November 2002

In this issue

3 CICS dynamic workload management – concepts
11 CICS and MQSeries: ReplyToQueue facility
18 CICS file control enhancements
24 Check start-up type for your production CICS
29 z/OS Version 1.4 announcement
30 Automatic control for the CICS ISC connections
44 CICS questions and answers
46 December 1999 – November 2002 index
48 CICS news

© Xophon plc 2002
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

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

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:

```plaintext
EXEC CICS RETURN NEXTTRANID(tranid) COMMAREA(state_data)
```

for the next segment.

```plaintext
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

![Figure 1: Affinity example](image)

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.
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) © IBM 2002

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

CRMQTE02 TITLE 'BATCH INTERFACE: TEST THE 3270 BRIDGE VIA MQSERIES'

**********************************************************************
The program is called via the batch.

A transaction (T000) is started via the 3270 bridge.

* 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 (T000) IS STARTED VIA THE 3270 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

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

***********************************************************************
*                  READ CONTROL RECORD FROM SYSIN                   *
***********************************************************************

READ     EQU   *
MVC   TRACE, =CL8'READ'
WTO   'CRMQTEØ2: READ'
OPEN  (SYSIN,(INPUT))
GET   SYSIN, IOAREA
CLC   IOAREA(4), =CL4'PROD'       PRODUCTION
BE    READ1000
CLC   IOAREA(4), =CL4'SYST'      SYSTEM
BE    READ1000
CLC   IOAREA(4), =CL4'TEST'     TEST
BE    READ1000
CLC   IOAREA(2), =CL2'AE'      TEST SPECIAL
BE    READ1000
CLC   IOAREA(4), =CL4'VPRD'    VPRD
BE    READ1000
B READ_ERROR

READ1000 EQU   *
MVC   QMGRNAME, BLANKS       USE DEFAULT QUEUE MANAGER
MVC   MQQUEUE+5(4), IOAREA   MQ QUEUE
MVC   MQQUEUER+5(4), IOAREA   MQ REPLY TO QUEUE
CLOSE  (SYSIN)
MVI   OPEN, C' Y'
WTOE  'SYSIN=', IOAREA
BR    SUBREG

READ_ERROR  EQU   *
WTOE  'INVALID PARAMETER SYSIN=', IOAREA
BR    SUBREG

SYSIN    DCB   DDNAME=MQTESTIN, DSORG=PS, MACRF=GM, EODAD=READ_ERROR

**CONNECT TO MQ MANAGER**

```
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**

```
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**

```
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 R0, 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 R0, R2 MVCL
MVC MYGMO_AREA, MQGMO_AREA
MVC FLENGTH, =AL4(QEND-QSTRT)
CALL MQGET, +
( HCONN, HOBJ, MYMD, MQGMO, FLENGTH, QUEUECO, FLENGTH, +
 COMPCODE, REASON), +
MF=(E, CALLLIST), VL
LA R0, MQCC_OK
C R0, 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 R0, 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 R0, R2 MVCL
LA R0, MQOO_OUTPUT
ST R0, OPTIONS
LA R0, MQOT_Q
ST R0, MYOD_OBJECTTYPE
MVC OBJECTNAME, MQUEUE
MVC MYOD_OBJECTNAME, OBJECTNAME
CALL MQOPEN, +
( HCONN, MYOD, OPTIONS, HOBJ, COMPCODE, REASON), +
MF=(E, CALLLIST), VL
LA R0, MQCC_OK
C R0, COMPCODE
BER SUBREG
WTO 'CRMQTE02: ERROR DURING MQOPEN!'
BR SUBREG
EJECT
***********************************************************************
* PUT A MESSAGE ON A QUEUE                                       *
***********************************************************************
SPACE
MQPUT EQU *
MVC TRACE,=CL8'MQPUT'
WTO 'CRMQTEØ2: MQPUT'
LA R0, 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 R0, R2    MVCL
*
MVC MYPMO_AREA, MQPMO_AREA
*
MVC FLENGTH, =AL4(QEND-QSTRT)
*
MVC MYMD_CORRELID, MQCI_NEW_SESSION
MVC MYMD_FORMAT, =CL8'MQCI CS'
MVC MYMD_REPLYTOQ, MQUEUE
*
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 R0, MQCC_OK
C R0, 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'
MQUEUE 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 DESCRIPTE
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

