204

CICS update

November 2002

In this issue

CICS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office
Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues
A year's subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the December 1999 issue, are available separately to subscribers for £16.00 ($24.00) each including postage.

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

Editor
Trevor Eddolls

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

CICS dynamic workload management – concepts

This article is the first in a series of articles that look at dynamic workload management within CICS. In this first article we will look at the various factors that need to be taken into account when building such a solution. Subsequent articles will cover the CICSPlex SM WLM model, affinity management, abend compensation, balancing algorithms, balancing specific workload types, exit customization, and usage scenarios.

First we will explore the evolution of the CICS programming model and how it facilitated the exploitation of dynamic workload management techniques, but also introduced some challenges.

THE CICS PROGRAMMING MODEL

The CICS transaction model that has evolved over time is the so-called pseudo-conversational model. This is somewhat different from the conventional program model used in other application environments. Conventional programs (so called conversational programs) remain in storage from initial invocation until eventual termination, holding on to resources for much longer than strictly necessary. These resources could be held over many minutes, during which time data entry and external operator decisions are being made, limiting the number of active tasks. What was needed was a new programming approach that gave the illusion of this conventional conversational program – this is the pseudo-conversational model.

In the pseudo-conversational model, between individual interactions with the end user, the program’s state data was kept distinct from the program, which could therefore be removed from main storage until the next interaction was made by the end user (typically by depressing the *Enter* key). The program could then be brought back into storage and executed, passing the state data to that program for execution. A single execution of a transaction could therefore result in multiple executions of the program, each program execution being a segment of that pseudo-conversation. Ultimately, the program finished its processing and terminated the pseudo-conversation.

The mechanism by which CICS provided this function was via:

```
EXEC CICS RETURN NEXTTRANID(tranid) COMMAREA(state_data)
                                                  for the next segment.
EXEC CICS RETURN
     (end of pseudo-conversation).
```

Other methods of passing data between these transaction segments evolved over time via both EXEC CICS API extensions and the various other facilities provided via the EXEC CICS API (eg Transient Data, Temporary Storage, GETMAIN SHARED storage).

This, of course, was fine when executing in a single address space.

With the advent of the CICS TOR/AOR model, this pseudo-conversational technique allowed CICS transactions initiated in the TOR (where the state data was usually maintained) to execute application code in the AOR, with each segment of the transaction thread potentially executing in a different AOR. Several of the earlier programming techniques employed, however, inhibited this potential, or at least reduced it, by saving state data in the AOR. This state data therefore led to affinities with the AOR that, dependent on the techniques employed, were related to various external attributes and existed for various lifetimes. These affinities are outlined below. Consequently, in the real world, dynamic transaction routing has to tolerate any inter-transaction and transaction-system affinities that may exist within the workload (unless the application provider removes such affinities).

As well as affinities, the health of the target AORs and their links, the probability of the transaction abending within a given region, workload separation criteria, and workload balancing also need to be taken into account to provide a robust routing environment. The reality of TORs and AORs living in separate MVS images, or even sysplexes, and the need to have load data and affinities propagated without significant effects on performance, had to be taken into consideration. The release levels of both routing and target regions and dynamic reconfiguration all need to be taken into account.

We will now define some terminology for use in the subsequent articles.

WORKLOAD MANAGEMENT RULES

The role a region plays in workload management is important. Three roles can be played, namely:

- Requestor – the region that makes the workload request.
- Router – the region that makes the routing decision.
- Target – the region where the requested work is delegated.

A region can play multiple roles within a given work request.

Workload balancing

Workload balancing is the act of selecting a given CICS region that is to process a transaction instance from a set of candidate regions. It is achieved by applying a balancing algorithm, which selects a region according to some criteria. Many such algorithms can be invented. A simple 'round robin' algorithm where each instance is allocated to the next region in the list is an example.

CICSPlex SM provides two algorithms – queue and goal.

Workload separation

The user may want to partition the incoming work to various subsets of targets. This could be because of different versions of an application being run, or separation based on geographical location or departmental/ bureau like activities. Different routing criteria could also be required from different subsets of routers.

CICSPlex SM allows the user to partition workloads to different sets of targets based on userid, luname, processtype, and groups of transactions. The corresponding target scope identifies the potential candidate targets across which balancing is to occur for that quadruplet.

Workload affinity

We have briefly discussed above the techniques that created affinities. While all of these methods worked for statically routed transactions, as soon as these transactions are allowed to run in a dynamic environment, access to their state data by the above methods can no longer be guaranteed.

As a simple example consider an application routed to AOR1 (see Figure 1). Whilst running in AOR1 it places some state data into the CWA. After soliciting some input from the end user, dynamic routing chooses AOR3 as the best place to route. Upon arrival in AOR3, the application addresses the CWA and performs some logic on the data before writing back to a database. Now one of two things can happen. The transaction can abend, giving the programmer a chance to realize

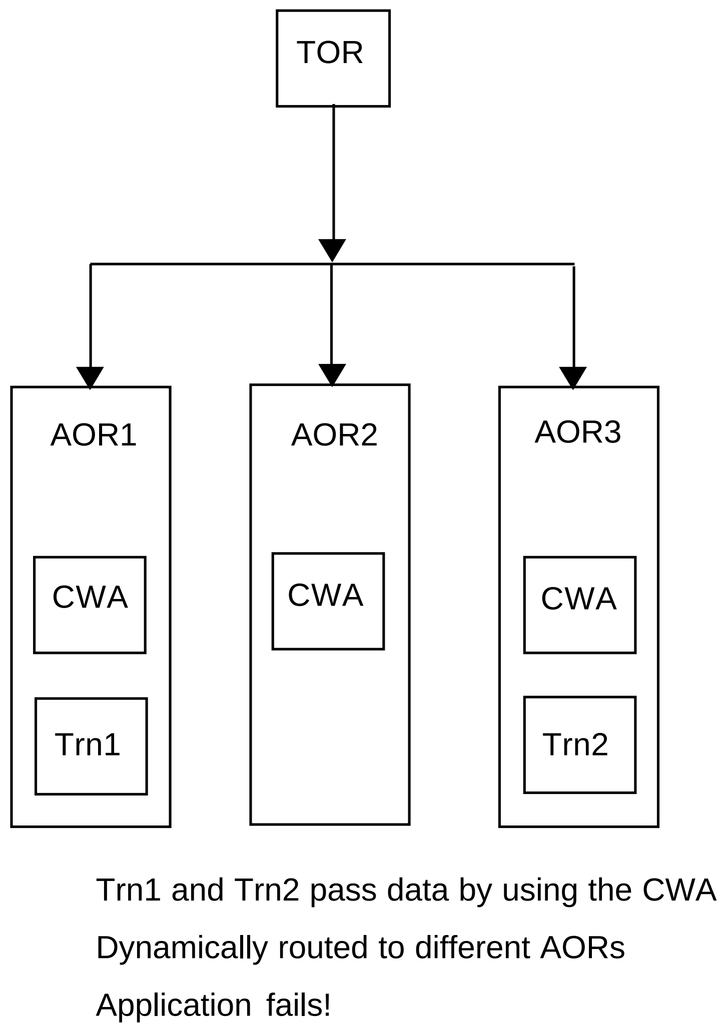
TOR
AOR1
AOR2
AOR3
CWA
CWA
CWA
Trn1
Trn2
Trn1 and Trn2 pass data by using the CWA
Dynamically routed to different AORs
Application fails!

Figure 1: Affinity example

that something is wrong, or it can write erroneous data back to the database (corrupting its contents in the process), and return normally to the end user without anyone being the wiser. As we can see from the above example, it is crucially important to know about such affinities and manage them. Another example of this would be using CICS temporary storage to contain data rather than passing the data back via a COMMAREA for subsequent use by the transaction.

CICSPlex SM manages a wide variety of such affinities and their associated lifetimes.

The affinity can be associated with various attributes of the environment, for example a userid or a terminal.

Affinities also last for a given amount of time. This affinity lifetime is dependent on the affinity type and is depicted in Figure 2:

- Pseudo-conversational where the data exists for only the length of the CICS pseudo-conversation (normally a short period of time).
- Signon (signon and signoff of a given user to the CICS TOR).
- Logon (VTAM logon to logoff).
- System (whilst the given CICS AOR is active).
- Permanent where the data exists for all time once an AOR has been chosen (thankfully few exist).
- Activity is to the end of the activity.
- Process is to the end of the process.

Various programming techniques, which utilized flavours of pseudo-conversational behaviour via menu front ends, gave rise to:

- Delim where a given transaction ‘delimits’ the affinity.
- Specific identification of transactions that ‘Start’ and ‘End’ an affinity.

The various affinities and lifetimes that can exist are shown in Figure 2.

Lifetimes

	Permanent	System	Signon	PConv	Delim	Logon
Userid	x	x	x	x	x	
LUName	x	x		x	x	x
Global	x	x				

For CICS Business Application Services, the following affinities and lifetimes are defined:

BAPPL	Permanent	System	Activity	Process

Figure 2: Affinities and lifetimes

AFFINITY DETECTION

CICS Transaction Server provides an affinity utility to detect affinities in applications. A simple load module scanner scans the contents of a load module for EXEC CICS commands, which could potentially create an affinity. A report is generated for the application programmer to review. An online set of exits can also be used to gather affinity information in a running environment. This data can subsequently be printed for offline analysis. Input from these to sources can be used to understand the affinities that exist in a given application. Note that these tools identify potential affinities. An affinity may not actually exist in the code. The reports should be reviewed by someone who understands the code, to determine whether any real affinities exist. After this review has been performed you can either eliminate the affinity or have CICSplex SM manage the affinity for you. The affinities utility also provides the ability to create CICSPlex SM batchrep input for detected affinities.

