225

CICS update

August 2004

In this issue

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

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

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs $270.00 in the USA and Canada; £175.00 in the UK; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the December 2000 issue, are available separately to subscribers for $24.00 (£16.00) each including postage.

CICS Update on-line

Code from *CICS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/cics; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

The CEKL transaction

CICS has recently been enhanced to provide support for the Kill function. This article describes the background to the CEKL transaction within CICS, and the means by which it can be used to issue inquiry commands and Kill requests in the CICS environment.

CEKL is provided as a means of invoking CICS task management functionality when other methods are not available. For example, a looping task may be preventing the QR TCB from being able to dispatch CEMT to inquire on, and then remove, the rogue task from the CICS system. While Kill support is available via CEKL, it is also implemented in the traditional areas of CICS system management, such as the CEMT task, CICS System Programming Interface (SPI), and the CICSPlex SM API. This article focuses on the role of CEKL, however.

CEKL

CEKL is intended to be used to remove tasks from a CICS system in a situation when this cannot be performed by existing methods, such as the CEMT transaction. Typically, this implies that the QR TCB is unavailable for other tasks to be dispatched under, and is most probably looping under the control of a rogue task within the CICS system.

CEKL can run from only the console interface. It provides the ability to inquire on tasks within a CICS system, and to set a task to be purged, forcepurged, or Killed. Issuing a CEKL INQUIRE TASK command returns information about selected tasks to the console. Issuing a CEKL SET TASK command allows the different means of task removal to be attempted against an individual task within CICS.

The CEKL transaction executes under its own dedicated CICS TCB (the CQ TCB). This allows CICS to process CEKL commands even when the rest of CICS is unable to respond to new input (if,

say, the QR TCB is not responding to new work).

CEKL may be started at an operating system console that has the authority to issue IBM MVS modify commands for a particular CICS system. Unlike normal CICS console access, CEKL does not require a CICS terminal definition for the relevant console. Standard MVS procedures and products may be used to issue these commands, eg SDSF.

Responses to CEKL commands are returned to the originating console in the normal way.

USING CEKL

An example of using CEKL to inquire on the task environment in a particular CICS system is shown below. In the example, the CICS jobname is TESTCICS. From a system console, the following command is entered:

```
F TESTCICS,CEKL INQUIRE TASK
```

The output from CEKL inquiries can be reduced by the specification of particular task numbers (if known in advance, of course). Other filtering restriction options include specifying particular transaction class (TRANCLASS) or transaction identifier (TRANSID) values. In addition, tasks can be selected based on whether they are dispatchable, running, or suspended within the CICS system.

The data that is returned includes the following information for a task. The task number is shown, which is needed when attempting to set a target task to be ended (ie by a subsequent purge, forcepurge, or Kill operation against it). The task's dispatcher state (dispatchable, running, or suspended) is shown. The TCB type that the task is executing under is given. This is either CK (for a CICS-key open TCB such as a J8 TCB for JVM programs), IN (for a CICS internal TCB, such as CO or RO), QR (for the quasi-reentrant TCB), or UK (for a user-key open TCB). If a task has not reached its point of initial dispatch under a TCB, the value is left blank. The response also returns a value to indicate whether a task is currently being purged, forcepurged, or killed, by displaying PUR, FOR, or KIL respectively. A blank here

indicates that none of these actions have been attempted against a task as yet. The transaction class and userid of the task are returned. The CPU time (in seconds) that has been consumed by this task is shown (unless monitoring is inactive, when ***** is shown instead). The time in seconds since the task was attached is returned. The runaway limit (in seconds) that CICS uses to monitor the task for a runaway condition is shown. A zero value here indicates that CICS is not monitoring the task for runaway. This could be because of the ICVR value for the CICS system being set to 0, or else the RUNAWAY setting for the task's transaction definition being defined as 0 instead. Finally, for suspended tasks only, the HTIME, HTYPE, and HVALUE values are returned too.

Having inquired on the tasks within a looping or stuck CICS system and identified the task that requires terminating, CEKL may also be used to issue a SET TASK command to attempt to remove the task from the system. An example of a console modify command invoking the CEKL transaction to do this is shown below. This example is using CEKL to invoke the Kill function against task 01310 within CICS.

```
F TESTCICS,CEKL SET TA(Ø131Ø) KILL
```

The options available are purge, forcepurge, or Kill. Given the importance of this function, and the fact that CEKL is expected to be used as a last resort, it is worthwhile clarifying the implications of its use. The following CICS documentation descriptions pertain to the use of each of these options from the CEKL transaction environment.

Purge

With purge the task is terminated. Task termination occurs only when system and data integrity can be maintained. There are situations when CICS will ignore a purge command in order to preserve integrity.

Forcepurge

With forcepurge the task is to be terminated. Data integrity is not

guaranteed. Before using forcepurge, you should use purge. In some cases, for example if a task is forcepurged during backout processing, CICS terminates abnormally. If you want to terminate a task but do not want to risk terminating CICS, you should use purge instead of forcepurge. There are situations when CICS will ignore a forcepurge command in order to preserve integrity.

Kill

With Kill the task is to be terminated. System and data integrity is not guaranteed. The Kill option extends the purge and forcepurge options. It should be used only after an attempt has been made to purge or forcepurge a task. The Kill option does not guarantee integrity of any kind, but in some situations it allows the user to free up a stalled region, enabling the region to continue processing. In some cases, for example, if a task is killed during backout processing, CICS terminates abnormally.

It should be noted that CEKL does not predicate that Kill requests be preceded by forcepurge attempts against target tasks. However, it is recommended that Kill be used only when purge and forcepurge requests have proven unable to remove a task from the system.

For each of the preceding options, if the target task is in any way associated with an open TCB (for example, it involves a Java program running on a J8 or H8 mode TCB, or an OpenAPI TRUE on a L8 TCB), a short delay could be experienced before the task is finally purged.

CEKL USAGE CONSIDERATIONS

There are a number of considerations to be aware of when using the Kill functionality within CICS from CEKL. These are summarized below.

Firstly, it should be emphasized that the use of Kill is intended as a last resort, when other means of task management have failed to alleviate the stuck or looping task(s). This is true when used from both CEKL and other means (such as CEMT). Secondly, it

is very important to remember that Kill cannot guarantee CICS system integrity; use of it may result in corruption of CICS control blocks and state data information. CICS itself may be terminated as a result of the use of Kill. Likewise, user data integrity may be affected as a result of using Kill against tasks within the CICS system. These points should be seriously considered before Kill utilization is attempted.

The CEKL transaction is not a true CICS transaction. One way that it differs from the other CICS-supplied transactions is that it may be invoked only from an operating system console. To avoid problems if the QR TCB is unavailable, the attachment of a CEKL task within CICS does not follow the normal task attach mechanism in the CICS Transaction Manager domain. Monitoring domain services are not utilized; neither are start-of-task and end-of-task TRUEs invoked. It is not safe to rely on the full-function code path within CICS that a normal transaction attachment has to follow. This is because any problems in these areas caused by the offending task(s) within the system could prevent the operation of CEKL to remove these tasks with Kill. As such, CICS is not able to provide transaction usage information for CEKL within its statistics reports or monitoring data.

CEKL is available for use only from the console. It does not run as a normal CICS transaction. As such, no CICS definitions are required for the CEKL transaction, and no CSD upgrade is required to provide the definitions for its use to CICS. Despite the fact that it is referred to as the CEKL transaction, it is very different from a traditional transaction (and the associated transactional environment) that runs within CICS.

Security for the use of the CEKL transaction is at the level of the MVS modify command itself. A console or user having MVS modify authority for a particular CICS region can execute the CEKL transaction there. CICS does not give restrictions that can be placed on the command; it is designed to be able to run under its own CQ TCB within CICS, even when the CICS environment is not responding. In practice this means when the QR TCB is unavailable because of some rogue task or loop situation. Since

CEKL is not a true transaction in the CICS sense, traditional CICS transaction security mechanisms cannot be applied to it.

Because the console is the interface to be used for CEKL usage, the various DFHCQ*xxxx* messages issued by CEKL to the console provide a means of tracking its use for audit purposes. These messages, together with further guidance on the use of the Kill function, will be discussed in a future article.

CONCLUSION

I hope this article has helped give a background to the implementation of CEKL as an aid to Kill support within CICS and offered some guidance on the ways in which it may be used.

Andy Wright (andy_wright@uk.ibm.com)
CICS Change Team Programmer
IBM (UK)