PARM     DS    F
LOADADDR DC    A(0)

***********************************************************************
*        PROGRAM CONSTANTS                                            *
***********************************************************************

SPACE

BLANKS         DC   256C' '
NULLS          DC   Ø16X'00'
ZEROS          DC   Ø16C'0'

DATE           DS   CL16
LENGTH         DS   H
LENGTH         DS   F
TRACE          DS   CL8
WORK           DS   CL16
OPEN           DS   CL1

MQCI_NEW_SESSION  DC
X'414D51214E45575F5345534944F4E5F434F5252454C4944X

* DS   0D
EYEC1          DC   CL8'**COCO**'
COMPCODE       DS   F               COMPLETION CODE
REASON         DS   F               REASON CODE QUALIFYING COMPCODE
HCONN          DS   F               CONNECTION HANDLE
OBJ            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'T000'         CICS TRANSACTION
CALLLIST      CALL , (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 CRMQTE02'
QEND     DS   ØC

SPACE
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 (e.g., 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 – i.e., 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 redispaching 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) © Xephon 2002
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 XXXXXXXXX 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 **'
*
DFHEI STG DSECT

***********************************************************************
* Message declarations.
***********************************************************************
  MSG_RESP   DS  F                   Response code from EXEC CICS
  MSG_BUFFER DS  OCL98               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  CL8ØØ               Message text
  *
    ORG   MSG_TEXT
  END_MSG_AUTO DS   OCL8Ø
               DS   CL16
  STARTUP_TYPE DS   CL1ØØ
                 DS   CL54
  *
    ORG   MSG_TEXT
  ERROR_MSG_AUTO DS  OCL8Ø
                  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 message type
STARTUP_TPE   DS  CL10     Start 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           Unknown startup type                      *
*
26 © 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
Cold Start: initial or cold.

Cold Start: initial or cold.

LBCLDSTR CLC COLDSTATUS, DFHVALUE(INITIAL)
  BE LBN003 Initial start
BE LBN001 Cold start
B LBE004 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.

LBN001 MVC STARTUP_TPE, =CL10'Cold'
B LBNMSG
LBN002 MVC STARTUP_TPE, =CL10'Warm'

B LBNMSG

LBN003 MVC STARTUP_TPE,=CL10'Initial.' B LBNMSG

LBN004 MVC STARTUP_TPE,=CL10'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.
***********************************************************************
LBE001 MVC ERROR_TXT,=CL46'Assign Failure.'
   B LBEMSG
LBE002 MVC ERROR_TXT,=CL46'Inquire System Failure.'
   B LBEMSG
LBE003 MVC ERROR_TXT,=CL46'Unknown Startup Type!'
   B LBEMSG
LBE004 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 CL02'TD' Terminal related task
NOTTERMINAL DC CL02'S' Nonterminal related task
LTORG
*
***********************************************************************
* Messages.
***********************************************************************
END_MSG DC ØCL8Ø'
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.

---

Why not share your expertise and earn money at the same time? _CICS Update_ is looking for technical articles and hints and tips that experienced CICS users have written to make their life, or the lives of their users, easier. We would also be interested in articles about performance and tuning of CICS. Articles can also be e-mailed to the editor at trevore@xephon.com.
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

