15       Linkage and return code                           *
*                                                                       *
*************************************************************************
         EJECT
*-----------------------------------------------------------------------*
*- Copybooks                                                           -*
*-----------------------------------------------------------------------*
         EJECT
         COPY  CM419Ø1               DSECT - CM419COM  (COMMAREA)
         EJECT
*-----------------------------------------------------------------------*
*- Register Equates                                                    -*
*-----------------------------------------------------------------------*
         DFHREGS                     CICS STANDARD EQUATES
         EJECT
*************************************************************************
*=       E  N  T  R  Y          P  O  I  N  T                         =*
*************************************************************************
```

```
CM419    CSECT
CM419    AMODE 31
CM419    RMODE 24
         SAVE  (14,12)             SAVE CALLER'S REGISTERS
         LR    R3,R15              SET BASE REGISTER
         USING CM419,R3            ESTABLISH ADDRESSABILITY
         L     R8,Ø(,R1)           --> COMMAREA CM419COM
         USING CM419COM,R8         MAP CM419COM
         ST    R13,CM419COM_SAVEAREA+4   SAVE CALLER'S SAVE AREA
         LR    R4,R13              STORE CALLER'S R13
         LA    R13,CM419COM_SAVEAREA     --> CM419 SAVE AREA
         ST    R13,8(R4)           SAVE MY SA IN CALLER'S SA
*-------------------------------------------------------------------*
*- Program Identification "Eye-Catchers"                           -*
*-------------------------------------------------------------------*
         B     AØØØ_MAINLINE           BRANCH OVER EYE-CATCHERS
ASMEYE   DC    C'*'                    ASTERISK
ASMPROG  DC    C'CM419   '             PROGRAM NAME
         DC    C'-'                    HYPHEN
ASMLVL   DC    C'CWMØØØØ1'             PROGRAM LEVEL
         DC    C' '                    BLANK
ASMDATE  DC    C'&SYSDATE'             DATE OF ASSEMBLY
         DC    C' '                    BLANK
ASMTIME  DC    C'&SYSTIME'             TIME OF ASSEMBLY
         DC    C'*'                    ASTERISK
ASMEYEL  EQU   *-ASMEYE                LENGTH OF EYE-CATCHER
         EJECT
*********************************************************************
*- A Ø Ø Ø _ M A I N L I N E :  Controls the flow of the program    -*
*********************************************************************
*- R3   CSECT - CM419                                              -*
*- R8   DSECT - CM419COM                                           -*
*- R13  SAVE AREA                                                  -*
*- R14  Linkage                                                    -*
*- R15  Return Code                                                -*
*********************************************************************
AØØØ_MAINLINE  DS  ØH
SELECTØØ EQU   *                       S E L E C T
WHENØØØØ CLI   CM419COM_FUNCTION,CM419COM_PUT         WHEN PUT
         BNE   WHENØØØ1
         BAL   R14,A2ØØ_PUT
         B     ENDØØ
WHENØØØ1 CLI   CM419COM_FUNCTION,CM419COM_GET         WHEN GET
         BNE   WHENØØØ2
         BAL   R14,A4ØØ_GET
         B     ENDØØ
WHENØØØ2 CLI   CM419COM_FUNCTION,CM419COM_PUT_CLOSE   WHEN CLOSE PUT
         BNE   WHENØØØ3
         BAL   R14,A6ØØ_PUT_CLOSE
         B     ENDØØ
```

```
WHEN0003 CLI   CM419COM_FUNCTION,CM419COM_GET_CLOSE   WHEN CLOSE GET
         BNE   WHEN0004
         BAL   R14,A800_GET_CLOSE
         B     END00
WHEN0004 EQU   *
OTHER00  EQU   *
         LA    R15,8                   SET ERROR RETURN CODE
END00    EQU   *                       E N D
*************************************************************************
*- A 0 0 0 _ R E T U R N :  Return Control                             -*
*************************************************************************
A000_RETURN  DS  0H
RETURN   L     R13,CM419COM_SAVEAREA+4 RESTORE CALLER'S R13
         RETURN (14,12),RC=(15)        RETURN TO CALLER
*************************************************************************
*=       E  X  I  T             P  O  I  N  T                         =*
*************************************************************************
         EJECT
*************************************************************************
*- A 2 0 0 _ P U T  :  Write record to SYSPRINT                        -*
*************************************************************************
*- R1   OPEN LIST                                                     -*
*- R3   CSECT - CM419                                                 -*
*- R8   DSECT - CM419COM                                              -*
*- R10  SYSPRINT DCB                                                  -*
*- R13  SAVE AREA                                                     -*
*- R14  Linkage                                                       -*
*- R15  Linkage                                                       -*
*************************************************************************
A200_PUT  DS  0H
         ST    R14,CM419COM_A200SR14   SAVE REGISTER 14
         CLI   CM419COM_SYSPRINT_SI,CM419COM_OPEN  IS SYSPRINT OPEN?
         BE    A200PUT                             YES - PUT RECORD
*---------------------------------------------------------------------*
*- OPEN SYSPRINT                                                      -*
*---------------------------------------------------------------------*
A200SYSP DS    0H
         MVC   CM419COM_SYSPRINT_DCB,SYSPRINT  MOVE DCB INTO STORAGE
         LA    R10,CM419COM_SYSPRINT_DCB       --> DCB
         MVC   CM419COM_SYSPRINT_OL,OPENLIST   MOVE OPENLIST
         LA    R1,CM419COM_SYSPRINT_OL         --> OPENLIST
         OPEN ((R10),(OUTPUT)),        PROCESS OPEN MACRO              X
               MODE=31,                                                X
               MF=(E,(1))
         MVI   CM419COM_SYSPRINT_SI,CM419COM_OPEN  SET AS OPEN
A200PUT  PUT   CM419COM_SYSPRINT_DCB,CM419COM_SYSPRINT_IOA
         XR    R15,R15                 CLEAR PUT ROUTINE ADDRESS
         MVC   CM419COM_SYSPRINT_IOA,=CL(L'CM419COM_SYSPRINT_IOA)' '
*---------------------------------------------------------------------*
*- Return to caller                                                   -*
```