CICSPlex SM API – Assembler programs (command-level interface): part 2

This month we conclude the article looking at Assembler programs and the CICSPlex SM API.

```
*- A 9 Ø Ø _ T E R M I N A T I O N  : Termination                    -*
*- R2   DSECT - CM431COM                                            -*
*- R3   BASE CSECT CM431                                            -*
*- R8   DSECT - CM4Ø1COM                                            -*
*- R9   DSECT - CM4Ø2COM                                            -*
*- R1Ø  DSECT - CM4Ø3COM                                            -*
*- R11  EIBREG DSECT - DFHEIBLK                                     -*
*- R12  BASE CSECT CM431                                            -*
*- R13  DYNREG DSECT - DFHEISTG                                     -*
*- R14  Linkage                                                     -*
A9ØØ_TERMINATION  DS  ØH
         ST    R14,A9ØØSR14           SAVE REGISTER 14
*- Set Return Code                                                 -*
         MVC   CM431COM_RC,RETC       SET RC FOR CALLER
*- Return to caller                                                -*
```

```
A900RET  L     R14,A900SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- Z 4 0 1 _ I S S U E _ M E S S A G E : Write Message to TDQ        -*
*- R3   BASE CSECT                                                  -*
*- R8   DSECT - CM401COM                                            -*
*- R9   DSECT - CM402COM                                            -*
*- R10  DSECT - CM403COM                                            -*
*- R11  EIBREG DSECT - DFHEIBLK                                     -*
*- R12  BASE CSECT                                                  -*
*- R13  DYNREG DSECT - DFHEISTG                                     -*
*- R14  Linkage                                                     -*
Z401_ISSUE_MESSAGE  DS  0H
         ST    R14,Z401SR14            SAVE REGISTER 14
*- LINK to CM401 - Write Message to TDQ                            -*
Z401LINK EXEC  CICS LINK                                             X
               PROGRAM('CM401')                                      X
               COMMAREA(CM401COM)                                    X
               LENGTH(L'CM401ST)                                     X
               RESP(CM402COM_RESP1)                                  X
               RESP2(CM402COM_RESP2)
*- Check EXEC response, if there is an error "try" to issue a       -*
*- WTO and then abend, because this indicates a serious problem.    -*
         MVC   CM402COM_FUNC(L'CM402COM_FUNC),EIBFN   MOVE FUNCTION
         CLC   CM402COM_RESP1,DFHRESP(NORMAL)  NORMAL COMPLETION?
         BE    Z401RET                 YES - BRANCH BACK
*                                      NO  - PROCESS ERROR
         BAL   R14,Z402_CICS_DIAGS     PERFORM CICS DIAGNOSTICS
         MVC   CM403COM_MSGNO,=CL(L'CM403COM_MSGNO)'     401E' NUMBER
         MVC   CM403COM_MSGNO(5),ASMPROG                       PROGRAM
         MVC   CM403COM_TEXT,=C'Link to CM401 failed.'         TEXT
         BAL   R14,Z403_ISSUE_WTO      ISSUE WTO
Z401ABND EXEC  CICS ABEND                                            X
               ABCODE(,Z401')
*- Return to caller                                                -*
Z401RET  L     R14,Z401SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- Z 4 0 2 _ C I C S _ D I A G S :  Perform CICS Diagnostics        -*
*- R3   BASE CSECT                                                  -*
*- R8   DSECT - CM401COM                                            -*
*- R9   DSECT - CM402COM                                            -*
*- R11  EIBREG DSECT - DFHEIBLK                                     -*
*- R12  BASE CSECT                                                  -*
*- R13  DYNREG DSECT - DFHEISTG                                     -*
*- R14  Linkage                                                     -*
Z402_CICS_DIAGS  DS  0H
         ST    R14,Z402SR14            SAVE REGISTER 14
*- LINK to CM402 - Diagnose CICS Error.                            -*
Z402LINK EXEC  CICS LINK                                             X
```

```
                PROGRAM('CM402')                                        X
                COMMAREA(CM402COM)                                      X
                LENGTH(L'CM402ST)                                       X
                RESP(Z402_RESP1)                                        X
                RESP2(Z402_RESP2)
*- Check EXEC response, if there is an error "try" to issue a       -*
*- message and then abend, because this indicates a serious problem.-*
          CLC   Z402_RESP1,DFHRESP(NORMAL)      NORMAL COMPLETION?
          BE    Z402RET                 YES - RETURN TO CALLER
*                                       NO  - ISSUE ERROR MESSAGE
          MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'     402E' NUMBER
          MVC   CM401COM_MSGNO(5),ASMPROG                       PROGRAM
          MVC   CM401COM_TEXT,=C'Link to CM402 failed.'         TEXT
          BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
Z402ABND EXEC  CICS ABEND                                               X
                ABCODE(,Z402')
*- Return to caller                                                 -*
Z402RET   L     R14,Z402SR14            RESTORE REGISTER 14
          BR    R14                     RETURN TO CALLER
          EJECT
*- Z 4 0 3 _ I S S U E _ W T O :  Write To Operator Message          -*
*- R3   BASE CSECT                                                  -*
*- R8   DSECT - CM401COM                                            -*
*- R9   DSECT - CM402COM                                            -*
*- R10  DSECT - CM403COM                                            -*
*- R11  EIBREG DSECT - DFHEIBLK                                     -*
*- R12  BASE CSECT                                                  -*
*- R13  DYNREG DSECT - DFHEISTG                                     -*
*- R14  Linkage                                                     -*
Z403_ISSUE_WTO  DS 0H
          ST    R14,Z403SR14            SAVE REGISTER 14
*- LINK to CM403 - Write To Operator                                -*
Z403LINK EXEC  CICS LINK                                                X
                PROGRAM('CM403')                                        X
                COMMAREA(CM403COM)                                      X
                LENGTH(L'CM403ST)                                       X
                RESP(CM402COM_RESP1)                                    X
                RESP2(CM402COM_RESP2)
*- Check EXEC response, if there is an error "try" to issue a       -*
*- message and then abend, because this indicates a serious problem.-*
          MVC   CM402COM_FUNC(L'CM402COM_FUNC),EIBFN   MOVE FUNCTION
          CLC   CM402COM_RESP1,DFHRESP(NORMAL)  NORMAL COMPLETION?
          BE    Z403RET                 YES - BRANCH BACK
*                                       NO  - PROCESS ERROR
          BAL   R14,Z402_CICS_DIAGS     PERFORM CICS DIAGNOSTICS
          MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'     403E' NUMBER
          MVC   CM401COM_MSGNO(5),ASMPROG                       PROGRAM
          MVC   CM401COM_TEXT,=C'Link to CM403 failed.'         TEXT
          BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
Z403ABND EXEC  CICS ABEND                                               X
```

```
               ABCODE(,Z4Ø3')
*- Return to caller                                                -*
Z4Ø3RET  L     R14,Z4Ø3SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- Z 9 Ø Ø _ D U M P _ T R A N : Dump Transaction                  -*
*- R3   BASE CSECT CM431                                           -*
*- R11  EIBREG DSECT - DFHEIBLK                                    -*
*- R12  BASE CSECT CM431                                           -*
*- R13  DYNREG DSECT - DFHEISTG                                    -*
*- R14  Linkage                                                    -*
Z9ØØ_DUMP_TRAN  DS  ØH
         ST    R14,Z9ØØSR14            SAVE REGISTER 14
Z9ØØDUMP EXEC  CICS DUMP TRANSACTION                                X
               DUMPCODE(DUMPCODE)
*- Return to caller                                                -*
Z9ØØRET  L     R14,Z9ØØSR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- Z 9 Ø 1 _ C P S M _ D I A G S :  Perform CPSM Diagnostics       -*
*- R3   BASE CSECT                                                 -*
*- R4   Work Register                                              -*
*- R8   DSECT - CM4Ø1COM                                           -*
*- R11  EIBREG DSECT - DFHEIBLK                                    -*
*- R12  BASE CSECT                                                 -*
*- R13  DYNREG DSECT - DFHEISTG                                    -*
*- R14  Linkage                                                    -*
Z9Ø1_CPSM_DIAGS  DS  ØH
         ST    R14,Z9Ø1SR14            SAVE REGISTER 14
*- Convert RESPONSE and REASON Codes to displayable characters.    -*
         MVC   CM4Ø1COM_MSGNO,=CL(L'CM4Ø1COM_MSGNO)'     9Ø1E' NUMBER
         MVC   CM4Ø1COM_MSGNO(5),ASMPROG                       PROGRAM
         MVC   CM4Ø1COM_TEXT(35),=C'Response(        ) Reason(        )X
               '
         L     R4,Z9Ø1_RESPONSE        LOAD RESPONSE CODE
         CVD   R4,Z9Ø1_D1              CONVERT TO DECIMAL
         UNPK  Z9Ø1_D2,Z9Ø1_D1         CONVERT TO ...
         OI    Z9Ø1_D2+7,X'FØ'         ... DISPLAYABLE DECIMAL
         MVC   CM4Ø1COM_TEXT+9(L'Z9Ø1_D2),Z9Ø1_D2            RESPONSE
         L     R4,Z9Ø1_REASON          LOAD REASON CODE
         CVD   R4,Z9Ø1_D1              CONVERT TO DECIMAL
         UNPK  Z9Ø1_D2,Z9Ø1_D1         CONVERT TO ...
         OI    Z9Ø1_D2+7,X'FØ'         ... DISPLAYABLE DECIMAL
         MVC   CM4Ø1COM_TEXT+26(L'Z9Ø1_D2),Z9Ø1_D2           RESPONSE
         BAL   R14,Z4Ø1_ISSUE_MESSAGE  ISSUE MESSAGE
*- Return to caller                                                -*
Z9Ø1RET  L     R14,Z9Ø1SR14            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                                                  -*
* None!
         EJECT
*- T A B L E S                                                        -*
* None !!
         EJECT
*- M E S S A G E S                                                    -*
CM431110E  DC  C'Error processing ASSIGN command.'
CM431113E  DC  C'Invalid Invocation - No COMMAREA provided.'
CM431200E  DC  C'Error CONNECTing to CICSPlex SM.'
CM431300E  DC  C'Error GETting Object TRANCLAS.'
CM431400E  DC  C'Error FETCHing Object TRANCLAS.'
CM431410E  DC  C'Error writing to TSQ.'
CM431500E  DC  C'Error TERMINATing CICSPlex SM API.'
         EJECT
*- E N D   CM431                                                      -*
         END   CM431
```

CM431A01

```
************************************************************************
*                 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 = CM431A01                                               *
* MODULE TYPE = DSECT CM431COM                                         *
* DESCRIPTION = Communications Area for CM431 (Assembler)              *
         EJECT
* CHANGE HISTORY:                                                      *
         EJECT
* C M 4 3 1    -     Communications Area                               *
CM431COM           DSECT
CM431COM_ALIGN     DS    0D            ALIGNMENT
CM431COM_EYECATCH  DS    CL16          EYECATCHER
CM431COM_TSQ       DS    CL16          TEMPORARY STORAGE QUEUE
CM431COM_RECORDS   DS    F             NUMBER OF RECORDS
CM431COM_RC        DS    F             RETURN CODE
CM431COM_CONTEXT   DS    CL8           CPSM CONTEXT
CM431COM_SCOPE     DS    CL8           CPSM SCOPE
CM431COM_CRITERIA_L DS   F             LENGTH OF CRITERIA
CM431COM_CRITERIA  DS    CL160         CRITERIA
CM431COM_RESERVE   DS    CL36          RESERVED/FILLER = 256 BYTES
CM431COM_LENGTH    EQU   *-CM431COM    LENGTH OF CM431COM
*- E N D   CM431A01                                                   -*
```

CM431A02

```
************************************************************************
```

```
*                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 = CM431AØ2                                           *
* MODULE TYPE = DSECT CM431TSQ                                     *
* DESCRIPTION = Temporary Storage Queue for CM431 (Assembler)      *
         EJECT
* CHANGE HISTORY:                                                  *
         EJECT
* C M 4 3 1    -     Temporary Storage Queue                       *
CM431TSQ           DSECT
CM431TSQ_REGION    DS    CL8           CICS REGION NAME
CM431TSQ_TRANCLAS  DS    CL8           CICS TRANSACTION CLASS
CM431TSQ_MAXACTIVE DS    F             MAX. TRANSACTIONS ALLOWED
CM431TSQ_ACTIVE    DS    F             NO. ACTIVE TRANSACTIONS
CM431TSQ_QUEUED    DS    F             NO. TRANSACTIONS QUEUED
CM431TSQ_PURGETHRESH DS  F             PURGE THRESHOLD
CM431TSQ_LENGTH    EQU   *-CM431TSQ    LENGTH OF CM431TSQ
*- E N D   CM431AØ2                                             -*
```

CM431C01

```
      ******************************************************************
      *              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 = CM431CØ1                                         *
      * MODULE TYPE = Copybook - CM431COM                              *
      * DESCRIPTION = Communications Area for CM431 (Cobol)            *
           EJECT
      * CHANGE HISTORY:                                                *
           EJECT
      * C M 4 3 1    -     Communications Area                         *
       Ø1  CM431COM.
           Ø2  CM431COM-EYECATCH           PIC X(16) VALUE SPACES.
           Ø2  CM431COM-TSQ                PIC X(16) VALUE SPACES.
           Ø2  CM431COM-RECORDS            PIC S9(8) COMP VALUE ZERO.
           Ø2  CM431COM-RC                 PIC S9(8) COMP VALUE ZERO.
           Ø2  CM431COM-CONTEXT            PIC  X(8) VALUE SPACES.
           Ø2  CM431COM-SCOPE              PIC  X(8) VALUE SPACES.
           Ø2  CM431COM-CRITERIA-L         PIC S9(8) COMP VALUE ZERO.
           Ø2  CM431COM-CRITERIA           PIC X(16Ø) VALUE SPACES.
           Ø2  CM431COM-RESERVE            PIC X(Ø36) VALUE SPACES.
      *- E N D   CM431CØ1                                           -*
           EJECT
```

CM431C02

```
      ******************************************************************
```

```
*             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 = CM431C02                                       *
* MODULE TYPE = Copybook - CM431TSQ                            *
* DESCRIPTION = Temporary Storage Queue for CM431 (Cobol)      *
     EJECT
* CHANGE HISTORY:                                              *
     EJECT
* C M 4 3 1    -     Temporary Storage Queue                   *
 01  CM431TSQ.
     02  CM431TSQ-REGION             PIC  X(8) VALUE SPACES.
     02  CM431TSQ-TRANCLAS           PIC  X(8) VALUE SPACES.
     02  CM431TSQ-MAXACTIVE          PIC S9(8) COMP VALUE ZERO.
     02  CM431TSQ-ACTIVE             PIC S9(8) COMP VALUE ZERO.
     02  CM431TSQ-QUEUED             PIC S9(8) COMP VALUE ZERO.
     02  CM431TSQ-PURGETHRESH        PIC S9(8) COMP VALUE ZERO.
*- E N D   CM431C02                                           -*
     EJECT
```

Programs CM401, CM402, CM403, and all their copybooks are used as common routines in this article. They are available from the Xephon Web site. Programs CM430/CM431 and CM432/CM433 are functional pairs. In true technical and programming terms, both pairs are very similar with the exception that they process a different resource type. One pair has been published in full, and the second pair is available on the Web site. All the files can be downloaded from www.xephon.com/extras/CPSMAPI.txt.