/* ISCCX000
   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
  else iterate
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,ISCCX010" NCICS''"
Interpret "'EXCMD AUTO2 EVERY Ø1:Ø0:Ø0,ID="IDTIMERB",SAVE,ISCCX010" NCICS''"
say '>>> ISCCX000 - ISC control scheduled for 'ncics
Return
ISC_Alloc_Error:
/* _________________________________________________________________
Routine of management errors
_________________________________________________________________ */
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCX000 - CICS ISC control.                      ***'
say time() '***                Allocation error for the file:     ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                   ***'
say time() '*********************************************************'
"Free file(isc00)"
Return
ISC_Read_Error:
say time() '*********************************************************'
say time() '***                                                   ***'
say time() '*** ISCCX000 - CICS ISC control.                      ***'
say time() '***              Read error for the table             ***'
say time() '***          'lparm'                                  ***'
say time() '***                                                   ***'
say time() '*********************************************************'
"Free file(isc00)"
Return

ISCCX010
/* ISCCX010
   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 = mvsvart(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(isc.r.i)
Do e = 3 to nconid
  x = e-2
  econid.x = word(isc.r.i, e)
End

econid.Ø = x
Return

ISC_Alloc_Error:

/* Routine of management errors */

"Free file(isc10)"
"Free file(isc110)"

say time() '******************************************************
say time() '*** ISCCXØ1Ø - Aoc: control ISC. ***
say time() '*** Error in the allocation of the file: ***
say time() '*** 'lparm' ***
say time() '******************************************************

Return

ISC_Read_Error:

Address Netview

"Free file(isc10)"

say time() '******************************************************
say time() '*** ISCCXØ1Ø - Aoc: control ISC. ***
say time() '*** Error in phase of reading of the chart: ***
say time() '*** 'lparm' ***
say time() '******************************************************

Return

ISC_Write_Error:
Address Netview
"Free file(iscl10)"
say time() '*********************************************************'
say time() '***
*** ISCCX010 - Aoc: control ISC.
***
*** Error in phase writing to the log:
***
*** 'lparm'
***
***
*********************************************************

Return
ISC_Control_all:
/
*-----------------------------------------------

Complete control. It carries out the control
of all the ISC connections.

Address Netview
"Free file(iscl10)"
say time() '*********************************************************'
say time() '***
*** ISCCX010 - ISC control.
***
*** Verify all CICS ISC connections
***
*** for 'ncics'.
***
*********************************************************

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 10 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 10 SECONDS

NRIGHE = ISCRESP("C=MVS F "ncics","CEMT I MODE AV(000) 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 = '000' 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 = 0 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.0 = recem
Call Send_Mail
Return

ISC_Control_no:
/* CICS in exclude list. No ISC connection control */
Address Netview
"Free file(isc10)"
say time() '**************************************************'
say time() '***
say time() '*** ISCXX010 - ISC control.  ***
say time() '*** Exclude list active.  ***
say time() '*** No control to 'ncics'.  ***
say time() '***
Return

ISC_Control_parz:
/* Some connections in exclude file, carry out partial control. */
Address Netview
Free file(isc10)

say time() '************************************************************
say time() '*** ISCCX010 - ISC control. ***
say time() '*** Partial control for connections ***
say time() '*** of the 'ncics'. ***
say time() '************************************************************

trace ?

NRIGHE = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(000)' 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 = '000' & econid.k ¬= conid THEN sw = 0
    else do
      sw = 1
      leave
    end
End

If sw = 0 then DO
  CONIDOK = KO
  "MVS F "ncics",'CEMT SET CONN("CONID") INS REL'"
  WAIT 10 SECONDS
  "MVS F "ncics",'CEMT SET CONN("CONID") INS ACQ'"
END
else sw = 0

END

IF CONIDOK = OK THEN Return
"WAIT 10 SECONDS"
NRIGHE = ISCRESP("C=MVS F "ncics",'CEMT I MODE AV(000)' 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 = '000' & econid.k ¬= conid THEN sw = 0
    else do
      sw = 1
      leave
    end
End