```
*--------------------------------------------------------------------*
A200RET  L     R14,CM419COM_A200SR14   RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
**********************************************************************
*- A 4 0 0 _ G E T  :  Read record from SYSIN                        -*
**********************************************************************
*- R1   OPEN LIST                                                   -*
*- R3   CSECT - CM419                                               -*
*- R8   DSECT - CM419COM                                            -*
*- R10  SYSIN DCB                                                   -*
*- R13  SAVE AREA                                                   -*
*- R14  Linkage                                                     -*
*- R15  Linkage                                                     -*
**********************************************************************
A400_GET  DS  0H
         ST    R14,CM419COM_A400SR14   SAVE REGISTER 14
         CLI   CM419COM_SYSIN_SI,CM419COM_OPEN     IS SYSIN OPEN?
         BE    A400GET                             YES - GET RECORD
*--------------------------------------------------------------------*
*- OPEN SYSIN                                                       -*
*--------------------------------------------------------------------*
A400SYSI DS    0H
         MVC   CM419COM_SYSIN_DCB,SYSIN        MOVE DCB INTO STORAGE
         LA    R10,CM419COM_SYSIN_DCB          --> DCB
         MVC   CM419COM_SYSIN_OL,OPENLIST      MOVE OPENLIST
         LA    R1,CM419COM_SYSIN_OL            --> OPENLIST
         OPEN ((R10),(INPUT)),         PROCESS OPEN MACRO              X
               MODE=31,                                                X
               MF=(E,(1))
         MVI   CM419COM_SYSIN_SI,CM419COM_OPEN     SET AS OPEN
A400GET  GET   CM419COM_SYSIN_DCB,CM419COM_SYSIN_IOA
         B     A400RET                 BRANCH OVER CLOSE
*--------------------------------------------------------------------*
*- EODAD Exit Routine as specified in the SYSIN DCB                 -*
*--------------------------------------------------------------------*
A400EOF  DS    0H                      EODAD EXIT ROUTINE
         MVI   CM419COM_SYSIN_EOF_F,CM419COM_SYSIN_EOF
         BAL   R14,A800_GET_CLOSE      CLOSE SYSIN
*--------------------------------------------------------------------*
*- Return to caller                                                 -*
*--------------------------------------------------------------------*
A400RET  L     R14,CM419COM_A400SR14   RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
**********************************************************************
*- A 6 0 0 _ P U T _ C L O S E  :  Close SYSPRINT                    -*
**********************************************************************
*- R1   CLOSE LIST                                                  -*
*- R3   CSECT - CM419                                               -*
```

```
*- R8   DSECT - CM419COM                                              -*
*- R13  SAVE AREA                                                     -*
*- R14  Linkage                                                       -*
*- R15  Linkage                                                       -*
***********************************************************************
A600_PUT_CLOSE  DS  0H
         ST    R14,CM419COM_A600SR14   SAVE REGISTER 14
         CLI   CM419COM_SYSPRINT_SI,CM419COM_OPEN  IS SYSPRINT OPEN?
         BNE   A600RET                           NO - DON'T CLOSE IT
*---------------------------------------------------------------------*
*- CLOSE SYSPRINT                                                     -*
*---------------------------------------------------------------------*
A600SYSP DS    0H
         MVC   CM419COM_SYSPRINT_CL,CLSELIST   MOVE CLOSE LIST
         LA    R1,CM419COM_SYSPRINT_CL         --> CLOSE LIST
         CLOSE CM419COM_SYSPRINT_DCB,  PROCESS CLOSE MACRO             X
               MODE=31,                                                X
               MF=(E,(1))
         MVI   CM419COM_SYSPRINT_SI,CM419COM_CLOSE  SET AS CLOSED
*---------------------------------------------------------------------*
*- Return to caller                                                   -*
*---------------------------------------------------------------------*
A600RET  L     R14,CM419COM_A600SR14   RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
***********************************************************************
*- A 8 0 0 _ G E T _ C L O S E  :  Close SYSIN                        -*
***********************************************************************
*- R1   CLOSE LIST                                                    -*
*- R3   CSECT - CM419                                                 -*
*- R8   DSECT - CM419COM                                              -*
*- R13  SAVE AREA                                                     -*
*- R14  Linkage                                                       -*
*- R15  Linkage                                                       -*
***********************************************************************
A800_GET_CLOSE  DS  0H
         ST    R14,CM419COM_A800SR14   SAVE REGISTER 14
         CLI   CM419COM_SYSIN_SI,CM419COM_OPEN     IS SYSIN OPEN?
         BNE   A800RET                           NO - DON'T CLOSE IT
*---------------------------------------------------------------------*
*- CLOSE SYSIN                                                        -*
*---------------------------------------------------------------------*
A800SYSI DS    0H
         MVC   CM419COM_SYSIN_CL,CLSELIST      MOVE CLOSE LIST
         LA    R1,CM419COM_SYSIN_CL            --> CLOSE LIST
         CLOSE CM419COM_SYSIN_DCB,     PROCESS CLOSE MACRO             X
               MODE=31,                                                X
               MF=(E,(1))
         MVI   CM419COM_SYSIN_SI,CM419COM_CLOSE     SET AS CLOSED
*---------------------------------------------------------------------*
```

```
*- Return to caller                                                  -*
*---------------------------------------------------------------------*
A800RET  L     R14,CM419COM_A800SR14   RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
***********************************************************************
*- M O D I F I A B L E   I N S T R U C T I O N S                     -*
***********************************************************************
* None !!
         EJECT
***********************************************************************
*- C O N S T A N T S                                                 -*
***********************************************************************
SYSPRINT DCB   DSORG=PS,               PHYSICAL SEQUENTIAL             X
               DDNAME=SYSPRINT,        DDNAME                          X
               MACRF=(PM),             PUT MOVE                        X
               LRECL=133,              LOGICAL RECORD LENGTH           X
               RECFM=FB                FIXED BLOCK
SYSPRINT_DCB_LEN      EQU   *-SYSPRINT
SYSIN    DCB   DSORG=PS,               PHYSICAL SEQUENTIAL             X
               DDNAME=SYSIN,           DDNAME                          X
               MACRF=(GM),             GET MOVE                        X
               LRECL=8Ø,               LOGICAL RECORD LENGTH           X
               RECFM=FB,               FIXED BLOCK                     X
               EODAD=A4ØØEOF           END-OF-DATA EXIT
SYSIN_DCB_LEN         EQU   *-SYSIN
OPENLIST OPEN  OPENLIST,                                               X
               MODE=31,                                                X
               MF=L
OPENLIST_LEN          EQU   *-OPENLIST
CLSELIST CLOSE CLSELIST,                                               X
               MODE=31,                                                X
               MF=L
CLSELIST_LEN          EQU   *-CLSELIST
         EJECT
***********************************************************************
*- T A B L E S                                                       -*
***********************************************************************
* None !!
         EJECT
***********************************************************************
*- M E S S A G E S                                                   -*
***********************************************************************
* None !!
         EJECT
***********************************************************************
*- E N D   CM419                                                     -*
***********************************************************************
         LTORG
         END   CM419
```

## CM419A01 – CM419COM