Of course, you need to run all potential paths to discover all potential affinities. This is easier than it sounds. Many applications have been built on others over many years and the set of programs touched by a given application is no longer apparent. The CICS Interdependency Utility provides the ability to identify such usage. In collaboration with the affinity utility it helps identify ‘when have I finished?’.

DYNAMIC CREATION/DELETION OF AFFINITIES

The affinities described so far are statically defined, ie only a relationship between the transaction names and their affinity is defined, with CICSPlex SM creating and deleting affinity elements for those affinities and lifetimes defined above. CICSPlex SM also supports the ability to dynamically create and destroy affinities based on information accessible to the dynamic routing program. This requires customization of the CICSPlex SM routing program to utilize the create and destroy affinity CICSPlex SM verbs. One possible use of this would be if the COMMAREA contained sufficient data for the routing program to deduce whether an affinity needed to be created or destroyed. The various techniques that could be employed are beyond the scope of this article.

HEALTH DATA AND RTA THRESHOLDING

When routing the transaction to a given set of CICS AORs, it obviously makes sense to send that work to the regions most capable of executing that work. CICSPlex SM’s Real Time Analysis (RTA) component can be used to help identify such regions. The System Availability Management (SAM) identifies various conditions that exist in the CICS regions that might make it less eligible to process work. The user can also specify an RTA event with which they can effect routing into an AOR. The full power of RTA can therefore be employed in the routing decision.

Abend avoidance

Abend avoidance provides the facility to preferentially route away from a candidate AOR if that transaction has abended within it in the recent past. After a given time, CICSPlex SM will reactivate the AOR and send a ‘sacrificial lamb’ into the AOR to test whether the problem still exists.

If no abend occurs, then the AOR is brought back ‘into the fold’ and it will be reactivated for transaction routing.

PHYSICAL REALITY

So far we have talked about dynamic routing in a fairly abstract sense. In reality, routers, requestors, and target regions can be in a single MVS image, multiple MVS images, or even multiple sysplexes. We must now address the issue of communicating (or not) the various pieces of information that are required to manage these workloads.

MULTIPLE CICS IN A SINGLE MVS IMAGE

Within a single MVS image, MVS dataspaces can be used to communicate and store data for access by the appropriate components in the management, routing, and target address spaces. Active affinities can be kept here, along with load counts for the various targets.

MULTIPLE MVS IMAGES IN A SYSPLEX

We now have the possibility of routing from routers in one MVS image to targets in another MVS image. Whilst information required in one image can be maintained in a dataspace, various cross-MVS interactions are necessary. An example would be the creation of an affinity that needs to be broadcast (eg system). This also requires cross-MVS system locking. Other pieces of information can be propagated via the management network and acted upon by the workload management component, so long as the frequency of propagation is not large. Information such as system health data and RTA events fall into this category. Finally, the routers need to know about loads within the target systems. For non-local targets, the propagation frequency would impact performance. For this reason, locally-routed counts are maintained, which are updated by actual counts as part of the heartbeat mechanism for the systems concerned.

Since we want to be able to cope with multi-sysplex workloads, we do not use the coupling facility to transmit this information. It is propagated via the CMAS-to-CMAS communication network (using CICS MRO, MRO/XCF, ISC facilities).

MULTIPLE SYSPLEX

Given that we do not ‘prereq’ any sysplex facilities, there are no further considerations to be made here, beyond the longer propagation times encountered by going over potentially geographically remote distances.

NEXT ARTICLE

In the next article we will look at the CICS and CICSPlex SM model, and the mechanics of affinity management and abend avoidance.

Dr Paul Johnson
CICS Transaction Server Systems Management Planning/Development
IBM (UK)

CICS and MQSeries: ReplyToQueue facility

INTRODUCTION

As we saw in Issue 203 of *CICS Update*, October 2002, it is quite simple to test the Bridgemonitor of the MQSeries in CICS by using the program CRMQTE01. This program works without a ReplyToQueue facility. Because of this, the program CRMQTE02 was written, which covers all requests from a modern MQSeries application. You can use the same RequestQueue as in program CRMQTE01. Also, the CICS program CIT000 can be used unchanged. An additional **define** from a ReplyToQueue is necessary, as shown in the following example. Also the program CRMQTE02 must be compiled as an MQSeries batch program. It was developed and tested under OS/390 2.10 in 64-bit mode.