If sw = 0 & conid ¬= '' then DO
  msg='Modegroup 'MODENAME' connection 'CONID' of the 'ncics
  msg=msg||' inactive. To Verify |||
  say msg
if recem = 0 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.'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(1),10,6)
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@Hostname.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)"
"DELSSTACK"
ADDRESS NETVIEW
"FREE FILE(FMAIL)"
End
filesend = 'TEMP.MAILT.DATE(J).T.TT.M' mm
ADDRESS NETVIEW
"ALLOC DATASET('filesend')",
"FILE(FSEND) UNIT(WORKA) SPACE(1 1),
"DSORG(PS) LRECL(80) BLKSIZE(80) RECFM(FB) NEW CATALOG"

filemail = 'CICSTS.MAIL.ISCCICS.D' DATE(J) '.T.TT.M' mm
ADDRESS MVS
"NEWSTACK"
QUEUE '//'SENDMAIL JOB (acct),'
QUEUE '//' CLASS=T, REGION=4M,'
QUEUE '//' MSGCLASS=Z,'
QUEUE '//' MSGLEVEL=(1,1)'
QUEUE '//'STEP1 EXEC PGM=IKJEFT01'
QUEUE '//'SYSTSRT DD SYSOUT=''
QUEUE '//'SYSTSSIN DD DDNAME=SYSIN'
QUEUE '//'SYSSIN DD *'
QUEUE "XMIT JES2.SMTPxx DA('filemail')"
QUEUE '*'

© 2002. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
* QUEUE '/*'
QUEUE
"EXECIO * DISKW FSEND (FINIS)"
ADDRESS NETVIEW
"SUBMIT 'filesend'"
ADDRESS MVS
"DELSSTACK"
ADDRESS NETVIEW
"FREE FILE(FSEND) DATASET('filename') 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 *'
  linterp "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

/* ISCCX020
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
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):

```plaintext
IF MSGID='IEF4041' & TOKEN(2) = JOB & TEXT = MESSAGE &
    ( TOKEN(2) = 'CIC' , !
    TOKEN(2) = 'CX' . )
THEN EXEC(CMD('ISCCX020 '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):

```plaintext
IF MSGID = 'DFHSI1517' & TEXT= . 'Control is being given to CICS' . & JOBNAME = JOB
THEN EXEC(CMD('ISCCX000 '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) |
|-------------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
```

© 2002, Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
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.

```
  _       CICSNAME   CON1       CON2        CON3        CON(n)
```

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

**SAMPLE ISC LOG FILE**

```
*------------------------------------------------------------------*
| Log CICS ISC connections control      -   subarea S082 |
| Init....19/06/02   00:01:06           |
*------------------------------------------------------------------*

```

```
19/06/02   00:08:38   S082_________  * CXIBCRM2 *
```

```
Modegroup SNASVCMG connection BB50 del CXIBCRM2 inactive
Modegroup LMODISC connection BB50 del CXIBCRM2 inactive
    19/06/02   00:20:54   S082_________  * CXIBDBS1 *
```

```
Modegroup SNASVCMG connection AGC1 del CXIBDBS1 inactive
Modegroup LMODISC connection AGC1 del CXIBDBS1 inactive
Modegroup SNASVCMG connection BB50 del CXIBDBS1 inactive
Modegroup LMODISC connection BB50 del CXIBDBS1 inactive
Modegroup SNASVCMG connection BT70 del CXIBDBS1 inactive
Modegroup LMODISC connection BT70 del CXIBDBS1 inactive
Modegroup SNASVCMG connection CSD0 del CXIBDBS1 inactive
Modegroup LMODISC connection CSD0 del CXIBDBS1 inactive
    19/06/02   15:26:14   S082_________  * CICSTSP8 *
```

```
Modegroup SNASVCMG connection YK50 del CICSTSP8 inactive
Modegroup LMODISC connection YK50 del CICSTSP8 inactive
    19/06/02   16:26:14   S082_________  * CICSESE3 *
```