```
*************************************************************************
*                 C A R L   W A D E   M C B U R N I E                   *
*                   -  I T   C O N S U L T A N T  -                     *
*                            www.cwmit.com                              *
*************************************************************************
* MODULE NAME = CM419AØ1                                                *
* MODULE TYPE = DSECT CM419COM                                          *
* DESCRIPTION = Communications Area for CM419 (Assembler)               *
*               I/O sub-program for SYSIN and SYSPRINT.                 *
*************************************************************************
         EJECT
*************************************************************************
* CHANGE HISTORY:                                                       *
*************************************************************************
         EJECT
*************************************************************************
* C M 4 1 9    -     Communications Area                                 *
*************************************************************************
CM419COM              DSECT
CM419COM_ALIGN        DS    ØD            ALIGNMENT
CM419COM_EYECATCH     DS    CL16          EYECATCHER
CM419COM_SAVEAREA     DS    18F           SAVE AREA
CM419COM_FUNCTION     DS    X             FUNCTION CODE
CM419COM_PUT          EQU   X'Ø1'         SYSPRINT - PUT REQUEST
CM419COM_PUT_CLOSE    EQU   X'11'         SYSPRINT - CLOSE REQUEST
CM419COM_GET          EQU   X'Ø2'         SYSIN - GET REQUEST
CM419COM_GET_CLOSE    EQU   X'12'         SYSIN - CLOSE REQUEST
CM419COM_SYSIN_SI     DS    X             SYSIN STATUS INDICATOR
CM419COM_SYSPRINT_SI  DS    X             SYSPRINT STATUS INDICATOR
CM419COM_OPEN         EQU   X'ØØ'         STATUS = OPEN
CM419COM_CLOSE        EQU   X'FF'         STATUS = CLOSE
CM419COM_SYSIN_EOF_F  DS    X             EOF INDICATOR
CM419COM_SYSIN_EOF    EQU   X'FF'         EOF FLAG SET
CM419COM_SYSPRINT_DCB DS    CL(CM419COM_SYSPRINT__L)       DCB
CM419COM_SYSPRINT_OL  DS    CL(CM419COM_OPENLIST__L)       OPEN PARM.
CM419COM_SYSPRINT_CL  DS    CL(CM419COM_CLSELIST__L)       CLOSE PARM.
CM419COM_SYSPRINT_IOA DS    ØCL133                SYSPRINT I/O AREA
CM419COM_SYSPRINT_CC  DS    CL1                   SYSPRINT ANSI CC
CM419COM_SYSPRINT_MSG DS    CL9                   SYSPRINT MSG. NO.
                      DS    CL1                   F I L L E R
CM419COM_SYSPRINT_DATA DS   CL122                 SYSPRINT DATA
CM419COM_SYSIN_DCB    DS    CL(CM419COM_SYSIN__L)       DCB
CM419COM_SYSIN_OL     DS    CL(CM419COM_OPENLIST__L)    OPEN PARM.
CM419COM_SYSIN_CL     DS    CL(CM419COM_CLSELIST__L)    CLOSE PARM.
CM419COM_SYSIN_IOA    DS    ØCL8Ø                 SYSIN    I/O AREA
CM419COM_SYSIN_DATA   DS    CL72                  SYSIN    DATA
CM419COM_SYSIN_NUMS   DS    CL8                   SYSIN    NUMBERS
CM419COM_A2ØØSR14     DS    F             SAVE REGISTER 14
```

```
CM419COM_A400SR14     DS    F             SAVE REGISTER 14
CM419COM_A600SR14     DS    F             SAVE REGISTER 14
CM419COM_A800SR14     DS    F             SAVE REGISTER 14
CM419COM_LENGTH       EQU   *-CM419COM    LENGTH OF CM419COM
*-------------------------------------------------------------------*
*- E N D   CM419A01                                                 -*
*-------------------------------------------------------------------*
         EJECT
CM419COM_SYSPRINT DCB DSORG=PS,        PHYSICAL SEQUENTIAL          X
               DDNAME=SYSPRINT,        DDNAME                       X
               MACRF=(PM),             PUT MOVE                     X
               LRECL=133,              LOGICAL RECORD LENGTH        X
               RECFM=F                 FIXED BLOCK
CM419COM_SYSPRINT__L  EQU   *-CM419COM_SYSPRINT
CM419COM_SYSIN DCB DSORG=PS,           PHYSICAL SEQUENTIAL          X
               DDNAME=SYSIN,           DDNAME                       X
               MACRF=(GM),             GET MOVE                     X
               LRECL=80,               LOGICAL RECORD LENGTH        X
               RECFM=F                 FIXED BLOCK
CM419COM_SYSIN__L     EQU   *-CM419COM_SYSIN
CM419COM_OPENLIST OPEN CM419COM_OPENLIST,                           X
               MODE=31,                                             X
               MF=L
CM419COM_OPENLIST__L  EQU   *-CM419COM_OPENLIST
CM419COM_CLSELIST CLOSE CM419COM_CLSELIST,                          X
               MODE=31,                                             X
               MF=L
CM419COM_CLSELIST__L  EQU   *-CM419COM_CLSELIST
         EJECT
```

## AN EXCI CLIENT PROGRAM

A forthcoming issue of *CICS Update* will include an EXCI client program linked to by CM420. The EXCI client program, CM412, provides an EXCI batch interface to the programmable interface for the resource definition on-line (DFHEDAP) program. This can be used to manage the CSD and it also supports the INSTALL command from a batch environment, which the DFHCSDUP utility does not.

*Carl Wade McBurnie*
*IT Consultant (Germany)*

## CICS – the well-known off-mainframe product?

With IBM celebrating 35 years of CICS back in August, and releasing Version 1.6 of z/OS, you would have thought that mainframes and CICS had a great future together. And, although estimates vary, it's reckoned that between 70% and 80% of the data in major companies is still accessed using CICS as the transaction processing system of choice. So why is the title of this article 'CICS – the well-known off-mainframe product?'? Well let's take a look at some recent announcements.

It all started earlier this year when Micro Focus and Microsoft announced an alliance to enable the migration of critical proprietary mainframe systems onto Windows using Microsoft's .NET technology. The Micro Focus Enterprise Server with its Mainframe Transaction Option was designed to enable the migration and deployment of CICS/COBOL mainframe applications onto the Windows platform. Why would anyone want to do that? The companies claimed that once an application had been re-hosted, it could be extended through the use of the .NET Framework, SQL Server 2000, XML, and Web services. The company also claimed it helped customers reduce the cost of maintaining and modernizing their mainframe environments, suggesting that cheaper platforms saved time and money.

Shortly after that, Micro Focus announced Mainframe Express Enterprise Edition, a Windows-based environment for mainframe application development. The product combined a mainframe emulation and development environment with application analysis and automated program and component generation. Again, there was a promise of existing CICS legacy services being extended to Web services, .NET, or J2EE.

Then came Micro Focus's partnership with Unilog – again to port mainframe applications, such as CICS, to Windows. Again the claim was better performance on low-cost platforms.

By now, Micro Focus had come up with a catchy name for what it was doing – 'Lift and Shift'. This it described as migrating legacy applications from older mainframes to contemporary platforms without the risks or costs associated with re-writing entire mission-critical applications. Those of you running on z990 mainframes might question the implication in the phrase 'older mainframes'.

By now, others were joining in the 'get CICS off the mainframe' way of thinking.

Fujitsu Software came out with Version 1.0 of NeoKicks. This provided a way for CICS applications to move to Microsoft .NET with ASP.NET pages, Windows Forms, and Visual Studio .NET using wizards. They claimed that it lowered platform maintenance costs, gave interfaces new life as ASP.NET Web applications or Windows Forms client applications, and integrated with Visual Studio .NET for higher developer productivity.

They maintained that offering a Windows server-based solution brought the applications to hardware platforms with highly-competitive price/performance ratios. They also suggested that migrating the code to the .NET environment allowed it to benefit from the best tools (in Visual Studio .NET) for maintenance and RAD, as well as placing the latest technologies fully within the reach of the migrated CICS applications. Transforming the CICS BMS screens into ASP.NET Web pages (Web Forms), or, optionally, Windows Forms, was seen as a way of freeing CICS applications from their character screen straitjacket. NeoKicks is available now in a limited membership Early Release Program.

Acucorp uses phrases such as 'Leveraging their existing CICS skills' and 'range of COBOL modernization capabilities not previously available on the mainframe' – you know where they're going with this. Their **extend** suite of products supports the IBM TXSeries for Multiplatforms.

What this means to CICS users is that (using **extend**) they can 'modernize' and integrate the COBOL business logic inherent

in their CICS systems with new technologies. This allows them to transition (Acucorp's word) chosen applications as required, while avoiding the cost and risk of re-engineering CICS mainframe systems.

ACUCOBOL-GT comes with a compiler, a run-time (called the COBOL Virtual Machine), indexed file system, support utilities, and a source-level interactive debugger. These tools provide the 'modernization' capabilities that, they claim, aren't available on a mainframe. 'The process of transitioning host systems is simplified by the use of the robust, secure, and scalable CICS facilities available in TXSeries-including familiar APIs, a common program structure, and interoperability with other CICS environments.'

The bottom line for Acucorp's offering is a way of preserving the existing CICS legacy systems while at the same time allowing them to be modernized and take advantage of distributed environments.

Not to be left out, September saw a new offering from Microsoft. They came up with Host Integration Server (HIS) 2004. This, they claimed, was for companies wanting to open up their mainframe assets to deeper integration with Windows-based servers, software, and the promise of Service-Oriented Architectures (SOA).

This is a different approach from Micro Focus's, which wanted to move CICS off the mainframe. This keeps CICS running on the mainframe, but makes it available to Internet users.

To be fair, Microsoft has gone to great lengths to ensure interoperability between IBM systems, DB2 databases, Windows Server, and MS SQL. For example, Transaction Integrator (TI) enables Windows developers to publish and extend business rules in mainframe CICS, IMS, and AS/400 applications as XML Web services. Its Enterprise Single Sign-On (ESSO) provides a way to authenticate security credentials between Windows Active Directory and non-Windows systems, enabling seamless authentication between HIS 2004, BizTalk Server 2004, IBM mainframe and mid-range systems, and

key IBM applications such as CICS, IMS, DB2, and MQSeries.

There's also Managed Provider for DB2, which enables developers to publish data stored in DB2 databases as XML Web services and integrate DB2 data with solutions based on Windows Forms, Web Forms, Web services, or Microsoft Office System applications such as Excel and InfoPath.

TI Host-Initiated Processing (HIP) allows a Windows Server-based computer to function as a peer to an IBM mainframe and AS/400 computer, enabling customers to build distributed peer-to-peer applications and move portions of their host application logic and data to (what they describe as) a more cost-effective infrastructure based on Windows Server and SQL Server.

Plus, there's IP-DLC (data link control) Link Service, which supports SNA over IP routing so that HIS 2004 computers can connect directly to z900 mainframes via high-speed IP networks. This means enterprises will not need to remotely administer branch cluster controllers, utilize expensive data link switching (DLSw)-capable routers, or maintain costly front-end processors.

NetManage has taken a similar approach with its Host Services Platform (HSP), a set of products and shared services, which they claim offer flexibility and speed of deployment when pursuing host-based business initiatives. It is designed to let users maximize host systems and incorporate them into a Service-Oriented Architecture (SOA). HSP offers development, emulation, end-to-end security, user management, reporting, and monitoring.

So, here are some questions to ask yourself. Are your CICS applications critical to business operations? Do you see them as an expensive burden? Do they run on costly hardware? Is there little innovation in maintenance or few enhancement options? I assume the answer to the first question is yes. If the answer to any of the other questions is also yes, then it would probably be worth your while investigating the non-mainframe development options.

*Editor's note: we would like to hear what other CICS users think about migrating off the mainframe to Windows. We'd also be interested to hear from anyone who has tried it. E-mail your comments to Trevor Eddolls at trevore@xephon.com.*

*Nick Nourse*
*Independent Consultant (UK)*


## CICSDUMP routine

The CICSDUMP REXX EXEC allows you to invoke IPCS for CICS SVC dump datasets from the ISPF 3.4 DSLIST screen as a line command. A pop-up allows you to pick your favourite CICS Verbexit options.