CRMQTE02

```
CRMQTEØ2 TITLE 'BATCH INTERFACE: TEST THE 327Ø BRIDGE VIA MQSERIES    '
***********************************************************************
```

```
*                                                                    *
* NAME:        CRMQTEØ2                                              *
*                                                                    *
* FUNCTION:    THIS PROGRAM PUTS MESSAGES TO MQSERIES FOR LATER      *
*              PROCESSING IN CICS.                                   *
*<==================================================================>*
*              MQSERIES FORCES THE USE OF A "REPLYTOQUEUE" WHICH     *
*              IS NOT NEEDED AS A PART OF THIS APPLICATION!!!!!!     *
*<==================================================================>*
*                                                                    *
*              THE PROGRAM IS CALLED VIA THE BATCH.                  *
*                                                                    *
*              A TRANSACTION (TØØØ) IS STARTED VIA THE 327Ø BRIDGE.  *
*                                                                    *
**********************************************************************
         DSECT
         CVT    DSECT=YES,LIST=NO     COMMUNICATIONS VECTOR TABLE
         SPACE
CRMQTEØ2 CSECT
CRMQTEØ2 AMODE ANY
CRMQTEØ2 RMODE 24
         SPACE
**********************************************************************
*        INITIALISATION                                              *
**********************************************************************
         SPACE
         STM   R14,R12,12(R13)         SAVE CALLER'S REGISTERS
         USING CRMQTEØ2,CODEREG,CODEREG2 ESTABLISH ADDRESSABILITY
         LR    CODEREG,R15             LOAD BASE REGISTER
         LR    CODEREG2,R15            LOAD BASE REGISTER
         AH    CODEREG2,H4Ø96          LOAD BASE REGISTER
         ST    R13,SAVEAREA+4
*
         LR    R2,R13
         LA    R13,SAVEAREA
         ST    R13,8(,R2)
*
         MVC   PARM,Ø(R1)
         B     MAIN
         DC    CL22'**CRMQTEØ2**CRMQTEØ2**'
         SPACE
H4Ø96    DC    H'4Ø96'
SAVEAREA DS    18F
         SPACE
**********************************************************************
*        MAINLINE                                                    *
**********************************************************************
         SPACE
MAIN     DS    ØH
```

```
         BAS   SUBREG,READ                   READ CARD VIA SYSIN
         BAS   SUBREG,MQCONN                 CONNECT MQSERIES
         BAS   SUBREG,MQOPEN                 OPEN THE QUEUE
         BAS   SUBREG,MQPUT                  PUT QUEUE
         BAS   SUBREG,MQCLOSE                CLOSE THE QUEUE
         BAS   SUBREG,MQDISC                 DISCONNECT FROM MQSERIES
         SPACE
MAINTERM DS    ØH
         L     R13,SAVEAREA+4
         LM    R14,R12,12(R13)               RESTORE CALLERS REGISTER
         XR    R15,R15                       CLEAR REGISTER 15
         BR    R14                           RETURN TO CALLER
         SPACE
***********************************************************************
*        READ CONTROL RECORD FROM SYSIN                               *
***********************************************************************
         SPACE
READ     EQU   *
         MVC   TRACE,=CL8'READ'
         WTO   'CRMQTEØ2: READ'
         OPEN  (SYSIN,(INPUT))
         GET   SYSIN,IOAREA
         CLC   IOAREA(4),=CL4'PROD'          PRODUCTION
         BE    READ1ØØØ
         CLC   IOAREA(4),=CL4'SYST'          SYSTEM
         BE    READ1ØØØ
         CLC   IOAREA(4),=CL4'TEST'          TEST
         BE    READ1ØØØ
         CLC   IOAREA(2),=CL2'AE'            TEST SPECIAL
         BE    READ1ØØØ
         CLC   IOAREA(4),=CL4'VPRD'          VPRD
         BE    READ1ØØØ
         B     READ_ERROR
         SPACE
READ1ØØØ EQU   *
         MVC   QMGRNAME,BLANKS               USE DEFAULT QUEUE MANAGER
         MVC   MQUEUE+5(4),IOAREA            MQ QUEUE
         MVC   MQUEUER+5(4),IOAREA           MQ REPLY TO QUEUE
         CLOSE (SYSIN)
         MVI   OPEN,C'Y'
         WTOE  'SYSIN=',IOAREA
         BR    SUBREG
         SPACE
READ_ERROR     EQU *
         WTOE  'INVALID PARAMETER SYSIN=',IOAREA
         BR    SUBREG
         SPACE
SYSIN    DCB   DDNAME=MQTESTIN,DSORG=PS,MACRF=GM,EODAD=READ_ERROR
         SPACE
```

```
IOAREA   DS    CL8Ø' '
         EJECT
***********************************************************************
*        CONNECT TO MQ MANAGER                                        *
***********************************************************************
         SPACE
MQCONN   EQU   *
         MVC   TRACE,=CL8'MQCONN'
         WTO   'CRMQTEØ2: MQCONN'
         XC    HCONN,HCONN
         CALL  MQCONN,                                                 +
               (QMGRNAME,HCONN,COMPCODE,REASON),                       +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
WTO   'CRMQTEØ2: ERROR DURING MQCONN!'
         BR    SUBREG
         EJECT
***********************************************************************
*        CLOSE A QUEUE                                                *
***********************************************************************
         SPACE
MQCLOSE  EQU   *
         MVC   TRACE,=CL8'MQCLOSE'
         WTO   'CRMQTEØ2: MQCLOSE'
         LA    RØ,MQCO_NONE
         ST    RØ,OPTIONS
         CALL  MQCLOSE,                                                +
               (HCONN,HOBJ,OPTIONS,COMPCODE,REASON),                   +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
         WTO   'CRMQTEØ2: ERROR DURING MQCLOSE!'
         BR    SUBREG
         EJECT
***********************************************************************
*        DISCONNECT FROM MQ MANAGER                                   *
***********************************************************************
         SPACE
MQDISC   EQU   *
         MVC   TRACE,=CL8'MQDISC'
         WTO   'CRMQTEØ2: MQDISC'
         CALL  MQDISC,                                                 +
               (HCONN,COMPCODE,REASON),                                +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
         WTO   'CRMQTEØ2: ERROR DURING MQDISC!'
         BR    SUBREG
         EJECT
***********************************************************************
*        GET A MESSAGE FROM A QUEUE                                   *
***********************************************************************
         SPACE
MQGET    EQU   *
         MVC   TRACE,=CL8'MQGET'
         WTO   'CRMQTEØ2: MQGET'
         LA    RØ,MYMD                  MVCL TO-OPERAND
         LA    R1,MYMD_LENGTH           MVCL TO-LENGTH(MAXIMUM)
         LA    R2,MQMD                  MVCL FROM-OPERAND
         LA    R3,MQMD_LENGTH           MVCL FROM-LENGTH
         MVCL  RØ,R2                    MVCL
         MVC   MYGMO_AREA,MQGMO_AREA
         MVC   FLENGTH,=AL4(QEND-QSTRT)
         CALL  MQGET,                                                  +
               (HCONN,HOBJ,MYMD,MYGMO,FLENGTH,QUEUECO,FLENGTH,         +
               COMPCODE,REASON),                                       +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
         WTO   'CRMQTEØ2: ERROR DURING MQGET!'
         BR    SUBREG
         EJECT
***********************************************************************
*        OPEN A QUEUE                                                 *
***********************************************************************
         SPACE
MQOPEN   EQU   *
         MVC   TRACE,=CL8'MQOPEN'
         WTO   'CRMQTEØ2: MQOPEN'
         LA    RØ,MYOD                  MVCL TO-OPERAND
         LA    R1,MYOD_LENGTH           MVCL TO-LENGTH(MAXIMUM)
         LA    R2,MQOD                  MVCL FROM-OPERAND
         LA    R3,MQOD_LENGTH           MVCL FROM-LENGTH
         MVCL  RØ,R2                    MVCL
         LA    RØ,MQOO_OUTPUT
         ST    RØ,OPTIONS
         LA    RØ,MQOT_Q
         ST    RØ,MYOD_OBJECTTYPE
         MVC   OBJECTNAME,MQUEUE
         MVC   MYOD_OBJECTNAME,OBJECTNAME
         CALL  MQOPEN,                                                 +
               (HCONN,MYOD,OPTIONS,HOBJ,COMPCODE,REASON),              +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
         WTO   'CRMQTEØ2: ERROR DURING MQOPEN!'
         BR    SUBREG
         EJECT
***********************************************************************
*        PUT A MESSAGE ON A QUEUE                                     *
***********************************************************************
         SPACE
MQPUT    EQU   *
         MVC   TRACE,=CL8'MQPUT'
         WTO   'CRMQTEØ2: MQPUT'
         LA    RØ,MYMD                   MVCL TO-OPERAND
         LA    R1,MYMD_LENGTH            MVCL TO-LENGTH(MAXIMUM)
         LA    R2,MQMD                   MVCL FROM-OPERAND
         LA    R3,MQMD_LENGTH            MVCL FROM-LENGTH
         MVCL  RØ,R2                     MVCL
*
         MVC   MYPMO_AREA,MQPMO_AREA
*
         MVC   FLENGTH,=AL4(QEND-QSTRT)
*
         MVC   MYMD_CORRELID,MQCI_NEW_SESSION
         MVC   MYMD_FORMAT,=CL8'MQCICS'
         MVC   MYMD_REPLYTOQ,MQUEUER
*
         MVC   MYCIH_AREA,MQCIH_AREA
         MVC   MYCIH_FORMAT,=C'CSQCBDCI'
         MVC   MYCIH_LINKTYPE,MQCLT_TRANSACTION
         MVC   MYCIH_TRANSACTIONID,TESTTRAN
*
         CALL  MQPUT,                                                  +
               (HCONN,HOBJ,MYMD,MYPMO,                                 +
               FLENGTH,MYCIH_AREA,COMPCODE,REASON),                    +
               MF=(E,CALLLIST),VL
         LA    RØ,MQCC_OK
         C     RØ,COMPCODE
         BER   SUBREG
         WTO   'CRMQTEØ2: ERROR DURING MQPUT!'
         BR    SUBREG
         EJECT
***********************************************************************
*        DYNAMIC STORAGE                                              *
***********************************************************************
         SPACE
MQUEUE   DC    CL48'CICS.XXXX.ISBSTCX.FROM.BATCH.TEST.QØØ1.REQUEST'
MQUEUER  DC    CL48'CICS.XXXX.ISBSTCX.FROM.BATCH.TEST.QØØ1.REPLY'
CICSID   DS    CL4
         SPACE
         CMQA   LIST=YES                EQUATES FOR MQ CONSTANTS
         CMQODA DSECT=NO,LIST=YES       OBJECT DESCRIPTER
         CMQMDA DSECT=NO,LIST=YES      MESSAGE DESCRIPTOR
         CMQPMOA DSECT=NO,LIST=YES     PUT MESSAGE OPTIONS
         CMQGMOA DSECT=NO,LIST=YES     GET MESSAGE OPTIONS
         CMQCIHA DSECT=NO,LIST=YES     CICS INFORMATION HEADER
         SPACE
PARM     DS    F
LOADADDR DC    A(Ø)
         SPACE
***********************************************************************
*        PROGRAM CONSTANTS                                            *
***********************************************************************
         SPACE
BLANKS         DC   256C' '
NULLS          DC   Ø16X'ØØ'
ZEROS          DC   Ø16C'Ø'
*
DATETIME       DS   CL16
LENGTH         DS   H
FLENGTH        DS   F
TRACE          DS   CL8
WORK           DS   CL16
OPEN           DS   CL1
*
MQCI_NEW_SESSION  DC
X'414D51214E45575F53455353494F4E5F434F5252454C4944X
               '
*
               DS   ØD
EYEC1          DC   CL8'**COCO**'
COMPCODE       DS   F                 COMPLETION CODE
REASON         DS   F                 REASON CODE QUALIFYING COMPCODE
HCONN          DS   F                 CONNECTION HANDLE
HOBJ           DS   F                 OBJECT HANDLE
EYEC2          DC   CL8'**CPCP**'
QMGRNAME       DS   CL48              QUEUE MANAGER NAME
OBJECTNAME     DS   CL48              OBJECT NAME (QUEUE NAME)
OPTIONS        DS   F                 COMMAND OPTIONS
TESTTRAN       DC   C'TØØØ'           CICS TRANSACTION
CALLLIST       CALL ,(Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø),VL,MF=L
         SPACE
MYOD     CMQODA  LIST=YES             OBJECT DESCRIPTOR
MYMD     CMQMDA  LIST=YES             MESSAGE DESCRIPTOR
MYPMO    CMQCIHA LIST=YES             PUT MESSAGE OPTIONS FOR CICS
MYGMO    CMQCIHA LIST=YES             GET MESSAGE OPTIONS FOR CICS
SPACE
QSTRT    DS   ØC                      "COMPLETE CICS HEADER"
MYCIH    CMQCIHA LIST=YES             CICS INFORMATION HEADER
QUEUECO  DC   C'TESTQUEUE FOR CRMQTEØ2'
QEND     DS   ØC
         SPACE
         DFHREGS
         SPACE
SUBREG   EQU   R9                  BAS REGISTER
CODEREG  EQU   R1Ø                 BASIS REGISTER
CODEREG2 EQU   R11                 BASIS REGISTER 2
         SPACE
         DC    C' '
         END   CRMQTEØ2
```

RUN CRMQTE02 JCL

```
//YOURUSRX JOB ØØ2665,'YOUR NAME',NOTIFY=YOURUSR,
//*─────────────────────────────────
//*   JOB  SUBMITTED FROM YOURUSR.MAIN.JCL(CRMQTEØ2)
//*   DOC: TEST "CICS AND MQSERIES VIA THE 327Ø BRIDGE"
//*   GRP:
//*   DATE: 27.Ø7.Ø2, TIME: ØØ:ØØ
//*─────────────────────────────────
//             CLASS=T,USER=YOURUSR,MSGCLASS=X,REGION=4M,RESTART=*
//*─────────────────────────────────
/*ROUTE PRINT U28
//*────────────────────────────────*
//CRMQTEØ2 EXEC PGM=CRMQTEØ2
//STEPLIB  DD DSN=YOURUSR.MAIN.LOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//MESSAGE  DD SYSOUT=*
//MQTESTIN DD *
TEST
/*
//
```

Claus Reis
CICS Systems Programmer
Nuernberger Lebensversicherung AG (Germany)

CICS file control enhancements

INTRODUCTION

CICS Transaction Server for z/OS V2.2 has enhanced support for VSAM file operations. CICS file control processing can now release exclusive control of Control Intervals (CIs) after having performed an EXEC CICS READ UPDATE request. In turn, this can avoid the potential for exclusive control conflicts occurring in the interval between the EXEC CICS READ UPDATE and its corresponding EXEC CICS REWRITE command. This article discusses the background to CI management within CICS and the benefits that this enhancement can bring to the system.