```
Modegroup SNASVCMG connection QAR1 del CICSESE3 inactive
Modegroup LU62MOD connection QAR1 del CICSESE3 inactive
```

---

Espedito Morvillo  
Systems Programmer (Italy)  © Xephon 2002

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.

© Xephon 2002

Leaving? You don’t have to give up CICS Update

You don’t have to lose your subscription when you move to another location – let us know your new address, and the name of your successor at your current address, and we will send CICS Update to both of you, for the duration of your subscription. There is no charge for the additional copies.
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.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Issue</th>
<th>Page(s)</th>
<th>Issue</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3270 bridge</td>
<td>169</td>
<td>203.37-44</td>
<td>169</td>
<td>196.20-41, 197.31-51</td>
</tr>
<tr>
<td>Affinities</td>
<td>177</td>
<td>32-47</td>
<td>174</td>
<td>191.28-47</td>
</tr>
<tr>
<td>Alter</td>
<td>178</td>
<td>5-23</td>
<td>175</td>
<td>201.24-</td>
</tr>
<tr>
<td>BMS</td>
<td>188</td>
<td>46-51</td>
<td>187</td>
<td>175.15-31</td>
</tr>
<tr>
<td>Autostart</td>
<td>172</td>
<td>3-10</td>
<td>177</td>
<td>200.3-7</td>
</tr>
<tr>
<td>Batch</td>
<td>198</td>
<td>27-42</td>
<td>171</td>
<td>183.25-35</td>
</tr>
<tr>
<td>Case</td>
<td>203</td>
<td>44-47</td>
<td>173</td>
<td>199.11-26</td>
</tr>
<tr>
<td>CEDA</td>
<td>190</td>
<td>11-12</td>
<td>192</td>
<td>204.21-30</td>
</tr>
<tr>
<td>CEDF</td>
<td>172</td>
<td>3-9</td>
<td>172</td>
<td>181.32-47</td>
</tr>
<tr>
<td>CETR</td>
<td>174</td>
<td>3-14</td>
<td>171</td>
<td>191.3-7</td>
</tr>
<tr>
<td>Chargeback</td>
<td>202</td>
<td>18-39</td>
<td>203</td>
<td>203.14-19</td>
</tr>
<tr>
<td>CICS TS 1.3</td>
<td>188</td>
<td>3-4</td>
<td>192</td>
<td>178.3-4</td>
</tr>
<tr>
<td>CICS TS 2.2</td>
<td>197</td>
<td>3-11</td>
<td>174</td>
<td>196.42</td>
</tr>
<tr>
<td>CICSplex SM API</td>
<td>170</td>
<td>30-47</td>
<td>171</td>
<td>189.25-35</td>
</tr>
<tr>
<td>CICSplex SM</td>
<td>194</td>
<td>19-28</td>
<td>193</td>
<td>204.8-13</td>
</tr>
<tr>
<td>Cloning</td>
<td>170</td>
<td>12-25</td>
<td>174</td>
<td>178.23-32</td>
</tr>
<tr>
<td>Closing files</td>
<td>203</td>
<td>3-14</td>
<td>174</td>
<td>181.14-28</td>
</tr>
<tr>
<td>Cold start</td>
<td>169</td>
<td>16-20</td>
<td>182</td>
<td>182.28-35</td>
</tr>
<tr>
<td>Compiler</td>
<td>200</td>
<td>22-30</td>
<td>192</td>
<td>197.13-17</td>
</tr>
<tr>
<td>COPY</td>
<td>181</td>
<td>28-31</td>
<td>186</td>
<td>202.7-18</td>
</tr>
<tr>
<td>CPSM</td>
<td>192</td>
<td>8-10</td>
<td>194</td>
<td>193.9-22</td>
</tr>
<tr>
<td>CREATE command</td>
<td>173</td>
<td>3-10</td>
<td>182</td>
<td>189.3-7</td>
</tr>
<tr>
<td>Cryptographic co-processor</td>
<td>197</td>
<td>13-17</td>
<td>180</td>
<td>201.6-11</td>
</tr>
<tr>
<td>CSD</td>
<td>169</td>
<td>3-16</td>
<td>171</td>
<td>182.17-28</td>
</tr>
<tr>
<td>CSD</td>
<td>172</td>
<td>29-47</td>
<td>173</td>
<td>204.30-44</td>
</tr>
<tr>
<td>CSD</td>
<td>174</td>
<td>30-47</td>
<td>175</td>
<td>181.3-13</td>
</tr>
<tr>
<td>CSD</td>
<td>176</td>
<td>25-44</td>
<td>182</td>
<td>193.3-9</td>
</tr>
<tr>
<td>CSD</td>
<td>183</td>
<td>38-47</td>
<td>183</td>
<td>187.40-45</td>
</tr>
<tr>
<td>CSD</td>
<td>186</td>
<td>32-44</td>
<td>185</td>
<td>201.3-5</td>
</tr>
<tr>
<td>CVDA</td>
<td>185</td>
<td>3-11</td>
<td>186</td>
<td>190.16-24</td>
</tr>
<tr>
<td>CWA</td>
<td>197</td>
<td>18-30, 199.7-12</td>
<td>171</td>
<td>171.13-25</td>
</tr>
<tr>
<td>DB2</td>
<td>182</td>
<td>35-47</td>
<td>182</td>
<td>180.6-9</td>
</tr>
<tr>
<td>Topic</td>
<td>Page Ranges</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>193.3-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maps</td>
<td>185.32-47, 202.3-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messages</td>
<td>186.9-22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MQSeries</td>
<td>153.15-28, 154.10-21, 203.37-44, 204.14-21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MRO</td>
<td>182.17-28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name Counter Service</td>
<td>195.14-26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>National language</td>
<td>169.37-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEWCOPY</td>
<td>169.21-36, 173.14-29, 174.21-29, 175.14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Null-use resources</td>