Carl Wade McBurnie
IT Consultant (Germany)

Automatic CICS PPT management using CICS statistics and autoinstall function

INTRODUCTION

This article talks about our experience dealing with the boring problem of CICS PPT management. We hope it will help people who have the same problem.

OBJECTIVES

Our objective was to combine the flexibility of the program autoinstall function with the control over PPT definitions given by RDO.

We wanted to save on manual work (my work!), time, and money, by managing the RDO definition process with an automatic procedure. A few programs are excluded from this management (see below).

We hoped to delegate the change management system, where all application programs are defined. The task of maintaining the information needed to manage the PPT in RDO frees us from managing RDO definitions with another product.

Furthermore, by defining in RDO format only those programs that are ‘statistically’ used in a CICS, we gain the following benefits:

- Better CICS performance (small overhead for autoinstall activity during CICS life).
- Memory savings (no useless PPT entries).
- Faster CICS start (fewer PPT entries to install means a faster start up).

Our environment was:

- z/OS 1.3 (not relevant to our procedure).
- CICS Transaction Server 1.3 (same as above).
- Endevor as our change management system (key for our procedures).

Our project requirements were:

- Endevor.
- The program autoinstall function enabled on CICS regions.
- CICS statistics collection enabled.

DESCRIPTION

In our shop, CICS statistics are downloaded daily and expanded to produce a sequential file, where for each CICS system there is a list of all the defined programs plus a use count during the previous month (the CONT_MP field). Our PPT management procedure starts from that information. An extract from our CICS program statistics sequential file is shown below:

```
----+----1----+----2----+----3---- [...] 9----+----+-- [...]
UTILIZZO PROGRAMS CICS
APPLID    LDRPNAME      COUNT                CONT_MP
          ========      =====                =======
          NOME          N. VOLTE             CONTATORE
          PROGRAM       CHIAMATO             MESE
                        DA TRANS             PRECEDENT
[...]
CXCRP1T4  XFHMSES1      53                   128296
[...]
```

CICS statistics tell us which programs have been used in a CICS system during the previous month. These programs are defined in our Endevor environment with a system, subsystem, and type. This information is all we need to make an RDO definition statement for a program. We use an Endevor program to make a report of its managed programs. This report will be used to load a VSAM file that we call User Master Control File (UMCF), which will be read by MCFREPRT to make searches faster than reading the report sequential file (USERMCF JCL). A record in our Endevor user MCF VSAM file is shown below:

```
+------------ key ------------+
----+----1----+----2----+----3-- -+----4 [...] -8
[...]
XFHMSES1 XC      TP      PTC
[...]
```

We use two-character systems and two-character subsystems to create the RDO group name and the Endevor type to make an RDO definition statement for a program or for a mapset.

The RDO group names are created following the standard, ssbb*PA*ii, where:

- ss is the Endevor system where the program is defined.

- bb is the Endevor subsystem where the program is defined.
- *P* denotes that the group contains a program definition.

In addition to RDO DEFINE statements, the CRTRDOCB program also makes ADD statements for the list you have passed to it. An extract of CRTRDOCB output is shown below:

```
[...]
DEF PROG(MPXCINS ) G(XCTPP)
ADD G(XCTPP) LI(CELISTT4)
[...]
DEF PROG(XFHMSES1) G(XCTPP)
[...]
```

Last month's programs are defined in our Automatic RDO Define Procedure (ARDP) in RDO format into a CSD, and so they are installed when CICS starts. Programs that aren't installed when CICS starts, because they are new or they weren't used last month (there are statistically a few), will be installed dynamically by the autoinstall function. Next month they will be referenced in the CICS statistics and will be defined statically in CSD by our procedure.

Very few definitions are excluded from this system, but they include those referencing our system programs (CICS user EXITs, programs used in PLTPI and PLTSD, etc), those pertaining to products (MERVA, OMEGAMON, etc), and those few belonging to programs with particular attributes like REMOTESYSTEM or EXECKEY(CICS).

This procedure is built from two JCLs. The first, named UMCF, creates the UMCF VSAM file from the Endevor report. The second, named ARDP, uses CICS statistics and the UMCF VSAM file to update the CICS CSD file.

The UMCF JCL is shown below:

```
//*********************************************************************
//UMCF      PROC DSNSYSIN='CR1Ø157.ENDEVOR.RDO',
//           CONLIB='SYS1.SSH.ENDEVOR.BPSYA.SP.LOADLIB.APF',
//           DSNPRFX='DA.CR1Ø157'
//*********************************************************************
//* DELETE OLD DATASETS
```

```
//*********************************************************************
//DELETE    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.DL)
//*********************************************************************
//* ENDEVOR REPORT
//*********************************************************************
//REPORT    EXEC PGM=NDVRC1,PARM=C1BR1000,REGION=4096K
//CONLIB    DD DISP=SHR,DSN=&CONLIB
//BSTRPTS   DD DISP=(NEW,CATLG,DELETE),            REPORT OUTPUT
//          DCB=(LRECL=133,RECFM=FBA,BLKSIZE=26600),
//          SPACE=(CYL,(5,15),RLSE),
//          DSN=&DSNPRFX..&ENDENV..REPORT03
//BSTINP    DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.RP)  SELECTION CRITERIA
//BSTPDS    DD DUMMY                               FOOTPRINT DATA SET
//BSTIPT    DD DUMMY                               FOOTPRINT CRITERIA
//SMFDATA   DD DUMMY                               SMF DATA SET
//UNLINPT   DD DUMMY                               UNLOAD DATA SET
//BSTPCH    DD DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(CYL,(300,50)),
//          DCB=(RECFM=FB,LRECL=416,BLKSIZE=4160),
//          DSN=&TEMP
//BSTLST    DD SYSOUT=*
//SORTIN    DD UNIT=(SYSDA,2),SPACE=(CYL,(100,150)),VOL=(,,,2)
//SORTOUT   DD UNIT=(SYSDA,2),SPACE=(CYL,(100,150)),VOL=(,,,2)
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(300,50))
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(300,50))
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(300,50))
//C1MSGS1   DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//FOOTDD    DD DUMMY
//*********************************************************************
//* EXTRACT INFO FROM ENDEVOR REPORT
//*********************************************************************
//SORTRPRT  EXEC PGM=SORT
//SORTWK01  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK02  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK03  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK04  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK05  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK06  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK07  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK08  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SORTWK09  DD UNIT=SYSDA,SPACE=(CYL,(100,10))
//SYSOUT    DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=&DSNPRFX..&ENDENV..REPORT03
//SORTOUT   DD DISP=(NEW,PASS),
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=FB,BLKSIZE=27920,LRECL=80),UNIT=SYSDA,
```

```
//          DSN=&SRTRPRT
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(#UMCFSRT)
//*******************************************************************
//* MAKE USER MCF VSAM
//*******************************************************************
//DEFUMCF   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.DF)
//*******************************************************************
//* REPRO ON USER MCF VSAM
//*******************************************************************
//REPUMCF   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//DDIN      DD DISP=SHR,DSN=&SRTRPRT
//DDOUT     DD DISP=OLD,DSN=&DSNPRFX..&ENDENV..USERMCF
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(#UMCFREP)
//*******************************************************************
//          PEND
//*******************************************************************
//MVSRIL    EXEC USERMCF,ENDENV=MVSRIL
//*******************************************************************
```

Firstly it deletes the old Endevor report sequential file and user MCF VSAM file. The &ENDENV.DL (MVSRILDL) member used as sysin for the DELETE JLC step is shown below:

```
DELETE DA.CR1Ø157.MVSRIL.REPORTØ3
DELETE DA.CR1Ø157.MVSRIL.USERMCF CL PURGE
SET MAXCC=Ø
```

Next, the Endevor report utility uses the &ENDENV.RP (MVSRILRP) member as sysin for the REPORT JLC step:

```
    REPORT      Ø3 .
    ENVIRONMENT MVSRIL .
    STAGE       R .
```

The SORTRPRT SORT extracts from the Endevor report just the records we want, ie CICS COBOL programs (type PTC in our Endevor installation), CICS Assembler programs (type PTA), batch COBOL programs (type PBC; it could be a generic routine used in CICS too), batch Assembler programs (type PBA), and CICS maps (type MAP). The #UMCFSRT member used as sysin for the SORTRPRT JLC step:

```
INCLUDE COND=(43,3,CH,EQ,C'PTA',OR,
              43,3,CH,EQ,C'PTC',OR,
              43,3,CH,EQ,C'PBA',OR,
```

```
              43,3,CH,EQ,C'PBC',OR,
              43,3,CH,EQ,C'MAP')
 SORT FIELDS=(1,49,CH,A)
 OUTREC FIELDS=(2,8,C' ',24,8,34,8,43,8)
```

The DEFUMCF step defines the user MCF. The &ENDENV.DF (MVSRILDF) member is used as sysin for the DEFUMCF JLC step:

```
 DEFINE CLUSTER -
           (NAME(DA.CR10157.MVSRIL.USERMCF) -
           INDEXED -
           SHAREOPTIONS(3,3) -
           NOERASE -
           SPEED -
           NOWRITECHECK -
           NOREUSE) -
        DATA -
           (NAME(DA.CR10157.MVSRIL.USERMCF.DATA) -
           KEYS(33 0) -
           CONTROLINTERVALSIZE(8192) -
           RECORDSIZE(80 80) -
           CYLINDER(10 5) -
           NONSPANNED -
           VOLUMES(DA0003)) -
        INDEX -
           (NAME(DA.CR10157.MVSRIL.USERMCF.INDEX) -
           CONTROLINTERVALSIZE(2048) -
           TRACK(15 5) -
           NOIMBED -
           UNORDERED -
           NOREPLICATE -
           VOLUMES(DA0003))
```

The REPUMCF step loads the user MCF with those records extracted from the Endevor report by the SORTRPRT step. The #UMCFREP member is used as sysin for the REPUMCF JLC step:

```
 REPRO INFILE(DDIN) OUTFILE(DDOUT)
```