Please note that VSAM Record Level Sharing (RLS) does not externalize the use of CIs; this enhancement is therefore specific to non-RLS VSAM file access by CICS.

BACKGROUND TO FILE CONTROL PROCESSING

CICS provides support for its applications to manipulate the state of VSAM and BDAM datasets. (Note: this article is specific to enhancements to the VSAM access method.) CICS provides its applications with an abstracted view of the resources by making the API work against files instead of datasets. In CICS terms, a file is the resource that applications can read records from, write records to, delete records from, etc; it has a 1 to 8-character resource name. A VSAM dataset is the underlying resource managed by the access method; it has a name string of up to 44 characters in total. The underlying VSAM datasets can be key-sequenced (KSDSs), entry-sequenced (ESDSs), or relative-record sequenced (RRDSs).

The CICS file control API provides user programs with the ability to exploit many VSAM functions against files defined to the CICS system. CICS recovery management services provide additional functionality over this, to coordinate recoverable changes made to files by Units Of Work (UOWs). CICS file definitions can specify no recovery, backwards recovery, or both backwards and forwards recovery. These options result in the logging of before and after images of recoverable changes made to the files. Before image log records are used by CICS during dynamic transaction backout if a task abends, or during system recovery if an emergency restart occurs. After image log records are used by an offline recovery utility such as IBM’s CICSVR product, to rebuild changes to a VSAM sphere after (for example) a DASD failure such as a head-crash.

CICS allows applications to modify VSAM files at the record level; for example, a KSDS can be modified by the addition, alteration, or deletion of particular keyed records. These records reside within particular CIs on the underlying VSAM dataset. During a file control operation, CICS will maintain a lock on a particular record in order to prevent other requests from changing the record state at the same time. For non-recoverable files, this record lock is maintained for the scope of the operation (eg write to the file, delete from the file, read for update, and rewrite of a record to the file). For recoverable files, the record lock is maintained until the UOW that made the file changes completes, and terminates with a syncpoint operation.

In addition to the record lock (maintained by CICS), VSAM itself will maintain exclusive control of the CI containing the record being manipulated. This control is held until an EXEC CICS UNLOCK (endreq operation) or EXEC CICS REWRITE is performed by CICS. Prior to CICS TS 2.2, the period between an EXEC CICS READ UPDATE of a record and the following EXEC CICS REWRITE of the record by the application will see the CI containing the record held in exclusive control by VSAM for use by that particular file control update request within CICS. The state of the request is maintained internally by VSAM. If other tasks in the CICS system needed to acquire ownership of the same CI for their own purposes, they would suffer a VSAM exclusive control conflict and be suspended until the original task had released its ownership of the CI. (Note: the need for shared or exclusive ownership of a CI relates to whether the file is defined to use LSR or NSR – ie to use VSAM local shared resources or non-shared resources. For files defined to use LSR, reads and browses against a CI will result in exclusive control conflicts if that CI is currently in use by another VSAM operation that requires exclusive use of the CI. For NSR files, reads and browses will use a copy of the CI and so not result in exclusive control conflicts.)

Such exclusive control conflicts result in CICS issuing unsolicited exception trace entries (trace point AP 04BA) for diagnostic purposes. These can be useful when investigating persistent exclusive control conflict situations.

CONTROL INTERVAL LOCKING AND THE CILOCK SIT OPTION

Prior to CICS TS 2.2, file control processing of an EXEC CICS READ UPDATE command was as follows. CICS would set up the VSWA for the task issuing the command, which would include manipulating the VSAM RPL to define the request to the access method. A CICS string would be acquired and a VSAM get update request would be issued to retrieve the record. This would cause VSAM to acquire exclusive control of the CI containing the record being read. CICS would then obtain its own lock on the record that had been read (to single-thread requests against the record until the EXEC CICS REWRITE had completed, or until syncpoint if the file was defined as recoverable). The lock would be based on the record's primary key. When the corresponding EXEC CICS REWRITE command was issued by the application, CICS file control would issue a VSAM put update request to modify the record. VSAM would release the exclusive ownership of the CI lock at completion of this. CICS would then unlock the record (if the file were non-recoverable) and perform housekeeping work such as releasing the string, etc.

In CICS TS 2.2, the situation is subtly different. After the record lock has been acquired during the EXEC CICS READ UPDATE command, CICS will issue a VSAM endreq request to release the exclusive ownership of the CI lock. The CICS string is also released. CICS will then preserve information pertaining to the original EXEC CICS READ UPDATE command so that it can be referenced at the corresponding EXEC CICS REWRITE. When this occurs, CICS file control can reissue a VSAM get update request to reestablish ownership of the CI. To do so it first acquires a CICS string, then issues the VSAM get update to honour the EXEC CICS REWRITE command, before completing as before.

Note: this enhancement to CI management also applies to the case of EXEC CICS READ UPDATE commands followed by the corresponding EXEC CICS DELETE commands, when no Ridfld (key) value is specified on the DELETE.

In making this change, the potential for exclusive control conflicts against the CI in question can be avoided. There could be a relatively long period between an EXEC CICS READ UPDATE and its corresponding EXEC CICS REWRITE command (perhaps even encompassing terminal or Web-based I/O, which is conversational behaviour and so not encouraged). Other tasks could need to reference different records within the same CI as the one to be updated. Note that requests against the same record would be forced to wait, since CICS has maintained its own record lock on the key of the record to be modified. However, there could be many different records within the same CI, and other tasks are now able to acquire exclusive control of the CI in order to modify these during the period between the EXEC CICS READ UPDATE and EXEC CICS REWRITE commands.

This enhancement can bring considerable benefits in transaction throughput for CICS systems that experience exclusive control conflicts between tasks on a regular basis. However, it may be that such events are rare, or do not occur, in some customer environments. As such, the benefit in releasing exclusive control of CIs on completion of EXEC CICS READ UPDATE commands in this way would not be seen, and the corresponding need to reacquire CI ownership at the time of the EXEC CICS REWRITE would be an overhead. Therefore, in order to provide compatibility support with the original CICS mechanism of CI management (where exclusive control was retained between the EXEC CICS READ UPDATE and corresponding EXEC CICS REWRITE commands), CICS TS 2.2 provides a new SIT parameter, CILOCK. This can be set to either YES or NO. If set to YES, CI locking is maintained when an EXEC CICS READ UPDATE command exits from CICS file control processing, and hence CICS handles such requests in the same manner as in previous releases. If set to NO, the new mechanism of releasing exclusive control of a CI after exiting from EXEC CICS READ UPDATE processing (and reacquiring it on the subsequent EXEC CICS REWRITE command) is implemented. Note that the default for the CILOCK SIT parameter is NO, so the new SIT option does not have to be specified in order to benefit from this file control enhancement.

PERFORMANCE IMPLICATIONS

On CICS systems which regularly see exclusive control conflicts between tasks performing updates to VSAM files, this enhancement can be expected to improve transaction throughput by avoiding the internal CICS task suspension and redispatching activity required to handle such situations. Quantification of the benefit would depend upon a number of factors. For example, the access pattern of records within CIs would have a bearing on the benefit to be seen. The type of resource sharing on the file definitions (LSR or NSR) has a bearing too, since LSR files would suffer from exclusive control conflicts when multiple reads or browses are issued against a given CI. Also, the size of records in relation to the CI size would be a factor. Clearly, concurrent updates to many small records within a given CI would give rise to greater numbers of exclusive control conflicts than if there were fewer (but larger) records in the CI. This can be extrapolated to the case of single records per CI, which (by definition) cannot result in exclusive control conflicts within CICS. Only requests against the particular record's key will pertain to this CI, and CICS will single-thread such requests because of its own internally-managed record lock anyway. In such an environment, CILOCK=YES should be used, since there is no benefit in releasing exclusive control of a CI between EXEC CICS READ UPDATE and EXEC CICS REWRITE commands.

CICS statistics and monitoring data should be used, together with analysis of VSAM dataset attributes such as CI and record sizes, when determining what effects this change to CICS file control processing may have upon system performance and throughput. For example, those systems which regularly suffered from exclusive control conflicts should see an improvement in their ratio of CI conflicts to transaction rates, and a corresponding reduction in the CPU time consumed per transaction. Conversely, if transaction throughput were seen to be degraded, and/or CPU time per transaction seen to increase, this would indicate that few (or no) exclusive control conflicts were present on a system originally, and hence there was no benefit in releasing control of a CI between EXEC CICS READ UPDATE and EXEC CICS REWRITE commands. If so, CILOCK=YES may be specified in the SIT.