```
/*********************************************************************/
/*                                REXX                               */
/*********************************************************************/
/* Purpose: CICS/IPCS Dump formatter                                 */
/*-------------------------------------------------------------------*/
/* Syntax:  CICSDUMP dsn                                             */
/*-------------------------------------------------------------------*/
/* Parms: DSN        - The SVC Dump dataset                          */
/*********************************************************************/
/*                       Change Log                                  */
/*********** @REFRESH BEGIN START    2004/03/06 13:16:32  ************/
/* Standard housekeeping activities                                  */
/*********************************************************************/
 call time 'r'
 parse arg parms
 signal on syntax name trap
 signal on failure name trap
 signal on novalue name trap
 probe = 'NONE'
 modtrace = 'NO'
 modspace = ''
 call stdentry 'DIAGMSGS'
 module = 'MAINLINE'
 push trace() time('L') module 'From:' 0 'Parms:' parms
 if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
 call modtrace 'START' 0
/*********************************************************************/
/* Set local esoteric names                                          */
```

```
/*********************************************************************/
 @vio   = 'VIO'
 @sysda = 'SYSDA'
/*********** @REFRESH END   START    2004/03/06 13:16:32 *************/
/* Accept DSN                                                        */
/*********************************************************************/
 arg dumpdsn
 if dumpdsn = '' then call rcexit 12 'SVC Dump dataset missing'
/*********************************************************************/
/* Set defaults if not found in ISPF profile                         */
/*********************************************************************/
 if ispwrap(8 "VGET (DUMPIPCS) PROFILE") > 0 then
    dumpipcs = 'NOPROBLEM NOCONFIRM NOTERMINAL NODISPLAY PRINT'
 if ispwrap(8 "VGET (DUMPOPTS) PROFILE") > 0 then
    dumpopts = 'KE=1,DS=1,XM=1,AP=1,RM=1,DB2=1'
 if ispwrap(8 "VGET (DUMPVREL) PROFILE") > 0 then
    dumpvrel = 'DFHPD620'
 if ispwrap(8 "VGET (DUMPEXIT) PROFILE") > 0 then
    dumpvrel = 'SYS2.CICS.SDFHLINK'
/*********************************************************************/
/* Display a pop-up to confirm all options                           */
/*********************************************************************/
 mem.1  = ")ATTR                                                    "
 mem.2  = " @ TYPE(TEXT) COLOR(TURQ)                                "
 mem.3  = " # TYPE(INPUT) CAPS(ON) COLOR(RED)                       "
 mem.4  = ")BODY EXPAND(//) WINDOW(70,5)                            "
 mem.5  = "@DUMP DSN             :#Z                                "
 mem.6  = "@IPCS Options         :#Z                                "
 mem.7  = "@CICS VERBEXIT Options:#Z                                "
 mem.8  = "@CICS VERBEXIT Release:#Z                                "
 mem.9  = "@Optional VERBEXIT DSN:#Z                                "
 mem.10 = ")INIT                                                    "
 mem.11 = " .ZVARS = '(DUMPDSN DUMPIPCS DUMPOPTS DUMPVREL DUMPEXIT)'   "
 mem.12 = ")PROC                                                    "
 mem.13 = " VER (&DUMPDSN,NB,DSNAMEQ)                               "
 mem.14 = " VER (&DUMPIPCS,NB)                                      "
 mem.15 = " VER (&DUMPOPTS,NB)                                      "
 mem.16 = " VER (&DUMPVREL,NB,LIST,CICS530,CICS620,DFHPD530,DFHPD620)  "
 mem.17 = " VER (&DUMPEXIT,DSNAMEQ)                                 "
 mem.18 = " IF  (&DUMPEXIT = &Z)                                    "
 mem.19 = "      &DUMPEXIT = NONE                                   "
 mem.20 = ")END                                                     "
/*********************************************************************/
/* Display the Dynamic Panel                                         */
/*********************************************************************/
 call popdyn 'MEM' 4 'Quick CICS/IPCS Dump Formatting Options'
 call ispwrap 8 "VPUT (DUMPIPCS DUMPOPTS DUMPVREL DUMPEXIT) PROFILE"
/*********************************************************************/
/* Build the IPCS SETDEF and CICS VERBEXIT command                   */
/*********************************************************************/
```