The ARDP JCL looks like:

```
//*******************************************************************
//ARDP      PROC DSNSYSIN='DA.CR10157.ENDEVOR.SOURCE',
//           STATEPGM='BTD.XC77.XCW00P17.STAT.EPGM',
//           LOADLIB='CR10157.LOAD',
//           DSNPRFX='DA.CR10157',
//           SDFHLOAD='PRD.CICSTS.TP.SP.SDFHLOAD',
```

```
//          DFHCSD='DA.CR1Ø157.DFHCSD'
//*******************************************************************
//DELETE    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENV.&CICSID.DEL)
//*******************************************************************
//SORTEPGM  EXEC PGM=SORT
//SORTWKØ1  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ2  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ3  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ4  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ5  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ6  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ7  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ8  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ9  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SYSOUT    DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=&STATEPGM
//SORTOUT   DD DISP=(NEW,PASS),
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=FB,BLKSIZE=266Ø,LRECL=133),UNIT=SYSDA,
//          DSN=&SORTEPGM
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENV.&CICSID.SORT)
//*******************************************************************
//MCFREPRT  EXEC PGM=MCFREPRT,PARM='RDO'
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB
//UMCF      DD DISP=SHR,DSN=&DSNPRFX..&ENDENV..USERMCF
//EPGM      DD DISP=SHR,DSN=&SORTEPGM
//PROGLIST  DD DISP=(NEW,PASS),
//          DCB=(LRECL=8Ø,RECFM=FB,BLKSIZE=2792Ø),
//          SPACE=(CYL,(1,1),RLSE),UNIT=SYSDA,
//          DSN=&PROGLIST
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSMDUMP  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSABOUT  DD SYSOUT=*
//*******************************************************************
//SORTPROG  EXEC PGM=SORT
//SORTWKØ1  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ2  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ3  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ4  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ5  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ6  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ7  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ8  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ9  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SYSOUT    DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=&PROGLIST
```

```
//SORTOUT   DD DISP=(,PASS),
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=FB,BLKSIZE=27920,LRECL=80),UNIT=SYSDA,
//          DSN=&SORTPROG
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(#ARDPSRT)
//*******************************************************************
//CRTRDOCB  EXEC PGM=CRTRDOCB,PARM='&RDOLIST'
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB
//PROGLIST  DD DISP=SHR,DSN=&SORTPROG
//RDODEF    DD DSN=&DSNPRFX..RDO.DEFINE.&ENV.&CICSID,
//          DISP=(,CATLG,DELETE),
//          DCB=(LRECL=80,RECFM=FB,BLKSIZE=0),
//          SPACE=(CYL,(1,1),RLSE),UNIT=SYSDA
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSMDUMP  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSABOUT  DD SYSOUT=*
//*******************************************************************
//CSDUPDT   EXEC PGM=DFHCSDUP,REGION=1024K
//STEPLIB   DD DISP=SHR,DSN=&SDFHLOAD
//DFHCSD    DD DISP=SHR,DSN=&DFHCSD
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNPRFX..RDO.DEFINE.&ENV.&CICSID
//*******************************************************************
//          PEND
//*******************************************************************
//P1T4      EXEC ARDP,ENV=P1,CICSID=T4,ENDENV=MVSRIL,RDOLIST=CELISTT4
//*******************************************************************
```

The first step deletes old output. The &ENV.&CICSID.DEL (P1T4DEL) member is used as sysin for the DELETE JLC step:

```
 DELETE DA.CR10157.RDO.DEFINE.P1T4
 SET MAXCC=0
```

From the CICS statistics sequential file we get only those records related to a particular CICS applid, with the CONT_MP field not null and not CICS (DFH...) modules. The &ENV.&CICSID.SORT (P1T4SORT) member is used as sysin for the SORTEPGM JCL step:

```
INCLUDE COND=(1,8,CH,EQ,C'CXCRP1T4',
              AND,95,1,CH,NE,C'0',AND,95,1,CH,NE,C'.',
              AND,11,3,CH,NE,C'DFH')
 OPTION COPY
```

The MCFREPRT program reads a list of programs and the

Endevor MCF searching for systems and subsystems:

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.                MCFREPRT.
       AUTHOR.            GIANLUCA BONZANO.
      ****************************************************************
     *                                                                *
      *----------------------------------------------------------------*
      ****************************************************************
       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
               DECIMAL-POINT IS COMMA.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT  UMCF-FILE       ASSIGN TO UMCF
                                   ORGANIZATION IS INDEXED
                                   ACCESS MODE IS DYNAMIC
                                   RECORD KEY IS UMCF-KEY
                                   FILE STATUS IS UMCF-FILE-STATUS.
           SELECT  EPGM-FILE       ASSIGN TO EPGM
                                   ORGANIZATION IS SEQUENTIAL
                                   FILE STATUS IS EPGM-FILE-STATUS.
           SELECT  PROGLIST-FILE   ASSIGN TO PROGLIST
                                   ORGANIZATION IS SEQUENTIAL
                                   FILE STATUS IS PROGLIST-FILE-STATUS.
       DATA DIVISION.
       FILE SECTION.
       FD  UMCF-FILE
           DATA RECORD IS UMCF-REC.
       Ø1  UMCF-REC.
           Ø5  UMCF-KEY.
               Ø7  UMCF-KEY-PROGNAME       PIC X(8).
               Ø7  FILLER                  PIC X(1).
               Ø7  UMCF-KEY-SYSTEM         PIC X(8).
               Ø7  UMCF-KEY-SUBSYSTEM      PIC X(8).
               Ø7  UMCF-KEY-TYPE           PIC X(8).
           Ø5  FILLER                      PIC X(47).
       FD  EPGM-FILE
           RECORDING MODE IS F
           RECORD CONTAINS 133 CHARACTERS
           DATA RECORD EPGM-REC.
       Ø1  EPGM-REC.
           Ø5  EPGM-APPLID                 PIC X(8).
           Ø5  FILLER                      PIC X(2).
           Ø5  EPGM-LDRPNAME               PIC X(8).
           Ø5  FILLER                      PIC X(76).
           Ø5  EPGM-CONT-MP                PIC X(8).
           Ø5  FILLER                      PIC X(31).
       FD  PROGLIST-FILE
```

```
    RECORDING MODE IS F
    RECORD CONTAINS 8Ø CHARACTERS
    DATA RECORD PROGLIST-REC.
Ø1  PROGLIST-REC                    PIC X(8Ø).
WORKING-STORAGE SECTION.
77  FILLER  PIC  X(3Ø)  VALUE  '*** START  WORKING STORAGE ***'.
Ø1  UMCF-FILE-STATUS                PIC X(2).
Ø1  EPGM-FILE-STATUS                PIC X(2).
Ø1  PROGLIST-FILE-STATUS            PIC X(2).
Ø1  EOF-INDICATOR                   PIC 9(1) VALUE Ø.
    88  PROG-EOF                            VALUE 1.
    88  PROGLIST-EOF                        VALUE 1.
    88  UMCF-EOF                            VALUE 1.
Ø1  OUT-HEADER1.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-PROGRAM         PIC X(8) VALUE 'PROGRAM '.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-SYSTEM          PIC X(8) VALUE 'SYSTEM'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-SUBSYSTEM       PIC X(9) VALUE 'SUBSYSTEM'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-TYPE            PIC X(4) VALUE 'TYPE'.
    Ø5  FILLER                      PIC X(6) VALUE SPACES.
    Ø5  OUT-HEADER1-COUNT           PIC X(1Ø) VALUE 'TIMES USED'.
Ø1  OUT-HEADER2.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-PROGRAM         PIC X(8) VALUE '--------'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-SYSTEM          PIC X(8) VALUE '--------'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-SUBSYSTEM       PIC X(9) VALUE '--------'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-TYPE            PIC X(8) VALUE '--------'.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-COUNT           PIC X(1Ø) VALUE '----------'.
Ø1  OUT-ITEM.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-PROGRAM            PIC X(8) VALUE SPACES.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-SYSTEM             PIC X(8) VALUE SPACES.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-SUBSYSTEM          PIC X(8) VALUE SPACES.
    Ø5  FILLER                      PIC X(3) VALUE SPACES.
    Ø5  OUT-ITEM-TYPE               PIC X(8) VALUE SPACES.
    Ø5  FILLER                      PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-COUNT              PIC X(8) VALUE SPACES.
Ø1  PROGLIST-ITEM.
    Ø5  PROGLIST-ITEM-PROGRAM       PIC X(8) VALUE SPACES.
    Ø5  FILLER                      PIC X(1) VALUE SPACES.
    Ø5  PROGLIST-ITEM-SYSTEM        PIC X(2) VALUE SPACES.
```

```
    Ø5  PROGLIST-ITEM-SUBSYSTEM     PIC X(2) VALUE SPACES.
    Ø5  FILLER                      PIC X(1) VALUE SPACES.
    Ø5  PROGLIST-ITEM-TYPE          PIC X(8) VALUE SPACES.
77  FILLER  PIC  X(3Ø)  VALUE  '**** END  WORKING STORAGE ****'.
LINKAGE SECTION.
Ø1  PARM-DATA.
    Ø5 FILLER               PIC X(2).
    Ø5 PARM-FUNCTION        PIC X(3).
PROCEDURE DIVISION USING PARM-DATA.
ØØØ-MAIN.
    PERFORM ØØ1-OPEN-FILE
    DISPLAY OUT-HEADER1
    DISPLAY OUT-HEADER2
    PERFORM UNTIL PROG-EOF
       MOVE SPACES TO EPGM-REC
       READ EPGM-FILE
          AT END
             SET PROG-EOF TO TRUE
          NOT AT END
             PERFORM ØØ2-BROWSE-UMCF
       END-READ
    END-PERFORM
    PERFORM ØØ3-CLOSE-FILE
    STOP RUN
    .
ØØ1-OPEN-FILE.
    OPEN INPUT  EPGM-FILE
                UMCF-FILE
    IF PARM-FUNCTION = 'RDO' THEN
       OPEN OUTPUT PROGLIST-FILE
    END-IF
    .
ØØ2-BROWSE-UMCF.
    MOVE LOW-VALUES TO UMCF-KEY
    MOVE EPGM-LDRPNAME TO UMCF-KEY-PROGNAME
    START UMCF-FILE KEY IS GREATER THAN UMCF-KEY
       INVALID KEY
          DISPLAY 'UMCF FILESTATUS: ', UMCF-FILE-STATUS
    END-START
    READ UMCF-FILE NEXT
    IF UMCF-KEY-PROGNAME = EPGM-LDRPNAME THEN
       MOVE UMCF-KEY-PROGNAME TO OUT-ITEM-PROGRAM
       MOVE UMCF-KEY-SYSTEM TO OUT-ITEM-SYSTEM
       MOVE UMCF-KEY-SUBSYSTEM TO OUT-ITEM-SUBSYSTEM
       MOVE UMCF-KEY-TYPE TO OUT-ITEM-TYPE
       IF PARM-FUNCTION = 'RDO' THEN
          MOVE UMCF-KEY-PROGNAME TO PROGLIST-ITEM-PROGRAM
          MOVE UMCF-KEY-SYSTEM TO PROGLIST-ITEM-SYSTEM
          MOVE UMCF-KEY-SUBSYSTEM TO PROGLIST-ITEM-SUBSYSTEM
          MOVE UMCF-KEY-TYPE TO PROGLIST-ITEM-TYPE
```

```
               WRITE PROGLIST-REC FROM PROGLIST-ITEM
            END-IF
         ELSE
            MOVE EPGM-LDRPNAME TO OUT-ITEM-PROGRAM
            MOVE '--' TO OUT-ITEM-SYSTEM
            MOVE '--' TO OUT-ITEM-SUBSYSTEM
            MOVE '--' TO OUT-ITEM-TYPE
         END-IF
         MOVE EPGM-CONT-MP TO OUT-ITEM-COUNT
         DISPLAY OUT-ITEM
         .
      ØØ3-CLOSE-FILE.
         CLOSE EPGM-FILE
               UMCF-FILE
         IF PARM-FUNCTION = 'RDO' THEN
            CLOSE PROGLIST-FILE
         END-IF
         .
      *---------------------------------------------------------------*
```

The program reads the sequential file created by the previous step and performs a search for systems and subsystems of input programs in the user MCF. An extract from the MCFREPRT report is shown below:

```
  PROGRAM   SYSTEM    SUBSYSTEM  TYPE      TIMES USED
  --------  --------  --------  --------  ----------
[...]
  XFHMSES1  XC        TP         PTC       128296
[...]
```

If PARM='RDO' is specified, it also reads the sequential file for program name, system+subsystem and type, like this:

```
[...]
XFHMSES1 XCTP PTC
[...]
```

SORTPROG sorts by system and subsystem the sequential file built by the previous step to identify those programs having the same system and subsystem. This prevents the next step from making duplicate ‘ADD GROUP’ RDO statements. The #ARDPSRT member is used as sysin for the SORTPROG JLC step of ARDP:

```
 SORT FIELDS=(1Ø,4,CH,A),FORMAT=CH
```

The CRTRDOCB program:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.                CRTRDOCB.
 AUTHOR.            GIANLUCA BONZANO.
 ****************************************************************
*                                                                *
 *----------------------------------------------------------------*
 ****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
         DECIMAL-POINT IS COMMA.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

     SELECT  PROGLIST-FILE   ASSIGN TO PROGLIST
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS PROGLIST-FILE-STATUS.
     SELECT  RDODEF-FILE     ASSIGN TO RDODEF
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS RDODEF-FILE-STATUS.
 DATA DIVISION.
 FILE SECTION.
 FD  RDODEF-FILE
     RECORDING MODE IS F
     RECORD CONTAINS 8Ø CHARACTERS
     DATA RECORD RDODEF-REC.
 Ø1  RDODEF-REC                      PIC X(8Ø).
 FD  PROGLIST-FILE
     RECORDING MODE IS F
     RECORD CONTAINS 8Ø CHARACTERS
     DATA RECORD PROGLIST-REC.
 Ø1  PROGLIST-REC.
     Ø5  PROGLIST-PROGRAM            PIC X(8).
     Ø5  FILLER                      PIC X(1).
     Ø5  PROGLIST-SYSTEM             PIC X(2).
     Ø5  PROGLIST-SUBSYSTEM          PIC X(2).
     Ø5  FILLER                      PIC X(1).
     Ø5  PROGLIST-TYPE               PIC X(8).
 WORKING-STORAGE SECTION.
 77  FILLER  PIC  X(3Ø)  VALUE  '*** START  WORKING STORAGE ***'.
 Ø1  RDODEF-FILE-STATUS              PIC X(2).
 Ø1  PROGLIST-FILE-STATUS            PIC X(2).
 Ø1  PREV-SYSTEM                     PIC X(2) VALUE SPACES.
 Ø1  PREV-SUBSYSTEM                  PIC X(2) VALUE SPACES.
 Ø1  EOF-INDICATOR                   PIC 9(1) VALUE Ø.
     88  RDODEF-EOF                          VALUE 1.
     88  PROGLIST-EOF                        VALUE 1.
 Ø1  RDODEFINE-ITEM.
     Ø5  FILLER                      PIC X(4)  VALUE 'DEF '.
     Ø5  RDODEFINE-ITEM-RESTYPE      PIC X(4)  VALUE SPACES.
```

```
    Ø5  FILLER                      PIC X(1)  VALUE '('.
    Ø5  RDODEFINE-ITEM-PROGRAM      PIC X(8)  VALUE SPACES.
    Ø5  FILLER                      PIC X(4)  VALUE ') G('.
    Ø5  RDODEFINE-ITEM-GROUP.
        Ø7  RDODEFINE-ITEM-GROUP-SYSTEM    PIC X(2) VALUE SPACES.
        Ø7  RDODEFINE-ITEM-GROUP-SUBSYSTEM PIC X(2) VALUE SPACES.
        Ø7  FILLER                         PIC X(4) VALUE 'P   '.
    Ø5  FILLER                      PIC X(1)  VALUE ')'.
    Ø5  FILLER                      PIC X(5Ø) VALUE SPACES.
Ø1  RDOADD-ITEM.
    Ø5  FILLER                      PIC X(6)  VALUE 'ADD G('.
    Ø5  RDOADD-ITEM-GROUP           PIC X(8)  VALUE SPACES.
    Ø5  FILLER                      PIC X(5)  VALUE ') LI('.
    Ø5  RDOADD-ITEM-LIST            PIC X(8)  VALUE SPACES.
    Ø5  FILLER                      PIC X(1)  VALUE ')'.
    Ø5  FILLER                      PIC X(52) VALUE SPACES.
77  FILLER  PIC  X(3Ø)  VALUE  '**** END  WORKING STORAGE ****'.
LINKAGE SECTION.
Ø1  PARM-DATA.
    Ø5 FILLER               PIC X(2).
    Ø5 PARM-LIST            PIC X(8).
PROCEDURE DIVISION USING PARM-DATA.
ØØØ-MAIN.
    PERFORM ØØ1-OPEN-FILE
    PERFORM UNTIL PROGLIST-EOF
       MOVE SPACES TO PROGLIST-REC
       READ PROGLIST-FILE
          AT END
             SET PROGLIST-EOF TO TRUE
          NOT AT END
             PERFORM ØØ2-WRITE-RDODEF
       END-READ
    END-PERFORM
    PERFORM ØØ3-CLOSE-FILE
    STOP RUN
    .
ØØ1-OPEN-FILE.
    OPEN INPUT  PROGLIST-FILE
    OPEN OUTPUT RDODEF-FILE
    .
ØØ2-WRITE-RDODEF.
    IF PROGLIST-TYPE = 'MAP' THEN
       MOVE 'MAPS' TO RDODEFINE-ITEM-RESTYPE
    ELSE
       MOVE 'PROG' TO RDODEFINE-ITEM-RESTYPE
    END-IF
    MOVE PROGLIST-PROGRAM TO RDODEFINE-ITEM-PROGRAM
    MOVE PROGLIST-SYSTEM TO RDODEFINE-ITEM-GROUP-SYSTEM
    MOVE PROGLIST-SUBSYSTEM TO RDODEFINE-ITEM-GROUP-SUBSYSTEM
    DISPLAY RDODEFINE-ITEM
```

```
           WRITE RDODEF-REC FROM RDODEFINE-ITEM
           IF PREV-SYSTEM NOT EQUAL PROGLIST-SYSTEM OR
              PREV-SUBSYSTEM NOT EQUAL PROGLIST-SUBSYSTEM THEN
              MOVE RDODEFINE-ITEM-GROUP TO RDOADD-ITEM-GROUP
              MOVE PARM-LIST TO RDOADD-ITEM-LIST
              DISPLAY RDOADD-ITEM
              WRITE RDODEF-REC FROM RDOADD-ITEM
           END-IF
           MOVE PROGLIST-SYSTEM TO PREV-SYSTEM
           MOVE PROGLIST-SUBSYSTEM TO PREV-SUBSYSTEM
           .
       003-CLOSE-FILE.
           CLOSE PROGLIST-FILE
                 RDODEF-FILE
           .
      *---------------------------------------------------------------*
```

The CRTRDOCB program reads a list of records with program, system+subsystem, and type, and then, using the RDOLIST parameters, makes 'DEFINE PROGRAM' and 'ADD GROUP' RDO statements.

In the CSDUPDT step of ARDP, the DFHCSDUP CICS utility updates the CSD file, reading the RDO statements just created.

A partitioned directory containing sysin used by the JCL is shown below:

```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
EDIT              CR10157.ENDEVOR.RDO                Row 00001 of 00016
Command ===>                                           Scroll ===> CSR
            Name    Prompt Size   Created          Changed           ID
__________ $ARDP             93  2004/01/13  2004/01/13 16:39:46  CR10157
__________ $USERMCF          87  2004/01/12  2004/01/13 17:14:43  CR10157
__________ #ARDPSRT           1  2004/01/13  2004/01/13 16:13:10  CR10157
__________ #UMCFREP           1  2004/01/13  2004/01/13 10:23:08  CR10157
__________ #UMCFSRT           5  2004/01/13  2004/01/13 10:24:10  CR10157
__________ MVSRILDF          25  2004/01/13  2004/01/13 10:21:45  CR10157
__________ MVSRILDL           3  2004/01/13  2004/01/13 12:48:47  CR10157
__________ MVSRILRP           3  2004/01/13  2004/01/13 10:20:09  CR10157
__________ P1T4DEL            2  2004/01/13  2004/01/13 11:40:49  CR10157
__________ P1T4SORT           4  2004/01/13  2004/01/13 11:50:42  CR10157
           **End**
```

Gianluca Bonzano
Systems Programmer
Cedacri (Italy)

Tuning CICS TS 2.3 EJB server utilizing enhanced EJB and Java support

CICS TS 2.3, which became GA on 19 December 2003, builds on the key capabilities and benefits delivered in V2.2, by dramatically enhancing CICS support for Java.

IBM made tremendous improvements in JVM storage management and allocation, added new TCBs for JVM processing, created new reusable types of JVM to address performance issues and to reduce CPU overhead, and provided storage protection for Java applications. All of these enhancements translate into more robust EJB and Java support and a very stable, manageable, and efficient environment to develop Java applications under CICS. Java applications now use less CPU and execute much faster, making this release a much better choice than V2.2.

CICS TS V2.3 exploits the innovative SDK 1.4, together with an architecture ensuring that Java applications have a high degree of isolation from each other and improved performance.

CICS TS 2.3 introduced a new shared class cache facility for the JVM, which improves JVM performance, storage usage, and start-up time. JVMs that use the shared class cache start up more quickly and have lower storage requirements than JVMs that don't. It should be pointed out that CICS is currently ahead of WebSphere support for shared class cache (and SDK 1.4.1), which is planned for WebSphere V5.1, which, at the time of writing this article, is not GA.

UNDERSTANDING JVM TCBS

First, let's take a closer look at which JVMs are used to support Java applications.

With CICS TS 2.2, J8 JVM TCB has been added to handle all Java requests. CICS TS 2.3 added two more JVM TCBs – JM and J9.

- JM TCB is used by the master JVM and cannot be used to run Java applications. It exists only to initialize and own the shared class cache.
- J8 TCB is used to run Java programs in a JVM that executes in CICS key.
- J9 TCB is used to run Java programs in a JVM that executes in user key.

CICS uses the Open Transaction Environment (OTE) to run JVMs. Each JVM runs on an MVS TCB, which is allocated from a pool of J8 and J9 mode open TCBs, managed by CICS in the CICS address space. This pool of open TCBs is called the JVM pool. The priority of the J8 and J9 mode open TCBs in the JVM pool is set lower than that of the main CICS QR TCB to ensure that Java programs do not affect the main CICS workload.

All new CICS transactions still start on the QR TCB and when a transaction makes use of a Java program CICS switches to either a J8 or a J9 TCB and runs the program under the control of a JVM there. When the program terminates, or if it needs to access a CICS managed resource, CICS switches control back to the QR TCB for that piece of processing. In this way, the JVM can pause without interrupting the rest of the CICS workload, so serialization of access to CICS resources and the management of the start and end of all transactions is done using the QR TCB.

CICS reduces the number of active JVMs automatically if the workload does not require them. If a JVM is inactive for 30 minutes, it is discarded. If necessary, it is possible also to terminate all the JVMs in the JVM pool by using the CEMT SET JVMPOOL PHASEOUT (or PURGE) or CEMT PERFORM CLASSCACHE PHASEOUT (or PURGE).

When an application requests the execution of a Java program, CICS first sees whether the Java program can reuse one of the existing JVMs in the JVM pool that is not currently allocated to a task. If the application can reuse an existing JVM, CICS has saved the cost of creating a new JVM. If a suitable JVM is not available, and the limit set by the MAXJVMTCBS system

initialization parameter has not yet been reached, CICS allocates a new open TCB in the correct mode (J8 or J9) and creates a new JVM.