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

Check start-up type for your production CICS

In my shop we always force the start-up of our production CICS to cold, even when we can start it warm. We use a program called CHECKGCD (see the *CICS Update* article entitled *Yet another cold start next time*, by myself, published in Issue 182, January 2001, and the revisited version published in Issue 184, March 2001), so that CICS won't use any global catalog information; we don't want our CICS to remember anything about its previous condition. Even an emergency start following an incorrect shutdown is not a good thing because CICS, besides doing the correct recovery operations, restores its own state at the moment of closure by reading the global catalog. This is good if CICS goes down during normal activity, but not so good if an emergency start happens the morning after an incorrect shutdown and before the beginning of the batch: CICS will start with some files that aren't enabled, plus some transactions and/or programs that are disabled. This is a typical pre-batch situation where your CICS applications are not fully active. If this situation arises, your customers will begin to call your customer support complaining about your service – so your service level agreement will go down and your boss will feel justified in stressing you more than he does normally! You will agree that this is not a good start to a working day.

If this does happen to you, the easy solution is to shut down CICS in the correct way and restart it again so it does a cold start (a forced cold start). To do this, your system operators have to recognize the dangerous condition. That is the purpose of this simple program – to notify on the system console the existence of a CICS emergency start.

It is sufficient to insert this program in PLTPI second phase (after DFHDELIM). If an emergency start happens, it performs a write to operator with 'action critical'. It should be sufficient to raise suspicions among your system operators that something has gone wrong. The program also works if started from a terminal, attached to a transaction, in which case it replies with a send text to video; or from a batch job, by an EXCI link, in which case it performs a write to operator with 'action eventual', even if an emergency start has occurred. I tested the program on CICS TS 1.3 but I think it can run on earlier CICS releases with minor changes (ie on CICS ESA initial start doesn't exist).

If your shop does not buy a console message processing control tool or you don't want to ask your boring MVS system programmer to modify his message processing facility to intercept the CICS message 'DFHSI1502I XXXXXXXX CICS startup is Emergency' (and then call a program testing whether the message applid belongs to a production CICS), you can use this program in the PLTPI of your production CICS.

CHKSTART

```
*ASM XOPTS(SP)
         TITLE '** CHKSTART - CHECK CICS STARTUP TYPE **'
*
DFHEISTG DSECT
***********************************************************************
* Message declarations.
***********************************************************************
MSG_RESP       DS  F                   Response code from EXEC CICS
MSG_BUFFER     DS  ØCL98               Message buffer
MSG_PROG       DS  CL8                 Program name
               DS  CL1                 Blank space
MSG_APPL       DS  CL8                 CICS applid
               DS  CL1                 Blank space
MSG_TEXT       DS  CL8Ø                Message text
*
         ORG   MSG_TEXT
END_MSG_AUTO   DS  ØCL8Ø
               DS  CL16
STARTUP_TYPE   DS  CL1Ø
               DS  CL54
*
         ORG   MSG_TEXT
ERROR_MSG_AUTO DS  ØCL8Ø
               DS  CL34
ERROR_TEXT     DS  CL46
*
***********************************************************************
* Various other variables.
***********************************************************************
RESP           DS  F                   Response code from EXEC CICS
STARTCODE      DS  CL2                 Facility type
CICSSTATUS     DS  F                   Current CICS status
COLDSTATUS     DS  F                   Cold start type
```

```
STARTSTATUS    DS  F                  Start CICS status
ACTION         DS  F                  WTO messsage type
STARTUP_TPE    DS  CL1Ø               Startup type message text
ERROR_TXT      DS  CL46               Error message text
*
DFHEIBLK DSECT
*
         DFHREGS ,
*
***********************************************************************
* CHKSTART mainline code.
***********************************************************************
CHKSTART AMODE 31
CHKSTART RMODE ANY
CHKSTART CSECT
***********************************************************************
* Obtain the program name and CICS applid for messages.
* In addition, obtain the startcode type.
***********************************************************************
         MVC   MSG_BUFFER,SPACES
         MVC   ACTION,DFHVALUE(EVENTUAL)
LBASSIGN EQU   *
         EXEC  CICS ASSIGN                                            *
                PROGRAM(MSG_PROG)                                     *
                APPLID(MSG_APPL)                                      *
                STARTCODE(STARTCODE)                                  *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBEØØ1                 Assign error
*
***********************************************************************
* Obtain the type of CICS startup (emergency, warm, cold, or initial)
* and the CICS status (startup, active, firstquiesce, or finalquiesce).
***********************************************************************
LBINQSYS EXEC  CICS INQUIRE SYSTEM                                    *
                CICSSTATUS(CICSSTATUS)                                *
                STARTUP(STARTSTATUS)                                  *
                COLDSTATUS(COLDSTATUS)                                *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBEØØ2                 Inquire system error
         CLC   STARTSTATUS,DFHVALUE(EMERGENCY)
         BE    LBNØØ4                 Emergency start
         CLC   STARTSTATUS,DFHVALUE(WARMSTART)
         BE    LBNØØ2                 Warm start
         CLC   STARTSTATUS,DFHVALUE(COLDSTART)
         BE    LBCLDSTR
         B     LBEØØ3                 Unknown startup type
*
```

```
***********************************************************************
* Cold Start: initial or cold.
***********************************************************************
LBCLDSTR CLC   COLDSTATUS,DFHVALUE(INITIAL)
         BE    LBNØØ3                  Initial start
         CLC   COLDSTATUS,DFHVALUE(COLD)
         BE    LBNØØ1                  Cold start
         B     LBEØØ4                  Unknown startup type
*
***********************************************************************
* End of procedure: select where to send output message.
***********************************************************************
LBENDRØ  EQU   *
         CLC   STARTCODE,TERMINAL      Start on terminal ?
         BNE   WTO_MSG
         B     SEND_MSG
*
***********************************************************************
* End the program and return to CICS.
***********************************************************************
LBRETURN EQU   *
         EXEC  CICS RETURN
*
***********************************************************************
* Procedure to issue a message on screen.
***********************************************************************
SEND_MSG DS    ØH
         EXEC  CICS SEND TEXT                                          *
                    FROM(MSG_BUFFER) LENGTH(L'MSG_BUFFER)              *
                    ERASE FREEKB                                       *
                    RESP(MSG_RESP)
         B     LBRETURN
*
***********************************************************************
* Procedure to issue a message on console.
***********************************************************************
WTO_MSG  DS    ØH
         EXEC  CICS WRITE OPERATOR                                     *
                    TEXT(MSG_BUFFER) TEXTLENGTH(L'MSG_BUFFER)          *
                    ACTION(ACTION)                                     *
                    RESP(MSG_RESP)
         B     LBRETURN
*
***********************************************************************
* Labels to branch to set startup type message.
***********************************************************************
LBNØØ1   MVC   STARTUP_TPE,=CL1Ø'Cold.     '
         B     LBNMSG
LBNØØ2   MVC   STARTUP_TPE,=CL1Ø'Warm.     '
```

```
         B     LBNMSG
LBNØØ3   MVC   STARTUP_TPE,=CL1Ø'Initial.  '
         B     LBNMSG
LBNØØ4   MVC   STARTUP_TPE,=CL1Ø'Emergency.'
         CLC   CICSSTATUS,DFHVALUE(STARTUP)   Invoked in PLT phase?
         BNE   LBNMSG
         MVC   ACTION,DFHVALUE(CRITICAL)      Yes, WTO read message
         B     LBNMSG
*
***********************************************************************
* Labels to branch to when a particular error occurs.
***********************************************************************
LBEØØ1   MVC   ERROR_TXT,=CL46'Assign Failure.'
         B     LBEMSG
LBEØØ2   MVC   ERROR_TXT,=CL46'Inquire System Failure.'
         B     LBEMSG
LBEØØ3   MVC   ERROR_TXT,=CL46'Unknown Startup Type!'
         B     LBEMSG
LBEØØ4   MVC   ERROR_TXT,=CL46'Unknown Cold Startup Type!'
         B     LBEMSG
*
**********************************************************************
* Write a normal message.
**********************************************************************
LBNMSG   EQU   *
         MVC   END_MSG_AUTO,END_MSG
         MVC   STARTUP_TYPE,STARTUP_TPE
         B     LBENDRØ
*
**********************************************************************
* Write an error message.
**********************************************************************
LBEMSG   EQU   *
         MVC   ERROR_MSG_AUTO,ERROR_MSG
         MVC   ERROR_TEXT,ERROR_TXT
         B     LBENDRØ
*
***********************************************************************
* Constants.
***********************************************************************
SPACES      DC   CL8Ø' '
TERMINAL    DC   CLØ2'TD'              Terminal related task
NOTTERMINAL DC   CLØ2'S '              Nonterminal related task
         LTORG
*
***********************************************************************
* Messages.
***********************************************************************
END_MSG     DC    ØCL8Ø' '
```

```
            DC    CL16'CICS startup is '
            DC    CL1Ø'xxxxxxxxxx'
            DC    CL54' '
ERROR_MSG   DC    ØCL8Ø' '
            DC    CL34'CICS Startup Check Program Error. '
            DC    CL46' '
*
***********************************************************************
* End of CHKSTART.
***********************************************************************
         END CHKSTART
```

Gianluca Bonzano
Systems Programmer
Cedacri Ovest (Italy)

z/OS Version 1.4 announcement

The IBM z/OS Version 1.4 announcement contains an interesting one liner:

The existing support for CICS autoinstall for terminals will be extended to provide a similar support for printer types, thereby reducing the labor associated with printer client changes.

The z/OS TN3270E Server is being enhanced to drive terminal/printer auto-install when the TN3270 Client passes an associated printer.

I confirmed this with IBM.

J P Lemmon
Lemon-Tree (UK)

Automatic control for the CICS ISC connections

INTRODUCTION

The control of the CICS ISC connections is often a problem if the number of CICS sessions to manage is large.