```
 setdef = "SETDEF DD(IPCSDUMP)" dumpipcs
 verbx  = "VERBEXIT" dumpvrel",'"dumpopts"'"
/*********************************************************************/
/* Lock the display while IPCS is formatting                         */
/*********************************************************************/
 call lock 'Formatting IPCS options' verbx
/*********************************************************************/
/* Allocate a temporary IPCSDDIR VSAM dataset                        */
/*********************************************************************/
 udate = 'D'space(translate(date('O'),'','/'),Ø)
 utime = 'T'left(space(translate(time('L'),'',':.'),Ø),7)
 dumpddir = userid()'.TEMP.IPCSDDIR.'udate'.'utime
 if sysdsn(qdsn(dumpddir)) <> 'OK' then
    do
     call tsotrap "DEFINE CLUSTER(NAME("qdsn(dumpddir)") INDEXED",
                  "REUSE SHR(1 3) KEYS(128 Ø) RECORDSIZE(384 3Ø72)",
                  "CYLINDERS(3Ø 1Ø))"
    "IPCSDDIR" qdsn(dumpddir)
     IPCSRC = RC
     if IPCSRC <> Ø then
        call rcexit IPCSRC 'Error initializing the IPCSDDIR' dumpddir
    end
 call tsotrap "ALLOC F(IPCSDDIR) DA("qdsn(dumpddir)") SHR REUSE"
/*********************************************************************/
/* Allocate the DUMP and IPCS output                                 */
/*********************************************************************/
 call tsotrap "ALLOC F(IPCSDUMP) DA("qdsn(dumpdsn)") SHR REUSE"
 call tsotrap "ALLOC F(IPCSPRNT) UNIT(VIO) SPACE(5 1) CYLINDERS",
              "RECFM(F B A) LRECL(136) BLKSIZE(Ø)"
/*********************************************************************/
/* Queue the IPCS commands                                           */
/*********************************************************************/
 call stack 'UNLOAD'
 queue 'DROPDUMP DD(IPCSDUMP)'
 queue  setdef
 queue  verbx
 queue 'END'
/*********************************************************************/
/* Invoke IPCS                                                       */
/*********************************************************************/
 call outtrap 'garbage',Ø
 if dumpexit = 'NONE' then
   "IPCS NOPARM"
 else
   "IPCS TASKLIB("qdsn(dumpexit)")"
 call outtrap 'off'
 call stack 'RELOAD'
/*********************************************************************/
/* Unlock the screen                                                 */
/*********************************************************************/
```

```
 call unlock
/*********************************************************************/
/* Browse the IPCS output                                            */
/*********************************************************************/
 call brwsdd 'IPCSPRNT'
/*********************************************************************/
/* Free and delete                                                   */
/*********************************************************************/
 call tsotrap "FREE F(IPCSDUMP IPCSDDIR)"
 call tsoquiet "DELETE" qdsn(dumpddir)
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
 shutdown: nop
/*********************************************************************/
/* Put unique shutdown logic before the call to stdexit              */
/*********** @REFRESH BEGIN STOP     2002/08/03 08:42:33 *************/
/* Shutdown message and terminate                                    */
/*********************************************************************/
           call stdexit time('e')
/*********** @REFRESH END   STOP     2002/08/03 08:42:33 *************/
/*********** @REFRESH BEGIN SUBBOX   2004/03/10 01:25:03 *************/
/* 33 Internal Subroutines provided in CICSDUMP                      */
/* Last Subroutine REFRESH was 1 Sep 2004 18:12:29                   */
/* RCEXIT   - Exit on non-zero return codes                          */
/* TRAP     - Issue a common trap error message using rcexit         */
/* ERRMSG   - Build common error message with failing line number    */
/* STDENTRY - Standard Entry logic                                   */
/* STDEXIT  - Standard Exit logic                                    */
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/* DDCHECK  - Determine whether a required DD is allocated           */
/* DDLIST   - Returns number of DDs and populates DDLIST variable    */
/* DDDSNS   - Returns number of DSNs in a DD and populates DDDSNS    */
/* QDSN     - Make sure there are only one set of quotes             */
/* UNIQDSN  - Create a unique dataset name                           */
/* TEMPMEM  - EXECIO a stem into a TEMP PDS                          */
/* ISPWRAP  - Wrapper for ISPF commands                              */
/* ISRWRAP  - Wrapper for ISPF Edit commands                         */
/* TSOTRAP  - Capture the output from a TSO command in a stem        */
/* TSOQUIET - Trap all output from a TSO command and ignore failures */
/* WAIT     - Wait for a specified number of seconds                 */
/* SETBORD  - Set the ISPF Pop-up active frame border colour         */
/* LOCK     - Put up a pop-up under foreground ISPF during long waits*/
/* UNLOCK   - Unlock from a pop-up under foreground ISPF             */
/* PANDSN   - Create a unique PDS(MEM) name for a dynamic panel      */
/* POPDYN   - Addpop a Dynamic Panel                                 */
/* SAYDD    - Print messages to the requested DD                     */
/* BRWSDD   - Invoke ISPF Browse on any DD                           */
/* JOBINFO  - Get job-related data from control blocks               */
/* STACK    - UNLOAD, RELOAD, or LIST the Stack                      */
```

```
/* PTR      - Pointer to a storage location                          */
/* STG      - Return the data from a storage location                */
/* DEBUG    - Use debugging services                                 */
/* MODTRACE - Module Trace                                           */
/*********** @REFRESH END   SUBBOX   2004/03/10 01:25:03 *************/
/*********** @REFRESH BEGIN RCEXIT   2003/05/14 12:24:50 *************/
/* RCEXIT   - Exit on non-zero return codes                          */
/*-------------------------------------------------------------------*/
/* EXITRC   - Return code to exit with (if non-zero)                 */
/* ZEDLMSG  - Message text for it with for non-zero EXITRCs          */
/*********************************************************************/
 rcexit: parse arg EXITRC zedlmsg
         if EXITRC <> 0 then
            do
             trace 'o'
/*********************************************************************/
/* If execution environment is ISPF then VPUT ZISPFRC                */
/*********************************************************************/
             if execenv = 'TSO' | execenv = 'ISPF' then
                do
                 if ispfenv = 'YES' then
                    do
                     zispfrc = EXITRC
/*********************************************************************/
/* Does not call ISPWRAP to avoid obscuring error message modules    */
/*********************************************************************/
                      address ISPEXEC "VPUT (ZISPFRC)"
                    end
                end
/*********************************************************************/
/* If a message is provided, wrap it in date, time, and EXITRC       */
/*********************************************************************/
             if zedlmsg <> '' then
                do
                 zedlmsg = time('L') execname zedlmsg 'RC='EXITRC
                 call msg zedlmsg
                end
/*********************************************************************/
/* Write the contents of the Parentage Stack                         */
/*********************************************************************/
             stacktitle = 'Parentage Stack Trace ('queued()' entries):'
/*********************************************************************/
/* Write to MSGDD if background and MSGDD exists                     */
/*********************************************************************/
             if tsoenv = 'BACK' then
                do
                 if subword(zedlmsg,9,1) = msgdd then
                    do
                     say zedlmsg
                     signal shutdown
```