CICS TS 2.3 also introduced new worker JVMs that are used to run Java applications and share the class cache. These JVMs help to maintain the isolation between the Java applications being processed in the system. The worker JVMs use the classes loaded in the shared class cache, instead of having to load these classes from the file system. Although the worker JVMs share the class cache, each worker JVM owns all the working data (objects and static variables) for the applications that run in it. Any worker JVM can modify the shared class cache. When worker JVMs perform just-in-time (JIT) compilation of classes that are in the shared class cache, they write the results of the compilation to the shared class cache, so that other worker JVMs can use the compiled classes. The master JVM that initializes the shared class cache is invoked in user key, so that worker JVMs that were invoked in user key can read and write to the shared class cache. Even if all the worker JVMs that share the class cache are invoked in CICS key, the master JVM and the shared class cache are still in user key.

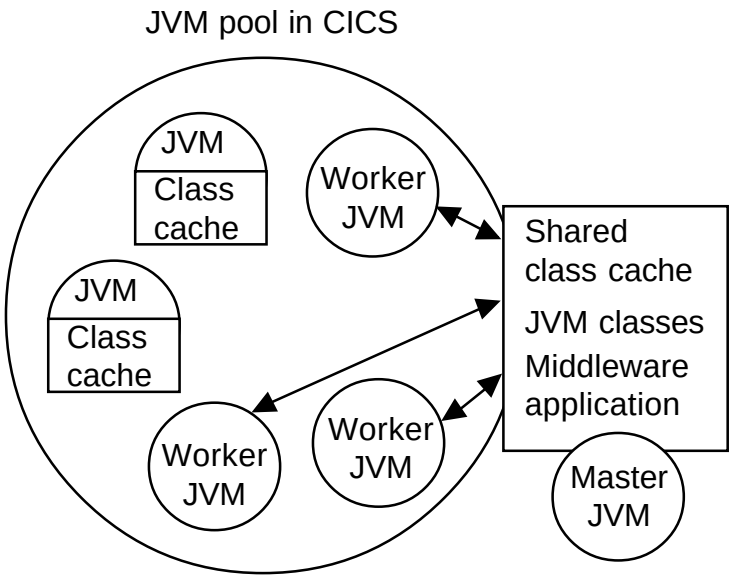


Figure 1: A sample JVM pool in a CICS address space

Figure 1 demonstrates a sample JVM pool in a CICS address space. CICS supports one active shared class cache in each region. Use of the shared class cache improves the performance of JVMs in two respects:

- Average start-up time for a new JVM is reduced, because sharable classes will, in most cases, have been preloaded in the shared class cache.
- The memory used by sharable classes is reduced, because there is a single copy of each class in the shared class cache, rather than one copy for each JVM.

Also, there is a new selection mechanism for creating, managing, and allocating work in JVMs. JVMs can have different characteristics and are grouped for management purposes. This increases the utilization of system resources and improves behaviour when there is a storage constraint and CICS is under stress.

JVMCCSIZE specifies the size of the shared class cache on an initial or cold start of CICS. The size of the shared class cache can be between 1MB and 2047MB, with the default set at 24MB. You can use the CEMT PERFORM CLASSCACHE START or RELOAD command to change the size of the shared class cache while CICS is running. On subsequent restarts, the value from the last CICS execution is used, unless you provide JVMCCSIZE as a SIT override.

If the storage in your shared class cache becomes full, worker JVMs can continue to use the classes and compiled code that are already present in it. However, if a worker JVM subsequently tries to add a new class or the results of JIT-compilation to the shared class cache, the worker JVM throws a java.lang.OutOfMemoryError. If you find that this is happening, you need to increase the size of the shared class.

JVM TYPES

To help you choose the correct configuration of CICS to support a specific Java workload, let's take a look at how you can select

JVM type (and action between JVM uses)	REUSE option in profile	Compatible with the shared class cache?	Are program invocations allowed to pass state to subsequent invocations?	Are programs allowed to change characteristics of the JVM?	Performance
Resettable (JVM reused and reset)	REUSE= RESET	Yes	No (JVM storage cleaned up after each use)	No (JVM destroyed if this occurs)	Medium (JVM reset, but not initialized for each use)
Continuous (JVM reused but not reset)	REUSE= YES	Yes	Yes	Yes	Highest (JVM not initialized or reset for each use)
Single-use (JVM destroyed)	REUSE= NO	No	No (JVM destroyed)	Yes	Lowest (JVM initialized for each use)

Figure 2: Three JVM types in CICS TS 2.3

an appropriate JVM type for your application, based on the REUSE setting.

There are three JVM types in CICS TS 2.3 – resettable, continuous, and single use – as shown in Figure 2.

Resettable and single-use JVMs have been available in V2.2 and had a relatively high overhead associated with JVM creation. To reduce JVM start-up requirements, storage usage, and CPU overhead associated with starting and stopping a JVM, CICS TS 2.3 introduced continuous JVM, which exploits the shared classes configuration of the JVM, with the use of the shared class cache.

The continuous JVM has a greater transaction throughput and lower CPU usage than the resettable JVM because it is not performing a reset. You need to keep in mind that the level of isolation of application state is reduced between transactions that reuse the same JVM.

The continuous JVM can be kept in the JVM pool for reuse, but unlike the resettable JVM, it does not prevent programs from being affected by the actions of a previous program invocation in the same JVM. This type of JVM is initialized once, and is reused many times, but it is not reset after each Java program has completed.

The choice of mode between resettable or continuous is application dependent. Although continuous JVMs provide the maximum performance benefits, the initial testing and deployment of applications is recommended in resettable JVM mode to provide the maximum assurance that the application is not compromising the integrity of the JVM by performing unresettable actions. If application code has broken the rules, JVM will become unresettable and CICS will destroy the JVM when the Java program has finished using it. The storage used by the JVM is then recovered, and a new JVM will be initialized, encountering an overhead associated with JVM creation. You need to review CICS statistics for JVM profiles – the field 'Number of times JVMs were unresettable for this profile' – to ensure that this condition is not occurring, causing JVM to re-initialize for each transaction, at a higher CPU cost.

If you are convinced that the application does not compromise the integrity of the JVM or allow unwanted application state to persist, and you require maximum performance, then moving to the use of continuous JVMs is recommended.

With continuous JVM, static or dynamic state persists in the JVM's storage heaps, and threads that are not quiesced will persist, along with their related storage. Sharable application classes are not re-initialized, and non-sharable application classes are not discarded and re-loaded, but are kept intact. Also, a continuous JVM does not invoke the ibmJvmTidyUp method to request the middleware classes to perform clean-up.

EJB SUPPORT – CICS AND WEBSPHERE

If you are just starting to look into how CICS TS 2.3 can help you Web-enable your existing applications, you may wonder why you would want to take advantage of enhanced CICS EJB support.

First let's take a quick look at what a JavaBean is. It is a self-contained, reusable software component, written in Java, that has a visual element and executes within some type of visual container when used in a desktop application. Server components

are application components that run in an application server such as CICS and, unlike desktop components, they do not have a visual element and they are known as Enterprise Beans (EJBs).

The benefit of using EJBs is that developers can create complex applications by focusing on business logic rather than environmental and transactional issues. EJB architecture is independent of any specific platform, proprietary protocol, or middleware infrastructure. Applications developed for one platform can be re-deployed on other platforms. The EJB architecture can improve the productivity of application developers by standardizing and automating the use of complex infrastructure services such as transaction management and security checking.

It is important to point out that the support for EJBs in WebSphere is not the same as in CICS, but they can complement each other. CICS is optimized to support short-running pseudo-conversational transactions and supports only session beans. There are two types of session bean:

- *A stateful session bean that manages its own transactions.* It can begin an OTS transaction in one method and commit or roll it back in a subsequent method.
- *A stateless session bean that manages its own transactions.* It begins an OTS transaction and must commit (or roll back) that transaction in the same method in which it was started. This a more common way of using session beans in CICS.

Enterprise beans in a CICS EJB server benefit from CICS transaction management services like system log management, performance optimizations, runaway and deadlock detection, TCLASS management and monitoring, and statistics.

WebSphere Application Server provides efficient support for entity beans because it is optimized to manage long-lived transactions that manage data. CICS can exploit that support with no impact on the application programming model. WebSphere Application Server for z/OS provides the most efficient implementation of entity beans that encapsulate DB2 data. For

example an application that needs access to DB2 can get the best of both worlds by running session beans in CICS and entity beans in WebSphere for z/OS. With both containers running on the same z/OS image there are opportunities for optimizing the data transport, providing very efficient secure inter-operation and avoiding data transformations.

You can use JCICS or the CCI Connector for CICS TS to build enterprise beans that make use of existing (non-Java) CICS programs. For an application developer with limited knowledge of CICS, programming using CCI Connector for CICS TS to access existing programs from an EJB may be a preferred choice over a JCICS method call. Using the CICS Connector enables the use of the VisualAge for Java Enterprise Access Builder tool to create simple command beans and navigator beans. The command and navigator beans are simple to use and their use requires no CICS programming skill. These command and navigator beans are portable between CICS and WebSphere on all the platforms that support the CICS Transaction Gateway.

One example of a design where you would use CICS EJB server with WebSphere would be writing a CICS Java program that can start with a main method that invokes the EJB residing inside WebSphere. You can then XC LINK to this with a CICS COMMAREA to pass the parameters with which to invoke the EJB. The invocation of this EJB would usually be coded to look up the address of the WebSphere server using an external JNDI/ LDAP database. To get better performance and improve portability you can populate the relevant objects directly within the Java Wrapper program, providing that you know where the WebSphere is located**.**

You do need to be aware that WebSphere V5 supports EJB 2.0 whereas CICS supports EJB 1.1. This means that the EJB you use within WebSphere and invoke from CICS has to be written to the lower EJB 1.1 specification so that it can be invoked from both places. One of the restrictions would be not to use extended home methods or a local interface for the EJB methods. CICS TS 2.3 does provide toleration for EJB 2.0 JAR files. If you create a new EJB JAR file with the Application Assembly Tool (AAT)

shipped with WebSphere Application Server V5, AAT saves it in EJB 2.0 format. Although CICS supports only Version 1.1 of the EJB specification, it can tolerate EJB 2.0 JAR files and ignores 2.0-specific features in the deployment descriptor.

PROGRAMMING CONSIDERATIONS

When you are developing Java applications to run in a continuous JVM, it is important to remember that the continuous JVM does not ensure that invocations of Java programs are isolated from changes made by previous invocations of programs in the same JVM. User application classes, as well as middleware classes that run in a continuous JVM, are able to change the state of the JVM in ways that might affect subsequent program invocations. To help eliminate these unresettable actions where they are not desired, during the development process, you can run the program in a resettable JVM (with the option REUSE=RESET). Resettable JVMs record unresettable actions, and you can use this information to help you re-design the application if necessary. When you are certain that the program does not change the state of the JVM in undesirable ways, you can move to using it in a continuous JVM.

Also remember that invocations of Java programs in a continuous JVM are able to pass on state to subsequent invocations of programs in the same JVM. You might be able to use this to your advantage in designing your Java applications if you want information to persist from one program invocation to the next. Because static state and object instances referenced through static state are not reset between JVM reuses in a continuous JVM, it is permissible for applications to create persistent items that might be of use to future executions of the same application in the same JVM.

The first one is the ability for JVMs to share a cache of commonly-used class files that are already loaded. This enables faster JVM start-up and reduced cost of class loading. When a new JVM that shares the class cache is initialized, it can use these pre-loaded classes instead of reading them from the file system. Also, if the

JVM performs JIT compilation for any of the classes, it can write the results back to the shared class cache, and other JVMs can then use the compiled classes. All the heap data (objects and static variables) is owned by the individual JVMs; this maintains the isolation between the applications being processed in the JVMs.