When the ISC sessions are unavailable between several CICS sessions, it can be very difficult to find the inactive sessions.

ISC sessions can become unavailable for various reasons including: closing/restarting the network, different CICS service periods, and the state of the connections in failure (eg OUTSERVICE). In these circumstances, finding the inactive sessions is a complicated task.

The utility that follows has as an objective the control of the state of the ISC connections and the notification of any inactive connections.

DESCRIPTION

This tool, developed in REXX, has the following main characteristics:

- The utility becomes active at the start of the CICS session and it stops when CICS closes. The best way to use this tool is to automate its execution. This can be achieved through the NetView environment using the scheduled timer.
- The utility displays ISC inactive sessions and tries to repair them. It executes the CICS command **CEMT INQUIRE MODENAME AVAILABLE(000)** to control the ISC connections that are currently without active sessions. It captures the output from the command and, if there are inactive connections, executes the usual CICS commands **CEMT SET** to repair the connections state. After the **CEMT SET** command, it verifies the state of the connection.
- After the execution of the CICS commands, if the ISC connections are still inactive, it updates a log (sequential dataset) with the name of the currently inactive connections.

- It does not need to know which ISC connections to control; by default it controls all of them.
- It is possible to exclude a specific CICS session from its control by inserting the name of the CICS session in an exclude file.
- It is possible to exclude one or more ISC connections (partial control) from its control by inserting the name of the connection to be excluded in an exclude file.
- It can send notification via SMTP from a mainframe directly to a specific group of people (eg CICS specialists). When it finds an inactive connection and it does not succeed in repairing it, the program can send an e-mail containing the identities of ISC connections that have problems. This feature requires the presence in the subarea host of a TCP/IP stack.
- It records all the activities that it carries out in log files.

This utility has been developed and used for the control of over 200 CICS Transaction Server regions (Version 1.3.0).

CONCLUSION

With this tool we have the ability to act directly on CICS as soon as problems are introduced. It is simple to operate and maintain and reduces the actions necessary for the control of the state of the CICS ISC connections.

REXX EXECS

ISCCX000

```
/*   ISCCXØØØ
      Executed for every CICS start. It carries out the setting of the
      timer to control ISC
      connections.
*/
Trace ?o
Arg ncics
lparm = 'CICSTS.ISCCICS.EXCLUDE'
rfound = Ø
```

```
tcics = strip(substr(ncics,3,6))
idtimera = 'A'| |tcics
idtimerb = 'B'| |tcics
/* _______________________________________________________

      Alloc and read CICS exclude table
   _______________________________________________________ */
say time() ' ISCCXØØØ - Allocation table for 'ncics '...'
Address Netview
"Free file(iscØØ)"
"Alloc dataset('"lparm"') file(iscØØ) shr free"
if rc ¬= Ø then Do
                 Call ISC_Alloc_Error
                 Exit
                End
say time() ' ISCCXØØØ – Read table for 'ncics '...'
ADDRESS MVS
  "NEWSTACK"
  "EXECIO * DISKR iscØØ (STEM iscr. FINIS"
  if rc ¬= Ø then Do
                   Call ISC_Read_Error
                   Exit
                  End
  "DELSTACK"
do i = 1 to iscr.Ø
    rtyp = substr(iscr.i,1,1)
    if rtyp ¬= '_' then iterate
    excl_cics = word(iscr.i,2)
    if excl_cics = ncics then do
                           excl_conid1 = word(iscr.i,3)
                           if excl_conid1 = '****' | excl_conid1 = ''
then do
/* ______________________________________________________________

   CICS found in exclude file. It deletes timer, if present,
   and it does not carry out the timer setting to control.
________________________________________________________________ */
Interpret "'EXCMD AUTO2 PURGE TIMER="IDTIMERB",OP=AUTO2'"
                     say '>>> ISCCXØØØ – ISC control excluded for 'ncics
                               rfound = 1
                               leave
                               End
                              Else do
                                    Call ISC_Schedula
                               rfound = 1
                               leave
                               End
                          End
                     else iterate
```

```
End
if rfound = Ø then Call ISC_Schedula
Exit
ISC_Schedula:
/* _________________________________________________________________

   Setting timer to control ISC connections.
   The first execution is after 1 minute and
   Then every one hour.
__________________________________________________________________ */
Address Netview
Interpret "'EXCMD AUTO2 PURGE TIMER="IDTIMERB",OP=AUTO2'"
Interpret "'EXCMD AUTO2 AFTER Ø1,ID="IDTIMERA",SAVE,ISCCXØ1Ø" NCICS"'"
Interpret "'EXCMD AUTO2 EVERY Ø1:ØØ:ØØ,ID="IDTIMERB",SAVE,ISCCXØ1Ø"
NCICS"'"
say '>>> ISCCXØØØ – ISC control scheduled for 'ncics
Return
ISC_Alloc_Error:
/* _________________________________________________________________

   Routine of management errors
   _________________________________________________________________ */
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØØØ – CICS ISC control.                      ***'
say time() '***                Allocation error for the file:     ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                   ***'
say time() '*********************************************************'
"Free file(iscØØ)"
Return
ISC_Read_Error:
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØØØ – CICS ISC control.                      ***'
say time() '***              Read error for the table             ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                   ***'
say time() '*********************************************************'
"Free file(iscØØ)"
Return
```