```
                  end
                else
                  do
                   call saydd msgdd 1 zedlmsg
                   call saydd msgdd 1 stacktitle
                  end
               end
            else
/*********************************************************************/
/* Write to the ISPF Log if foreground                               */
/*********************************************************************/
               do
                zerrlm = zedlmsg
                address ISPEXEC "LOG MSG(ISRZØØ3)"
                zerrlm = center(' 'stacktitle' ',78,'-')
                address ISPEXEC "LOG MSG(ISRZØØ3)"
               end
/*********************************************************************/
/* Unload the Parentage Stack                                        */
/*********************************************************************/
            do queued()
               pull stackinfo
               if tsoenv = 'BACK' then
                  do
                   call saydd msgdd Ø stackinfo
                  end
               else
                  do
                   zerrlm = stackinfo
                   address ISPEXEC "LOG MSG(ISRZØØ3)"
                  end
            end
/*********************************************************************/
/* Put a terminator in the ISPF Log for the Parentage Stack          */
/*********************************************************************/
            if tsoenv = 'FORE' then
               do
                zerrlm = center(' 'stacktitle' ',78,'-')
                address ISPEXEC "LOG MSG(ISRZØØ3)"
               end
/*********************************************************************/
/* Signal SHUTDOWN.  SHUTDOWN label MUST exist in the program        */
/*********************************************************************/
            signal shutdown
           end
         else
            return
/*********** @REFRESH END   RCEXIT   2ØØ3/Ø5/14 12:24:5Ø *************/
/*********** @REFRESH BEGIN TRAP     2ØØ2/Ø8/Ø7 11:48:14 *************/
/* TRAP     - Issue a common trap error message using rcexit         */
```

```
/*-------------------------------------------------------------------*/
/* PARM     - N/A                                                    */
/*********************************************************************/
 trap: traptype = condition('C')
       if traptype = 'SYNTAX' then
          msg = errortext(RC)
       else
          msg = condition('D')
       trapline = strip(sourceline(sigl))
       msg = traptype 'TRAP:' msg', Line:' sigl '"'trapline'"'
       call rcexit 666 msg
/*********** @REFRESH END   TRAP     2002/08/07 11:48:14 *************/
/*********** @REFRESH BEGIN ERRMSG   2002/08/10 16:53:04 *************/
/* ERRMSG   - Build common error message with failing line number    */
/*-------------------------------------------------------------------*/
/* ERRLINE  - The failing line number passed by caller from SIGL     */
/* TEXT     - Error message text passed by caller                    */
/*********************************************************************/
 errmsg: nop
         parse arg errline text
         return 'Error on statement' errline',' text
/*********** @REFRESH END   ERRMSG   2002/08/10 16:53:04 *************/
/*********** @REFRESH BEGIN STDENTRY 2004/08/28 21:10:25 *************/
/* STDENTRY - Standard Entry logic                                   */
/*-------------------------------------------------------------------*/
/* MSGDD    - Optional MSGDD used only in background                 */
/*********************************************************************/
 stdentry: module = 'STDENTRY'
           if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
           parse arg sparms
           push trace() time('L') module 'From:' sigl 'Parms:' sparms
           arg msgdd
           parse upper source . . execname . execdsn . . execenv .
/*********************************************************************/
/* Start-up values                                                   */
/*********************************************************************/
           EXITRC = 0
           MAXRC = 0
           ispfenv = 'NO'
           popup = 'NO'
           lockpop = 'NO'
           headoff = 'NO'
           hcreator = 'NO'
           keepstack = 'NO'
           lpar = mvsvar('SYSNAME')
           zedlmsg = 'Default shutdown message'
/*********************************************************************/
/* Determine environment                                             */
/*********************************************************************/
           if substr(execenv,1,3) <> 'TSO' & execenv <> 'ISPF' then
```

```
            tsoenv = 'NONE'
         else
            do
             tsoenv = sysvar('SYSENV')
             signal off failure
            "ISPQRY"
             ISPRC = RC
             if ISPRC = Ø then
                do
                 ispfenv = 'YES'
/*********************************************************************/
/* Check if HEADING ISPF table exists already, if so set HEADOFF=YES */
/*********************************************************************/
                 call ispwrap "VGET (ZSCREEN)"
                 if tsoenv = 'BACK' then
                    htable = jobinfo(1)||jobinfo(2)
                 else
                    htable = userid()||zscreen
                 TBCRC = ispwrap(8 "TBCREATE" htable "KEYS(HEAD)")
                 if TBCRC = Ø then
                    do
                     headoff = 'NO'
                     hcreator = 'YES'
                    end
                 else
                    do
                     headoff = 'YES'
                    end
                end
             signal on failure name trap
            end
/*********************************************************************/
/* MODTRACE must occur after the setting of ISPFENV                  */
/*********************************************************************/
         call modtrace 'START' sigl
/*********************************************************************/
/* Start-up message (if batch)                                       */
/*********************************************************************/
         startmsg = execname 'started' date() time() 'on' lpar
         if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
            headoff = 'NO' then
            do
             jobname = mvsvar('SYMDEF','JOBNAME')
             jobinfo = jobinfo()
             parse var jobinfo jobtype jobnum .
             say jobname center(' 'startmsg' ',61,'-') jobtype jobnum
             say
             if ISPRC = -3 then
                do
                 call saydd msgdd 1 'ISPF ISPQRY module not found,',
```

```
                                    'ISPQRY is usually in the LINKLST'
                  call rcexit 20 'ISPF ISPQRY module is missing'
                end
/*********************************************************************/
/* If MSGDD is provided, write the STARTMSG and SYSEXEC DSN to MSGDD */
/*********************************************************************/
             if msgdd <> '' then
                do
                 call ddcheck msgdd
                 call saydd msgdd 1 startmsg
                 call ddcheck 'SYSEXEC'
                 call saydd msgdd 0 execname 'loaded from' sysdsname
/*********************************************************************/
/* If there are PARMS, write them to the MSGDD                       */
/*********************************************************************/
                 if parms <> '' then
                    call saydd msgdd 0 'Parms:' parms
/*********************************************************************/
/* If there is a STEPLIB, write the STEPLIB DSN MSGDD                */
/*********************************************************************/
                 if listdsi('STEPLIB' 'FILE') = 0 then
                    do
                     steplibs = dddsns('STEPLIB')
                     call saydd msgdd 0 'STEPLIB executables loaded',
                         'from' word(dddsns,1)
                     if dddsns('STEPLIB') > 1 then
                        do
                         do stl=2 to steplibs
                            call saydd msgdd 0 copies(' ',31),
                                 word(dddsns,stl)
                         end
                        end
                    end
                end
             end
/*********************************************************************/
/* If foreground, save ZFKA and turn off the FKA display             */
/*********************************************************************/
          else
             do
              fkaset = 'OFF'
              call ispwrap "VGET (ZFKA) PROFILE"
              if zfka <> 'OFF' & tsoenv = 'FORE' then
                 do
                  fkaset = zfka
                  fkacmd = 'FKA OFF'
                  call ispwrap "CONTROL DISPLAY SAVE"
                 call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKACMD)"
                  call ispwrap "CONTROL DISPLAY RESTORE"
                 end
```