The simplified storage management of the continuous JVM offers large performance benefits over the resettable JVM. However, there are risks associated with running applications that do not conform to the reuse requirements of this mode. Java programs running in this mode should not carry out actions that alter the JVM's state unless they restore the original values before ending. The most common cause of this is from the inclusion of static storage objects in Java programs. These are not automatically reinitialized on subsequent program invocations and should be used with care, when running in this mode. One way to prevent this is by defining all class fields as final.

CONSIDERATIONS FOR DEVELOPERS WHEN USING PERSISTENT REUSABLE JVM (REUSE=RESET)

Trusted middleware classes in the persistent reusable JVM are privileged classes that are able to operate without the restrictions imposed on non-trusted application classes such as EJB beans. Trusted classes can load and manage the resettability state of native routines, which cannot be checked at run-time for their resettability characteristics. Also, because they can have a persistent state across a JVM reset, they can act as a long-term cache for middleware objects, which might provide optimizations to applications.

Trusted middleware classes remain persistent across a JVM reset, and so do not need to be reloaded for each transaction. The developers need to ensure that any classes that were used by the application code for a transaction are reinitialized before reuse by the next transaction following a JVM reset, and need to tidy up resources acquired while running the application code for a transaction when the application code completes.

The performance gains possible with the persistent reusable JVM technology are available only if the developers of middleware and application code are prepared to accept some constraints on their programming practice. These constraints have been minimized, but it is important to understand the choices that have been made during the persistent reusable JVM development. An understanding of these choices allows middleware developers to choose design options that best exploit the underlying JVM.

As previously stressed, it is important that application classes do not perform actions that make the JVM unresettable, causing recycling of the JVM, and paying for additional CPU overhead associated with this process. When a persistent reusable JVM is used, the assumption is that middleware performs whatever control and clean-up is necessary to restore the JVM state to an acceptable value prior to the next transaction. You need to watch out for actions that make a JVM unresettable. Please refer to *New IBM Technology featuring Persistent Reusable Java Virtual Machines*, publication SC34-6034-01, for guidelines on determining why a JVM becomes unresettable and a complete list of unresettable actions and conditions.

To help you debug the reason a JVM becomes unresettable, you will need to set the JVM system property ibm.jvm.unresettable.events.level to enable the logging of unresettable events in a resettable JVM, and set the level of logging required. Specifying *min* produces a list of reason codes that define the unresettable events found; specifying *max* produces the reason codes and also a stack trace where appropriate. To see these events, you will also need to set the ibm.jvm.events.output system property to enable event logging.

ALLOCATION OF FREE JVMS – TUNING OPPORTUNITIES

With CICS TS 2.3, IBM implemented enhancements in allocating free JVMs, which result in greater overall throughput of Java programs. This is the area where CICS systems programmers can review CICS statistics and make tuning changes to improve Java applications' performance.

Let's look first how 'mismatch' and 'stealing' occur while processing requests for a JVM.

When an application requests a JVM, CICS first tries to find a suitable JVM that is available for reuse in the JVM pool. If a suitable JVM, with the correct JVM profile and execution key, is not available, and the MAXJVMTCBS limit for the JVM pool has not yet been reached, CICS can create a new JVM for the application. If there are no suitable JVMs and no space in the JVM pool, CICS can fulfil an application's request for a JVM by destroying and re-initializing an available JVM that had the wrong execution key or profile for the request. This is called a mismatch if the JVM is destroyed and re-initialized but the TCB is kept and reused. It's called a steal if both the JVM and the TCB are destroyed and replaced.

There is a selection mechanism that CICS goes through to avoid excessive mismatches and steals, while ensuring that no application waits too long to receive a JVM. Although CICS attempts to maintain an appropriate balance between the different types of JVM in the JVM pool (without operator intervention), in some cases reviewing the JVM statistics may show inefficiencies. This new selection mechanism also makes more efficient use of free JVMs when the number of JVMs is at its maximum limit and when MVS storage is severely constrained.

You may want to limit the numbers of a certain type of JVM, if you are not satisfied with the measures CICS is taking to avoid mismatches and steals for that type of JVM. You might find that the requests are waiting longer than the critical period defined by CICS, and so CICS is giving the requests free JVMs with the wrong profile or execution key, and causing a mismatch or a steal.

You can use the CICS statistics to see whether the incidence of mismatches and steals in the JVM pool is greater than you would like. These statistics can be accessed online using the EXEC CICS COLLECT STATISTICS JVMPROFILE command, which shows the number of TCB mismatches and TCB steals for the TCB modes J8 and J9, and shows the overall incidence of mismatches and steals in the JVM pool.

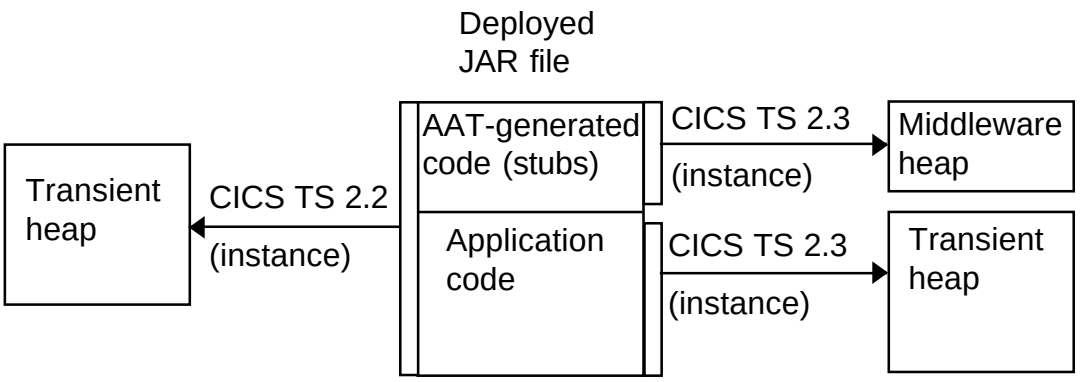


Figure 3: Heap storage allocation differences

There is no parameter to control the number of JVMs with each JVM profile that CICS keeps in the JVM pool. However, you can indirectly limit the number of JVMs with a particular JVM profile by limiting the number of transactions that request a JVM with that profile. This can be accomplished by assigning the same TRANCLASS for the transactions which execute JVM programs requiring that JVM profile (and setting an appropriate limit for TRANCLASS definition). This technique will limit the number of concurrent executions of JVM programs requiring that JVM profile, and it will limit the maximum number of JVMs with that JVM profile that will be in the JVM pool at any one time.

Another way to control the number of mismatches and steals is to reduce the number of different JVM profiles that are used by your CICS region. You may want to ensure that all your JVM profiles and their associated JVM properties files do actually specify different options and you could investigate whether you could combine compatible options in different JVM profiles to create a single JVM profile.

ENTERPRISE BEANS' PERFORMANCE

CICS TS 2.3 has improved enterprise beans' performance as a result of objects being cached for reuse when a JVM is reset and because of changes in heap storage allocation. In CICS TS 2.2 an EJB deployed JAR file was considered to contain only

application code and was not recommended to be loaded into the middleware heap. In CICS TS 2.3, the CICS-generated code within the deployed JAR file (the implementation of the beans' home and component interfaces) is treated as middleware. This means that some objects that were previously held in the application heap and discarded when the JVM was reset have become long-lived middleware heap objects.

Figure 3 demonstrates the difference in heap storage allocation between CICS TS V2.2 and V2.3.

Some objects that were previously held in the transient heap and discarded when the JVM was reset have become long-lived middleware heap objects. Now, when a JVM is re-used to process requests against the same enterprise beans, the cached objects are reused, improving performance.

Under CICS TS 2.3, multiple JVMs can share a single cache of class files that have already been loaded, including some that have been optimized by compilation. The shared class cache replaces the system heap and the application-class system heap for those JVMs, and it can contain middleware and application classes. JVMs that use the shared class cache start up more quickly and have lower storage requirements than JVMs that don't.

Trusted middleware class path is built automatically in CICS, using the information you specify in the CICS_DIRECTORY option in the JVM profile. Any paths you specify on the optional parameters TMPREFIX and TMSUFFIX in the JVM profile are added either at the beginning or at the end of the paths constructed by CICS.

TUNING TIPS FOR CICS TS 2.2/2.3

Running Java code under CICS presents new challenges related to its relatively high CPU overhead and memory usage. In CICS TS 2.2, IBM implemented a way to avoid a full JVM initialization every time by reusing the JVM on subsequent invocations. Multiple transactions can be executed serially in a single long-

running JVM by using persistent reusable JVMs. The serial reuse of a JVM for multiple transactions, while resetting the JVM to a known state between each transaction, provides isolation without paying the high cost of a full JVM initialization for every transaction. There is only one transaction using a JVM at any one time. At transaction termination time, the JVM is made available for reuse by another transaction.

Doing extensive testing with CICS TS 2.2 in our environment, we discovered that you need to watch out for the number of JVM TCBs created, controlled by the MAXJVMTCBS value, to avoid out-of-memory allocation in MVS private storage above the line. We also learned that the overhead and memory allocation for creating a new JVM is much higher than the overhead of Java requests queueing up for an available JVM. You need to calculate the maximum number of JVM TCBs that can be created, based on MVS storage available above the 16MB line and xmx setting. Most CICS system programmers consider queueing for any resource as a bottleneck, but in CICS TS 2.2 JVM TCB case, controlling the number of TCBs is the only way to run a high volume of Java applications in a single address space. Please keep in mind that over-allocating MAXJVMTCB will cause a region outage in CICS TS 2.2 because there is no mechanism to release JVM storage when CICS is under stress. This issue has been addressed in CICS TS 2.3 with a new storage monitor provided for managing MVS storage.

Watch out for the following Java memory errors, recorded in dfhjvmerr.applid.date.txt:

```
Unable to allocate an initial java heap of 35651584 bytes.
**Out of memory, aborting**
*** panic: JVMSTØ16: Cannot allocate memory for initial java heap
*** panic: JVMSTØ17: Cannot allocate memory in
initializeMarkAndAllocBits
*** panic: JVMSTØ18: Cannot allocate memory for
initializeMarkAndAllocBits
```

A JVM in CICS runs as a Unix System Services process in a CEEPIPI enclave, using MVS LE services rather than CICS LE services. As a result, all storage obtained by the JVM is MVS storage, obtained by calls to MVS LE services. This storage

resides within the CICS address space, but is not included in DSAs. Check CICS shutdown statistics to ensure that ‘Private Area storage available above 16MB’ is not over-allocated. See the example below:

```
Private Area storage available below 16Mb . . . . . :       5,362K
Max User storage allocated above 16Mb (EXT) . . . . :     829,680K
_____________________________________________________________________
Private Area storage available above 16Mb . . . . . :     632,420K
```

Another tuning tip is to set a large enough heap size to avoid heap size expansion, have less garbage collection, and use less CPU. My recommendation is to set xmx to 60MB and xms to 30MB. Please review your heap allocation statistics to tune your system for optimum performance.

To get LE statistics, update DFHSJJ80 as follows:

- DC C'RPTO(ON) ' to report LE options.
- DC C'RPTS(ON) ' to report LE storage.

This will generate an LE report when a JVM ends, as a part of CICS output. To force this report to be generated, issue a CEMT S JVM PHASEOUT command.

Please keep in mind that running with these options set will increase CPU usage. They should be used in a test environment only.