ISCCX010

```
/*   ISCCXØ1Ø
      It carries out the ISC connections control.
*/
Trace ?o
Arg ncics
```

```
lparm = 'CICSTS.ISCCICS.EXCLUDE'
rfound = Ø
recem  = Ø
conidok = OK
dt =  date(e)
bl = copies(' ',9)
qc = '  * 'ncics' *'
nsys  = mvsvar(sysname)

say time() ' ISCCXØ1Ø - Allocation table for 'ncics '...'
Address Netview
"Free file(isc1Ø)"
"Alloc dataset('"lparm"') file(isc1Ø) shr free"
if rc ¬= Ø then Do
                 Call ISC_Alloc_Error
                 Exit
                End

say time() ' ISCCXØ1Ø – Read table for 'ncics '...'
ADDRESS MVS
  "NEWSTACK"
  "EXECIO * DISKR isc1Ø (STEM iscr. FINIS"
  if rc ¬= Ø then Do
                   Call ISC_Read_Error
                   Exit
                  End
  "DELSTACK"
do i = 1 to iscr.Ø
    rtyp = substr(iscr.i,1,1)
    if rtyp ¬= '_' then iterate
    excl_cics = word(iscr.i,2)
    if excl_cics = ncics then do
                           excl_conid1 = word(iscr.i,3)
                           if excl_conid1 = '****' | excl_conid1 = ''
then do
/* _________________________________________________________________

      CICS in exclude list,  no ISC connections control
   _________________________________________________________________  */
say
  '>>> ISCCXØ1Ø - Exclude list modified. ISC control disabled for 'ncics
                               rfound = 1
                               leave
                               End
                              Else do
                                    Call ISC_Exclude_Conid
                               rfound = 2
                               leave
                               End
```

```
                          End
                     else iterate
End
if rfound = Ø then Call ISC_Control_all
if rfound = 1 then Call ISC_Control_no
if rfound = 2 then Call ISC_Control_parz

Exit

ISC_Exclude_Conid:
/* _________________________________________________________________

   Partial control. Verify which ISC connections to
   exclude from the control.
_________________________________________________________________*/
nconid = words(iscr.i)
Do e=3 to nconid
  x = e-2
  econid.x = word(iscr.i,e)
End
econid.Ø = x
Return
ISC_Alloc_Error:
/* _________________________________________________________________

   Routine of management errors
   _________________________________________________________________ */
"Free file(isc1Ø)"
"Free file(iscl1Ø)"
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØ1Ø - Aoc: control ISC.                      ***'
say time() '***           Error in the allocation of the file:    ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                   ***'
say time() '*********************************************************'
Return
ISC_Read_Error:
Address Netview
“Free file(isc10)”
say time() ‘*********************************************************’
say time() ‘***                                                   ***’
say time() '*** ISCCXØ1Ø - Aoc: control ISC.                      ***'
say time() '***           Error in phase of reading of the chart: ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                .  ***'
say time() '*********************************************************'
Return
ISC_Write_Error:
```

```
Address Netview
"Free file(iscl1Ø)"
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØ1Ø - Aoc: control ISC.                      ***'
say time() '***          Errore in phase writing to the log:      ***'
say time() '***         'lparm'                                   ***'
say time() '***                                                   ***'
say time() '*********************************************************'
Return
ISC_Control_all:
/* _________________________________________________________________

    Complete control. It carries out the control
    of all the ISC connections.
___________________________________________________________________*/
Address Netview
"Free file(isc1Ø)"
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØ1Ø -  ISC control.                          ***'
say time() '***                   Verify all CICS ISC connections ***'
say time() '***                       for 'ncics'.                ***'
say time() '***                                                   ***'
say time() '*********************************************************'
trace ?o
/*___________________________________________________________________

  If CICS field "MODENAME AVAILABLE" equal to Ø = ISC connection problem
  If CICS field "MODENAME AVAILABLE" not equal to Ø = ISC connection
state is OK
  ___________________________________________________________________*/
NRIGHE  = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(ØØØ)' W=5")
DO I = 1 TO NRIGHE
 PARSE UPPER PULL 'MOD(' MODENAME ')' B1 'CON(' CONID ')'  B2 ,
                  'MAX(' NUMMAX ')'   B3 'AVA(' NUMAVA ')' B4 ,
                  'ACT(' NUMACT ')'   B5
 IF NUMAVA = 'ØØØ'  THEN
   DO
    CONIDOK = KO
    "MVS F "ncics",'CEMT SET CONN("CONID") INS REL'"
     WAIT 1Ø SECONDS
    "MVS F "ncics",'CEMT SET CONN("CONID") INS ACQ'"
   END
END
IF CONIDOK = OK THEN Return
/* _____________________________________________________________

    Retry control after CICS command (Cemt SET)
   ____________________________________________________________*/
'WAIT 1Ø SECONDS'
NRIGHE = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(ØØØ)' W=5")
DO I = 1 TO NRIGHE
 PARSE UPPER PULL 'MOD(' MODENAME ')' B1 'CON(' CONID ')'  B2 ,
                  'MAX(' NUMMAX ')'   B3 'AVA(' NUMAVA ')' B4 ,
                  'ACT(' NUMACT ')'   B5
 IF NUMAVA = 'ØØØ' THEN
   DO
     msg='Modegroup 'MODENAME' connection 'CONID' of the 'ncics
     msg=msg||' inactive. To VERIFY |||'
     say msg
     /* __________________________________________________________

        To prepare records for log and mail
        _________________________________________________________ */
     if recem = Ø then do
 rmail.1 = bl'__________ 'dt'   'time()'   'nsys' __________'qc
                        recem = 1
                       End
     recem = recem + 1
     rmsg = substr(msg,1,57)
     rmail.recem = rmsg
   END
END
rmail.Ø = recem
Call Send_Mail
Return
ISC_Control_no:
/* ___________________________________________________________

   CICS in exclude list. No ISC connection control
   ___________________________________________________________*/
Address Netview
"Free file(isc1Ø)"
say time() '***********************************************************'
say time() '***                                                      ***'
say time() '*** ISCCXØ1Ø - ISC control.                              ***'
say time() '***                    Exclude list active.              ***'
say time() '***                    No control to 'ncics'.            ***'
say time() '***                                                      ***'
say time() '***********************************************************'
Return
ISC_Control_parz:
/* ___________________________________________________________

    Some connections in exclude file, carry out partial control.
   ___________________________________________________________*/
Address Netview
```

```
"Free file(isc1Ø)"
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCXØ1Ø -  ISC control.                          ***'
say time() '***                  Partial control for connections  ***'
say time() '***                      of the 'ncics'.              ***'
say time() '***                                                   ***'
say time() '*********************************************************'
trace ?o

NRIGHE  = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(ØØØ)' W=5")
DO I = 1 TO NRIGHE
 PARSE UPPER PULL 'MOD(' MODENAME ')' B1 'CON(' CONID ')'  B2 ,
                  'MAX(' NUMMAX ')'   B3 'AVA(' NUMAVA ')' B4 ,
                  'ACT(' NUMACT ')'   B5
Do k = 1 to x
  IF NUMAVA = 'ØØØ' & econid.k ¬= conid THEN sw = Ø
                                        else do
                                              sw = 1
                                              leave
                                             end
 End
 If sw = Ø then DO
                 CONIDOK = KO
                 "MVS F "ncics",'CEMT SET CONN("CONID") INS REL'"
                 WAIT 1Ø SECONDS
                 "MVS F "ncics",'CEMT SET CONN("CONID") INS ACQ'"
                END
           else sw = Ø
END
IF CONIDOK = OK THEN Return
'WAIT 1Ø SECONDS'
NRIGHE = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(ØØØ)' W=5")
DO I = 1 TO NRIGHE
 PARSE UPPER PULL 'MOD(' MODENAME ')' B1 'CON(' CONID ')'  B2 ,
                  'MAX(' NUMMAX ')'   B3 'AVA(' NUMAVA ')' B4 ,
                  'ACT(' NUMACT ')'   B5
Do k = 1 to x
  IF NUMAVA = 'ØØØ' & econid.k ¬= conid THEN sw = Ø
                                        else do
                                              sw = 1
                                              leave
                                             end
 End
 If sw = Ø & conid ¬= '' then DO
              msg='Modegroup 'MODENAME' connection 'CONID' of the 'ncics
                 msg=msg||' inactive. To Verify |||'
                 say msg
if recem = Ø then do
```

```
rmail.1 = bl'__________ 'dt'   'time()'   'nsys'___________'qc'   <P>'
                                    recem = 1
                                   End
                 recem = recem + 1
                 rmsg = substr(msg,1,57)
                 rmail.recem = rmsg
                END
If sw = 1 & conid ¬= '' then DO
                 msg='Modegroup 'MODENAME' connection 'CONID' of the
'ncics
                 msg=msg||' NOT ACTIVE and in EXCLUDE LIST, Ignore |||'
                 say msg
                END
End
rmail.Ø = recem
Call Send_Mail
Return
/* _______________________________________________________

   Update log file and  send e-mail
   _______________________________________________________*/
Send_Mail:
sa = substr(nsys,3,2)
isclog = 'CICSTS.ISCCICS.LOG.S'sa
say time() ' ISCCXØ1Ø - Allocation IscLog for 'ncics '...'
"Free file(iscl1Ø)"
"Alloc dataset('"isclog"') file(iscl1Ø) mod"
if rc ¬= Ø then Do
                 lparm = isclog
                 Call ISC_Alloc_Error
                 Return
                End
say time() ' ISCCXØ1Ø – Write IscLog for 'ncics '...'
  ADDRESS MVS
  "NEWSTACK"
  "execio * diskw iscl1Ø (finis stem rmail."
  if rc ¬= Ø then Do
                   Call ISC_Write_Error
                   Return
                  End
  "DELSTACK"
ADDRESS Netview
"Free file(iscl1Ø)"


tt = time()
tt = translate(tt,'',':')
tt = space(tt,Ø)
mm = substr(time(l),1Ø,6)
```

```
ADDRESS NETVIEW
  "ALLOC
DATASET('CICSTS.MAIL.ISCCICS.D"||DATE(J)||".T"||TT||".M"||mm||"')",
  "FILE(FMAIL) UNIT(WORKA) SPACE(1 1)",
  "DSORG(PS) LRECL(132) BLKSIZE(136) RECFM(VBA) NEW CATALOG"
ADDRESS MVS
  "NEWSTACK"
  QUEUE 'HELO JES2'
  QUEUE 'MAIL FROM:<NETOPER@Hostaname.dominio.it>'
  QUEUE 'RCPT TO:<CICSGROUP@dominio.it>'
  QUEUE 'DATA'
  QUEUE 'TO:<CICSGROUP@dominio.it>'
  QUEUE 'Date:'
  QUEUE 'Subject: ISC CICS control - 'ncics
  QUEUE ' '
  QUEUE ' ———————————————————————'
  QUEUE ' E-mail: NETOPER@Hostname.dominio.it'
  QUEUE ' Data: 'date()
  QUEUE ' Ora:  'time()
  QUEUE ' ———————————————————————'
  QUEUE ' '
    do x = 1 to rmail.Ø
        QUEUE ' 'rmail.x
    end
  QUEUE
  "EXECIO * DISKW FMAIL (FINIS)"
  "DELSTACK"
  ADDRESS NETVIEW
  "FREE FILE(FMAIL)"
End
filesend = 'TEMP.MAILT.D'DATE(J)'.T'TT'.M'mm
  ADDRESS NETVIEW
  "ALLOC DATASET('"filesend"')",
  "FILE(FSEND) UNIT(WORKA) SPACE(1 1)",
  "DSORG(PS) LRECL(8Ø) BLKSIZE(8Ø) RECFM(FB) NEW CATALOG"

filemail = 'CICSTS.MAIL.ISCCICS.D'DATE(J)'.T'TT'.M'mm
  ADDRESS MVS
  "NEWSTACK"
  QUEUE '//SENDMAIL  JOB (accnt),'
  QUEUE '//         CLASS=T,REGION=4M,'
  QUEUE '//         MSGCLASS=Z,'
  QUEUE '//         MSGLEVEL=(1,1)'
  QUEUE '//STEP1    EXEC PGM=IKJEFTØ1'
  QUEUE '//SYSTSPRT     DD SYSOUT=*'
  QUEUE '//SYSTSIN      DD DDNAME=SYSIN'
  QUEUE '//SYSIN        DD *'
  QUEUE "  XMIT JES2.SMTPxx DA('"filemail"')"
  QUEUE '/*'
  QUEUE '/*'
  QUEUE
  "EXECIO * DISKW FSEND (FINIS)"
  ADDRESS NETVIEW
  "SUBMIT '"filesend"'"
  ADDRESS MVS
  "DELSTACK"
ADDRESS NETVIEW
"FREE FILE(FSEND) DATASET('"filesend"') DELETE"
say time() ' ISC CICS control for 'ncics'. Send e-mail to CICS group …'
Return
```

ISCRESP

```
/*   ISCRESP
     Queued messages to command
*/

  Parse upper arg B1 'C=' CMD 'W=' WAITTIME ' ' B2
     cmd = strip(cmd)
     waittime = strip(waittime)
     if waittime = '' then waittime = 1
     'TRAP AND SUPPRESS MESSAGES *'
     Iinterpret "cmd"
     'wait 'waittime' seconds'
     'TRAP NO MESSAGES'
     'MSGREAD'
     Do until rc = 4
        'GETMSIZE NLINEE'
        Do i = 1 to nlinee
          'GETMLINE riga ' i
          QUEUE(riga)
          End
        'MSGREAD'
        End
     Return QUEUED()
     EXIT
```