```
             end
/*********************************************************************/
           pull tracelvl . module . sigl . sparms
           call modtrace 'STOP' sigl
           interpret 'trace' tracelvl
           return
/*********** @REFRESH END   STDENTRY 2004/08/28 21:10:25 *************/
/*********** @REFRESH BEGIN STDEXIT  2004/08/02 06:06:40 *************/
/* STDEXIT  - Standard Exit logic                                    */
/*-------------------------------------------------------------------*/
/* ENDTIME  - Elapsed time                                           */
/* Note: Caller must set KEEPSTACK if the stack is valid             */
/*********************************************************************/
 stdexit: module = 'STDEXIT'
          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 endtime
          endmsg = execname 'ended' date() time() format(endtime,,1)
/*********************************************************************/
/* if MAXRC is greater then EXITRC then set EXITRC to MAXRC          */
/*********************************************************************/
          EXITRC = max(EXITRC,MAXRC)
          endmsg = endmsg 'on' lpar 'RC='EXITRC
          if tsoenv = 'BACK' & sysvar('SYSNEST') = 'NO' &,
             headoff = 'NO' then
             do
              say
              say jobname center(' 'endmsg' ',61,'-') jobtype jobnum
/*********************************************************************/
/* Make sure this isn't a MSGDD missing error then log to MSGDD      */
/*********************************************************************/
              if msgdd <> '' & subword(zedlmsg,9,1) <> msgdd then
                 do
                  call saydd msgdd 1 execname 'ran in' endtime 'seconds'
                  call saydd msgdd 0 endmsg
                 end
             end
/*********************************************************************/
/* If foreground, reset the FKA if necessary                         */
/*********************************************************************/
          else
             do
              if fkaset <> 'OFF' then
                 do
                  fkafix = 'FKA'
                  call ispwrap "CONTROL DISPLAY SAVE"
                  call ispwrap "DISPLAY PANEL(ISPBLANK) COMMAND(FKAFIX)"
                  if fkaset = 'SHORT' then
```

```
                  call ispwrap "DISPLAY PANEL(ISPBLANK)",
                               "COMMAND(FKAFIX)"
               call ispwrap "CONTROL DISPLAY RESTORE"
              end
          end
/*********************************************************************/
/* Clean up the temporary HEADING table                              */
/*********************************************************************/
       if ispfenv = 'YES' & hcreator = 'YES' then
          call ispwrap "TBEND" htable
/*********************************************************************/
/* Remove STDEXIT and MAINLINE Parentage Stack entries, if there     */
/*********************************************************************/
       call modtrace 'STOP' sigl
       if queued() > 0 then pull . . module . sigl . sparms
       if queued() > 0 then pull . . module . sigl . sparms
       if tsoenv = 'FORE' & queued() > 0 & keepstack = 'NO' then
          pull . . module . sigl . sparms
/*********************************************************************/
/* if the Parentage Stack is not empty, display its contents         */
/*********************************************************************/
       if queued() > 0 & keepstack = 'NO' then
          do
           say queued() 'Leftover Parentage Stack Entries:'
           say
           do queued()
              pull stackundo
              say stackundo
           end
           EXITRC = 1
          end
/*********************************************************************/
/* Exit                                                              */
/*********************************************************************/
       exit(EXITRC)
/*********** @REFRESH END   STDEXIT  2004/08/02 06:06:40 *************/
/*********** @REFRESH BEGIN MSG      2002/09/11 01:35:53 *************/
/* MSG      - Determine whether to SAY or ISPEXEC SETMSG the message */
/*-------------------------------------------------------------------*/
/* ZEDLMSG  - The long message variable                              */
/*********************************************************************/
 msg: module = 'MSG'
      parse arg zedlmsg
      if wordpos(module,probe) <> 0 then trace 'r'; else trace 'n'
      parse arg sparms
      push trace() time('L') module 'From:' sigl 'Parms:' sparms
      call modtrace 'START' sigl
/*********************************************************************/
/* If this is background or OMVS use SAY                             */
/*********************************************************************/
```

```
      if tsoenv = 'BACK' | execenv = 'OMVS' then
         say zedlmsg
      else
/*********************************************************************/
/* If this is foreground and ISPF is available, use SETMSG           */
/*********************************************************************/
         do
          if ispfenv = 'YES' then
/*********************************************************************/
/* Does not call ISPWRAP to avoid obscuring error message modules    */
/*********************************************************************/
             address ISPEXEC "SETMSG MSG(ISRZ000)"
          else
             say zedlmsg
         end
      pull tracelvl . module . sigl . sparms
      call modtrace 'STOP' sigl
      interpret 'trace' tracelvl
      return
/*********** @REFRESH END   MSG      2002/09/11 01:35:53 *************/
/*********** @REFRESH BEGIN DDCHECK  2002/09/11 01:08:30 *************/
/* DDCHECK  - Determine if a required DD is allocated                */
/*-------------------------------------------------------------------*/
/* DD       - DDNAME to confirm                                      */
/*********************************************************************/
 ddcheck: module = 'DDCHECK'
          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 dd
          dderrmsg = 'OK'
          LRC = listdsi(dd "FILE")
/*********************************************************************/
/* Allow sysreason=3 to verify SYSOUT DD statements                  */
/*********************************************************************/
          if LRC <> 0 & strip(sysreason,'L',0) <> 3 then
             do
              dderrmsg = errmsg(sigl 'Required DD' dd 'is missing')
              call rcexit LRC dderrmsg sysmsglvl2
             end
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return
/*********** @REFRESH END   DDCHECK  2002/09/11 01:08:30 *************/
/*********** @REFRESH BEGIN DDLIST   2002/12/15 04:54:32 *************/
/* DDLIST   - Returns number of DD's and populates DDLIST variable   */
/*-------------------------------------------------------------------*/
/* N/A      - None                                                   */
```

```
/*********************************************************************/
 ddlist: module = 'DDLIST'
         if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
/*********************************************************************/
/* Trap the output from the LISTA STATUS command                     */
/*********************************************************************/
         call outtrap 'lines.'
         address TSO "LISTALC STATUS"
         call outtrap 'off'
         ddnum = Ø
/*********************************************************************/
/* Parse out the DDNAMEs and concatenate into a list                 */
/*********************************************************************/
         ddlist = ''
         do ddl=1 to lines.Ø
            if words(lines.ddl) = 2 then
               do
                parse upper var lines.ddl ddname .
                ddlist = ddlist ddname
                ddnum = ddnum + 1
               end
            else
               do
                iterate
               end
         end
/*********************************************************************/
/* Return the number of DDs                                          */
/*********************************************************************/
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return ddnum
/*********** @REFRESH END   DDLIST   2ØØ2/12/15 Ø4:54:32 *************/
/*********** @REFRESH BEGIN DDDSNS   2ØØ2/Ø9/11 ØØ:37:36 *************/
/* DDDSNS   - Returns number of DSNs in a DD and populates DDDSNS    */
/*-------------------------------------------------------------------*/
/* TARGDD   - DD to return DSNs for                                  */
/*********************************************************************/
 dddsns: module = 'DDDSNS'
         if wordpos(module,probe) <> Ø then trace 'r'; else trace 'n'
         parse arg sparms
         push trace() time('L') module 'From:' sigl 'Parms:' sparms
         call modtrace 'START' sigl
         arg targdd
         if targdd = '' then call rcexit 77 'DD missing for DDDSNS'
/*********************************************************************/
```