Below is an example of an LE report, showing HEAP allocation data:

```
HEAP statistics:
  Initial size:                                    4194304
  Increment size:                                  1048576
  Total heap storage used (sugg. initial size):   20888944
  Successful Get Heap requests:                      24974
  Successful Free Heap requests:                     18652
  Number of segments allocated:                         53
  Number of segments freed:                             44
HEAP24 statistics:
  Initial size:                                          0
  Increment size:                                     4080
  Total heap storage used (sugg. initial size):          0
  Successful Get Heap requests:                          0
  Successful Free Heap requests:                         0
```

```
   Number of segments allocated:                          Ø
   Number of segments freed:                              Ø
 ANYHEAP statistics:
   Initial size:                                       4Ø96
   Increment size:                                     8176
   Total heap storage used (sugg. initial size):     633160
   Successful Get Heap requests:                        453
   Successful Free Heap requests:                       169
   Number of segments allocated:                         11
   Number of segments freed:                              3
 BELOWHEAP statistics:
   Initial size:                                       4Ø96
   Increment size:                                     2Ø48
   Total heap storage used (sugg. initial size):       2528
   Successful Get Heap requests:                          4
   Successful Free Heap requests:                         3
   Number of segments allocated:                          1
   Number of segments freed:                              Ø
 Additional Heap statistics:
   Successful Create Heap requests:                       4
   Successful Discard Heap requests:                      Ø
   Total heap storage used:                        34779136
   Successful Get Heap requests:                          4
   Successful Free Heap requests:                         Ø
   Number of segments allocated:                          4
   Number of segments freed:                              Ø
 Largest number of threads concurrently active:           5
```

Based on the statistics above, you can change the heap size allocation from the default values to the values above:

```
ANYHEAP(4Ø96,8176,ANYWHERE,FREE)   ===>
ANYHEAP(633160,8176,ANYWHERE,FREE)
BELOWHEAP(4Ø96,2Ø48,FREE)         ===>
BELOWHEAP(2528,2Ø48,FREE)
HEAP(4194Ø4,1Ø48576,ANYWHERE,FREE,Ø,4Ø8Ø)  ===>
HEAP(2Ø888944,1Ø48576,ANYWHERE,FREE,Ø,4Ø8Ø)
```

You can override LE run-time options and set your own values using the DFHSJJ8O user-replaceable module, available in V2.2 and higher. In CICS TS 2.3 there are now two new user-replaceable programs – DFHAPH8O, which allows you to alter the default Language Environment run-time options for the Java hot-pooling environment, and DFHJHPAT, an optional module that can be used for tracing. Another user-replaceable module that you need to be aware of is DFHJVMRO. It specifies the run-time options that are used to create the Language Environment

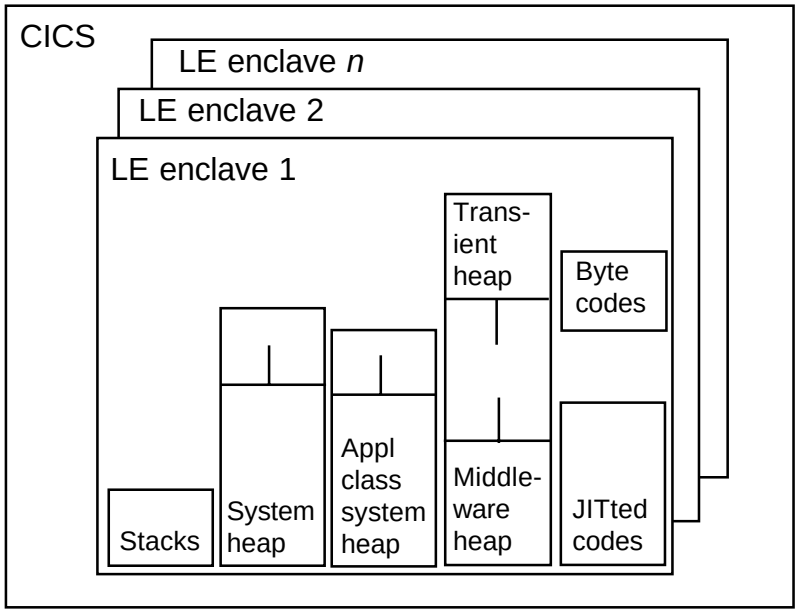
enclave in which a JVM runs. It defines storage allocation parameters for heap and stack, and a number of other options.

You can find the source for a user-replaceable module in CICS.SDFHSAMP.

To find out whether the performance of your CICS region could benefit from an increase or a reduction in the number of JVMs, you can use CICS monitoring. Here is an example of CICS shutdown statistics that should be used to properly set MAXJVMTCB limit and to analyse storage usage above the line:

```
TCB Pool . . . . . . . . . . . . . . . . . . . :              JVM
 Current TCBs attached in this TCB Pool . . :                 7
 Peak TCBs attached in this TCB Pool. . . . :                 8
 Max TCB Pool Limit (MAXJVMTCBS). . . . . . :                 7
 Requests Delayed by Max TCB Pool Limit . . :                81
 Total Max TCB Pool Limit delay time. . . . :  00:00:03.07745
 Average Max TCB Pool Limit delay time. . . :  00:00:00.03798

 Current TCBs in use in this TCB Pool . . . . . :              0
 Peak TCBs in use in this TCB Pool. . . . . . . :              8
 Times at Max TCB Pool Limit (MAXJVMTCBS) . . . :            204
 Current Requests Delayed by Max TCB Pool Limit :              0
 Peak Requests Delayed by Max TCB Pool Limit. . :              4
 Total Delay time for current delayed . . . . . :  00:00:00.00000
 Average Delay time for current delayed . . . . :  00:00:00.00000
```



Each JVM executes in a J8 TCB.

Each JVM has an LE enclave.

JVM storage allocated within the heap of the LE enclave:

- heaps
- stack
- sysmalloc requests

Figure 4: LE enclave allocation

CICS STATISTICS – LE ENCLAVE HEAP STORAGE USAGE

CICS LE statistics can be useful in sizing heap storage (xmx setting). They show how much of the LE enclave heap storage has been used by each JVM. You should review the 'Peak Language Environment (LE) heap storage used' in the JVM profile statistics to see the high-water mark amount of LE enclave heap allocated.

An alternative method uses the RPTO(ON) and RPTS(ON) options in DFHJVMRO to obtain storage reports for identifying a suitable value for the initial allocation for the amount of LE enclave heap storage. This uses the LEHEAPSTATS parameter.

You can use the following commands and settings to collect LE statistics:

- Specify the option LEHEAPSTATS=YES in each of the JVM profiles that you have identified in order to get CICS statistics that show how much LE enclave heap storage is used by your JVMs.
- Issue JVM profile statistics using the EXEC CICS COLLECT STATISTICS JVMPROFILE or CEMT PERFORM STATISTICS JVMPROFILE command in order to view the JVM profile statistics that have been collected during the statistics interval.

Although CICS LE statistics can be very useful, there is an overhead with data collection. Collecting this statistic affects the performance of JVMs and you should not continue this process of collecting LE statistics in a production environment.

You can reset collecting LE statistics by removing the option LEHEAPSTATS=YES from your JVM profiles, or by changing it to NO (which is the default). To pick up this change while CICS is up, you will need to purge your JVMs using the CEMT SET JVMPOOL PHASEOUT command to ensure that they are re-created with the option LEHEAPSTATS=NO.

TUNING HEAP STORAGE

Tuning JVM storage options gives you an opportunity to reduce the storage required for each JVM and to eliminate unnecessary CPU time spent expanding heaps and running garbage collection. As with most tuning solutions, there is usually a trade-off. For example, a large setting for the non-system heap reduces the frequency of garbage collection, but also reduces the number of JVMs that CICS can have in its address space.

Heap storage requirements vary depending on the size of the JAR files and the complexity of the Java applications that run in the JVM. With CICS TS 2.3 enhancements, type of JVM used and the level of reusability for JVM are the key contributing factors for heap storage requirements.

Figure 4 shows how an LE enclave is allocated (using J8 TCB in this example).

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

Elena Nanos
IBM Certified Solution Expert in CICS Web Enablement
Zurich NA (USA)

CICS questions and answers

Q We have a CICS region that has several different TCPIPSERVICES going into it, some of which use SSL. We're experiencing problems where we run out of available SSL TCBs, but are having difficulty in seeing which TCPIPSERVICES are in use and who has taken all the TCBs!

A CEMT I TCPIPS will tell you the number of current connections per TCPIPSERVICE, and DFHSO0122 does tell you which TCPIPSERVICE is having problems. What you could do is to define alternatives to CWXN, the Web attach transaction,

and then define different transactions to each TCPIPSERVICE.

This way you can see how easily many Web attach transactions are in the system for each TCPIPSERVICE. An additional benefit can be obtained if you TClass each of these Web attach transactions. If you set the Maxactive parameter to less than or equal to the number of SSLTCBs, you can control how many concurrent connections you can have through each TCPIPSERVICE. This way you can allocate SSL TCBs to each TCPIPSERVICE if desired, and if more come in than you can handle, the excess will wait in CICS on a TClass rather than being rejected.

If you have any CICS-related questions, please send them in and we will do our best to find answers. Alternatively, e-mail them directly to cicsq@xephon.net.

Micro Focus has announced Mainframe Express Enterprise Edition, a Windows-based environment for mainframe application development.

The product combines a mainframe emulation and development environment with application analysis and automated program and component generation. Mainframe Express Enterprise Edition also drives improvements in application performance and delivery, reducing operating costs and risks, the company claim. In addition, existing CICS legacy services can be extended to Web Services, .NET, or J2EE.

For further information contact:
Micro Focus, 9420 Key West Avenue, Rockville, MD 20850, USA.
Tel: 301 838 5000.
URL: http://www.microfocus.com/press/releases/20040608.asp.

* * *

Bristol Technology has announced TransactionVision CICS Sensor, which enables customers to automatically and non-intrusively discover transactional flows by correlating asynchronous parallel events from all the applications in the business process. This provides improved problem determination, increased service level visibility based on real-time transactional performance, and improved capacity planning. TransactionVision can now also track transactions across Web Services, J2EE, JMS, and WebSphere MQ.

For further information contact:
Bristol Technology, 39 Old Ridgebury Road, Danbury, CT 06810-5113, USA.
Tel: (203) 798 1007.
URL: http://www.bristol.com/news/pr/2004/tv_20040614_CICS.htm.

* * *

iWay Software has announced Data Migrator and Data Migrator RT, its data access and transformation tools that provide both batch (Data Migrator) and real-time (Data Migrator RT) analysis, transformation, and forwarding of message transactions from and to disparate systems. These include transaction systems such as CICS, relational databases, packaged applications such as SAP, and over 80 legacy databases including IMS data.

Data Migrator helps with migrating data from one process to another, and allows administrators to design processes in which messages are collected from one system in real-time (when using the RT tool), managed via the XML engine, and then sent to one or multiple destination systems in the necessary format.

For further information contact:
iWay Software, Two Penn Plaza, New York, NY 10121-2898, USA.
Tel: (212) 330 1700.
URL: http://www.iwaysoftware.com/cgi-shell/press/intpr/f_intpr.pl?intpr_code=05_25_04_data.

* * *

IBM has announced Version 1.2 of CICS Business Event Publisher for MQSeries, which enables extension and re-use of existing CICS applications, through events, to drive new business processes and utilize new technology. It generates user-defined WebSphere MQ messages as a side effect when certain EXEC CICS commands are executed by an application, or when VSAM, DB2, or IMS data is modified. Users can select criteria to determine which events are published and to determine content of resulting messages.

For more information contact your local IBM representative.