ISCCX020

```
/*   ISCCXØ2Ø
     Delete timer for the control of the CICS connections.
     Clist executed to every closing of the cics.
*/
Trace ?o
Arg ncics
tcics = strip(substr(ncics,3,6))
idtimerb = 'B'||tcics
```

```
say '>>> ISCCXØ2Ø – ISC control delete timer for 'ncics
Interpret "'EXCMD AUTO2 PURGE TIMER="IDTIMERB",OP=AUTO2'"
Exit
```

NeETVIEW CUSTOMIZATION

In order to activate the timers in phase with the start of every CICS, add the following statements to the NetView automation table (DSITBLxx in the NetView DSIPARM library):

```
IF MSGID='IEF4Ø4I' & TOKEN(2) = JOB &
    TEXT = MESSAGE &
     ( TOKEN(2) = 'CIC' .    !
       TOKEN(2) = 'CX' . )
THEN EXEC(CMD('ISCCXØ2Ø 'JOB) ROUTE(ONE AUTO2))
    DISPLAY(N) BEEP(N) HOLD(N) NETLOG(Y) SYSLOG(Y);
```

In order to disable the timer in phase with the closing of every CICS, add the following statements to the NetView automation table (DSITBLxx in the NetView DSIPARM library):

```
IF MSGID  = 'DFHSI1517' & TEXT= . 'Control is being given to CICS' .
       & JOBNAME = JOB
  THEN EXEC(CMD('ISCCXØØØ 'JOB) ROUTE(ONE AUTO2))
       DISPLAY(Y) NETLOG(Y) SYSLOG(N) CONTINUE(Y);
```

SAMPLE EXCLUDE TABLE

```
*______________________________________________________________________*

         EXCLUDE table to control ISC CICS
*______________________________________________________________________*

|    The records in the first position with a character
|    not equal to "_"  are considered to be comments
|     and therefore it IGNORES them.
|
|    Every useful record must have following fields:
|
|       Record_Type     =  character "_" in position 1
|       Name_CICS        =  name-cics-proc
|       Name_Conid(n) =  ISC-name-connection
|
|       Example:
|Record_Type Name_CICS Name_Conid1 Name_Conid2 Name_Conid3 Name_Conid(n)
|----------- --------- ----------- ----------- ----------- ------------
```

```
|    _       CICSNAME   CON1       CON2        CON3       CON(n)
|
|  In order to exclude from the control all ISC connections of a
|    determined CICS, to set up
|  Name_Conid1 with the blank value or with '****' value.

*______________________________________________________________________*

_ Namecics1
_ Namecics2
_ Namecics3
_ Namecics4 con1 con2 con3 con4 con5 conn
_ Namecics5
_ Namecicsn ****
```

SAMPLE ISC LOG FILE

```
*______________________________________________________________________*
                Log CICS ISC connections control      -   subarea SØ82
                                           Init....19/Ø6/Ø2   ØØ:Ø1:Ø6
*______________________________________________________________________*

         __________ 19/Ø6/Ø2   ØØ:Ø8:38   SØ82___________  * CXIBCRM2 *

Modegroup SNASVCMG connection BB5Ø del CXIBCRM2 inactive
Modegroup LMODISC     connection BB5Ø del CXIBCRM2 inactive
         __________ 19/Ø6/Ø2   ØØ:2Ø:54   SØ82 __________  * CXIBDBS1 *
Modegroup SNASVCMG connection AGC1 del CXIBDBS1 inactive
Modegroup LMODISC     connection AGC1 del CXIBDBS1 inactive
Modegroup SNASVCMG connection BB5Ø  del CXIBDBS1 inactive
Modegroup LMODISC     connection BB5Ø  del CXIBDBS1 inactive
Modegroup SNASVCMG connection BT7Ø  del CXIBDBS1 inactive
Modegroup LMODISC     connection BT7Ø  del CXIBDBS1 inactive
Modegroup SNASVCMG connection CSDØ del CXIBDBS1 inactive
Modegroup LMODISC     connection CSDØ del CXIBDBS1 inactive
          __________ 19/Ø6/Ø2   15:26:14   SØ82 __________  * CICSTSP8 *
Modegroup SNASVCMG connection YK5Ø del CICSTSP8 inactive
Modegroup LMODISC     connection YK5Ø del CICSTSP8 inactive
         __________ 19/Ø6/Ø2   16:26:14   SØ82 __________  * CICSESE3 *
Modegroup SNASVCMG connection QAR1 del CICSESE3 inactive
Modegroup LU62MOD     connection QAR1 del CICSESE3 inactive
```

Espedito Morvillo
Systems Programmer (Italy)

CICS questions and answers

Q What does the SIT parm FSSTAFF actually do? I set it to YES, but don’t seem to get Alias termids.

A This option will cause the Terminal Auto-install program (AITM) to be driven in the AOR so that the AITM program can provide an Alias terminal ID. This is needed if you have two TORs that route transactions to the same AOR – the shipped definition could clash with an already-shipped definition from the other TOR. The old solution to this problem was to ensure that the two TORs used different TERMIDs. FSSTAFF allows you to get around this by driving AITM in the AOR to provide an Alias TERMID – the IBM-supplied default AITM program DFHZATDY needs the following lines commented in the section, Function 7 (Install shipped terminal definition) to make use of the default-supplied Alias:

```
*        L     R8,INSTALL_SHIPPED_TERMID_PTR
*        MVC   SELECTED_SHIPPED_TERMID,Ø(R8)
```

Q We run MRO CICS and use transaction routing. I’ve seen the IBM sample exit DFHXTENF in the sample library, but when and why would I need this?

A It’s used when an ATI request (EXEC CICS START) runs in an AOR against a terminal where the terminal definition has not yet been shipped (no transaction routing has been done yet). The exit program should provide the Netname or Sysidnt of the CICS region that holds the local definition. CICS can then ship the definition. You’ll need to have a method to figure out the TOR’s netname in the exit. If your application does not use EXEC CICS START against terminals before they have been shipped you do not need this exit.

Q Is there a way to protect my CICS region from being affected by the APPC connected ‘Server’ going slow or sticking and causing the CICS transactions to back-up/queue and eventually hang my system (sympathy sickness). TClass-ing has issues as we have many transids using the connection – we end up having all transids in the TClass.

A It’s possible to limit the queue by setting the number of sessions to what the connected ‘Server’ can cope with, and what limit you want queued in your CICS region. Then by using QUEUELIMIT and MAXQTIME (see CONNECTION definition) you can manage the queue/backlog. As an example: having a QUEUELIMIT of 0 and a MAXQTIME of 30 means that after all sessions are used any new transactions waiting for a session are subject to the 30 second timeout while they wait for a session to become free. This stops the CICS region from going MAXTASKS when the remote ‘Server’ has problems. It’s also possible to code a PROFILE on the EXEC CICS ALLOC command and timeout those transactions that have obtained a session but are waiting for a response from the remote ‘Server’.

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.

December 1999 – November 2002 index

Items below are references to articles that have appeared in *CICS Update* since Issue 169, December 1999. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

CICS news

ClientSoft has announced that it has added support for VSE, MQSeries, CTG, EXCI, and Microsoft COMTI, as well as enhanced Web services support, to ClientSoft Tanit Objects (CTO) Version 3.4.

This latest release includes a fully updated and enhanced version of Terminal XML (tXML), which was previously introduced in 1999, now called Terminal Web services (TWs). TWs are said to deliver 'instant WSDL' and 'instant XML' from CICS and IMS applications, without any development effort, while optimizing the XML/SOAP payload.

With the addition of TWs, the Tanit family of products now has the capability to provide simple/complex and non-reengineered/reengineered generation of XML and Web services from CICS and IMS applications, without reliance on traditional emulation-based host connectivity.

Any mainframe application capable of calling a COBOL sub-program can now be XML/SOAP enabled.

For further information contact:
ClientSoft, 8323 Northwest 12 Street, Suite 216, Miami, FL 33126, USA.
Tel: (305) 716 1007.
URL: http://www.clientsoft.com/products/tanit.htm.

* * *

CommerceQuest has introduced its CommerceQuest Suite for IBM WebSphere, which is said to act as an enabling engine to create additional connectivity to enhance WebSphere software capabilities, enable faster implementations, and complement existing systems.

This suite helps IBM and CommerceQuest sites minimize the integration efforts needed to integrate existing data and applications as XML Web services interfaces to support new WebSphere-powered applications. It will enable functional access and visibility to distributed disparate data and applications.

The suite comprises applications, tools, and expertise to support mainframe applications, MQ Integration middleware, and WebSphere. The specific elements include Rapid CICS Enabler for WebSphere, Rapid Web Services Enabler for WebSphere, Rapid Application Enabler for WebSphere, and Rapid Database Enabler for WebSphere.

Other tools and applications include Rapid File Transfer Enabler for WebSphere, Rapid MQ Enabler for WebSphere, Rapid MQ Integrator Enabler for WebSphere, and Rapid POS Enabler for WebSphere.

For further information contact:
CommerceQuest, 2202 N West Shore Blvd, Suite 600, Tampa, FL, 33607, USA.
Tel: (813) 639 6300.
URL: http://www.commercequest.com/press_release_detail.asp?id=285.

* * *

IBM has announced Version 3.1 of its Debug Tool for z/OS and OS/390, which helps examine, monitor, and control the execution of application programs by interactively debugging an application as it runs. It supports debugging of applications in environments including CICS.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/ad