```
/* Trap the output from the LISTA STATUS command                     */
/*********************************************************************/
         x = outtrap('lines.')
         address TSO "LISTALC STATUS"
         dsnnum = Ø
         ddname = '$DDNAME$'
/*********************************************************************/
/* Parse out the DDNAMEs, locate the target DD and concatentate DSNs */
/*********************************************************************/
         do ddd=1 to lines.Ø
            select
               when words(lines.ddd) = 1 & targdd = ddname &,
                    lines.ddd <> 'KEEP' then
                    dddsns = dddsns strip(lines.ddd)
               when words(lines.ddd) = 1 & strip(lines.ddd),
                    <> 'KEEP' then
                    dddsn.ddd = strip(lines.ddd)
               when words(lines.ddd) = 2 then
                    do
                     parse upper var lines.ddd ddname .
                     if targdd = ddname then
                        do
                         fdsn = ddd - 1
                         dddsns = lines.fdsn
                        end
                    end
               otherwise iterate
            end
         end
/*********************************************************************/
/* Get the last DD                                                   */
/*********************************************************************/
         ddnum = ddlist()
         lastdd = word(ddlist,ddnum)
/*********************************************************************/
/* Remove the last DSN from the list if not the last DD or SYSEXEC   */
/*********************************************************************/
         if targdd <> 'SYSEXEC' & targdd <> lastdd then
            do
             dsnnum = words(dddsns) - 1
             dddsns = subword(dddsns,1,dsnnum)
            end
/*********************************************************************/
/* Return the number of DSNs in the DD                               */
/*********************************************************************/
         pull tracelvl . module . sigl . sparms
         call modtrace 'STOP' sigl
         interpret 'trace' tracelvl
         return dsnnum
/*********** @REFRESH END   DDDSNS   2002/09/11 00:37:36 *************/
```

```
/*********** @REFRESH BEGIN QDSN     2002/09/11 01:15:23 *************/
/* QDSN     - Make sure there are only one set of quotes             */
/*-------------------------------------------------------------------*/
/* QDSN     - The DSN                                                */
/*********************************************************************/
 qdsn: module = 'QDSN'
       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 qdsn
       qdsn = "'"strip(qdsn,"B","'")"'"
       pull tracelvl . module . sigl . sparms
       call modtrace 'STOP' sigl
       interpret 'trace' tracelvl
       return qdsn
/*********** @REFRESH END   QDSN     2002/09/11 01:15:23 *************/
/*********** @REFRESH BEGIN UNIQDSN  2004/09/01 18:03:04 *************/
/* UNIQDSN  - Create a unique dataset name                           */
/*-------------------------------------------------------------------*/
/* PARM     - N/A                                                    */
/*********************************************************************/
 uniqdsn: module = 'UNIQDSN'
          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
/*********************************************************************/
/* Compose a DSN using userid, exec name, job number, date and time  */
/*********************************************************************/
          jnum = jobinfo(1) || jobinfo(2)
          udate = 'D'space(translate(date('O'),'','/'),0)
          utime = 'T'left(space(translate(time('L'),'',':.'),0),7)
          uniqdsn = userid()'.'execname'.'jnum'.'udate'.'utime
          if sysdsn(qdsn(uniqdsn)) = 'OK' then
             do
/*********************************************************************/
/* Wait 1 seconds to ensure a unique dataset (necessary on z990)     */
/*********************************************************************/
              RC = syscalls('ON')
              address SYSCALL "SLEEP 1"
              RC = syscalls('OFF')
              uniqdsn = uniqdsn()
             end
/*********************************************************************/
          pull tracelvl . module . sigl . sparms
          call modtrace 'STOP' sigl
          interpret 'trace' tracelvl
          return uniqdsn
/*********** @REFRESH END   UNIQDSN  2004/09/01 18:03:04 *************/
```

```
/*********** @REFRESH BEGIN TEMPMEM  2004/09/01 17:20:19 *************/
/* TEMPMEM  - EXECIO a stem into a TEMP PDS                          */
/*-------------------------------------------------------------------*/
/* TEMPMEM  - The member to create                                   */
/*********************************************************************/
 tempmem: module = 'TEMPMEM'
          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 tempmem
/*********************************************************************/
/* Create a unique DD name                                           */
/*********************************************************************/
          if length(tempmem) < 7 then
             tempdd = '$'tempmem'$'
          else
             tempdd = '$'substr(tempmem,2,6)'$'
/*********************************************************************/
/* If TEMPDD exists, then FREE it                                    */
/*********************************************************************/
          if listdsi(tempdd 'FILE') = 0 then "FREE F("tempdd")"
```

*Editor's note: the code will be concluded next month.*

*Robert Zenuk*
*Systems Programmer (USA)*


Please note that the correct contact address for Xephon Inc is PO Box 550547, Dallas, TX 75355, USA. The phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

GT Software has announced Ivory Web Services, which features Ivory Studio and Ivory Server. These provide Web services functionality to mainframe applications, and link mainframe data into SOAs (Service-Oriented Architectures).

Ivory Studio is a Windows PC-based development application for graphically building and publishing Web services from mainframe assets. Ivory Server is a mainframe-based SOAP server for deploying Web services. Ivory Server is a mainframe-based solution running CICS, and it gives users the ability to build Web services for existing mainframe applications and also allows users to access mainframe data.

A user site could expose information from a CICS-based insurance application on the mainframe as a Web service.

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

***

IBM has announced Version 5.3 of its Tivoli Monitoring for Transaction Performance, which allows organizations to see business transactions as they flow through an IT environment, and find out what systems the transactions use. The software then provides response times for each step. The product can help customers detect performance bottlenecks across their distributed computing environment.

Tivoli Monitoring for Transaction Performance 5.3 has been expanded to view Web services, Web servers, CICS, IMS, DB2, and SAP back-end services, as well as network delays between ARM-instrumented nodes. The new software also lets users group single transactions into a set of policy groups to boost usability in large computing systems.

For further information contact your local IBM representative.
URL: www-306.ibm.com/software/tivoli/products/monitor-transaction.

***

Sonata Software and Micro Focus have announced a strategic partnership for migrating mainframe-based CICS applications to Microsoft Windows technologies.

Micro Focus's 'Lift and Shift' approach helps customers migrate by re-using mainframe COBOL/CICS applications. Sonata provides a mainframe migration methodology, eNAbLE, which helps mainframe customers migrate.

For further information contact:
URL: www.microfocus.com.
URL: www.sonata-software.com.

***

A new version of CICS is in the works and is expected to be released early next year. It is expected to include greater automation in mainframe development and management environments.

This automation will include automating the interaction between CICS and J2EE.

For further information contact your local IBM representative.