<td>174.30-47, 175.32-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opening files</td>
<td>203.3-14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTE</td>
<td>201.40-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCL</td>
<td>196.10-20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td>195.12-13, 198.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL/I</td>
<td>189.3-7, 189.7-24, 194.10-18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL/I transactions</td>
<td>187.3-6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printer assignment</td>
<td>183.39-47, 184.11-36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printing</td>
<td>194.29-43, 195.26-33, 196.10-20, 202.3-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming</td>
<td>195.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Questions and answers</td>
<td>195.41-43, 196.43, 197.51, 198.43, 199.50-51, 201.47, 202.47, 203.47, 204.44-45, 203.14-19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QMF</td>
<td>197.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDO</td>
<td>169.3-16, 170.26-30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDO definition</td>
<td>188.5-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reactivating</td>
<td>201.24-40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refresh programs</td>
<td>200.42-51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rununits</td>
<td>178.3-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Session reuse</td>
<td>200.3-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared TS queues</td>
<td>175.3-14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIGNON</td>
<td>197.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIT parameter</td>
<td>184.3-5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMP/E zones</td>
<td>171.3-12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMTP</td>
<td>176.39-47, 177.9-16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Started regions</td>
<td>176.3-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start-up type</td>
<td>204.3-8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Statistics</td>
<td>199.12-40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage</td>
<td>199.3-7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage management</td>
<td>178.3-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage use</td>
<td>186.23-32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stress</td>
<td>182.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary storage</td>
<td>179.3-5, 191.8-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temporary storage queues</td>
<td>192.11-24, 194.3-9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminals</td>
<td>186.33-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testing</td>
<td>189.7-24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Monitor</td>
<td>178.33-47, 179.17-34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time Machine</td>
<td>197.18-30, 199.7-12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRACE TABLE</td>
<td>174.3-14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction information</td>
<td>201.11-24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transaction Server 1.3</td>
<td>170.3-11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSO file</td>
<td>194.29-43, 195.26-33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuning</td>
<td>174.14-20, 191.28-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VB6</td>
<td>194.10-18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSAM</td>
<td>175.15-31, 181.14-28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VSE</td>
<td>181.32-47</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web</td>
<td>180.38-45, 195.34-40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web documents</td>
<td>189.36-51</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web interface</td>
<td>185.11-12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web User Interface (WUI)</td>
<td>191.12-21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebSphere</td>
<td>193.9-22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WML</td>
<td>179.6-16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XMEOUT</td>
<td>198.3-10, 202.7-18</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

* * *

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.

* * *

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