September 1999

In this issue

3   A shutdown assist and task purge utility
10  Interpreting temporary storage behaviour
19  CEMT Logger – an alternative design – part 2
28  Displaying CPU usage by TCB – revisited
29  Accessing CICS control blocks in COBOL
42  Listing the TMONCICS control file RLC definitions
48  CICS news
Contributions
Articles published in *CICS Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

*CICS Update* on-line
Code from *CICS Update* can be downloaded from our Web site at http://www.xephon.com/cicsupdate.html; you will need the user-id shown on your address label.

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.

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 January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

© Xephon plc 1999. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

*Printed in England.*
INTRODUCTION

This article presents a program to purge CICS tasks. It functions either as a shutdown assist program or as a utility to cancel tasks by transaction-id and/or user-id. To use as a shutdown assist, you simply add the program to the PLTSD (for CICS Releases 4.1 and below) or define an associated transaction (eg KILL) and specify it as SDTRAN (for Transaction Server releases). To use in utility mode, you must invoke the program as transaction KILL from a console or terminal and supply the appropriate terminal input.

Here are some console examples:

- ‘F PRODCICS,KILL TRAN=UPDT’ cancels tasks with the transaction-id UPDT.
- ‘F PRODCICS,KILL USER=MRHAPPY’ cancels tasks with the user-id MRHAPPY.
- ‘F PRODCICS,KILL TRAN=UPDT,USER=MRHAPPY’ cancels tasks meeting both criteria.

Operators and systems programmers favour this utility over CEMT to cancel tasks because they are freed from the burden of remembering task numbers and because they can cancel more than one task at a time. The ability to cancel multiple tasks having the same transaction-id is especially useful during deadlock situations.

KILLTASK

*ASM XOPTS(CICS,FE,SP)
*
* PROGRAM: KILLTASK
*
* PURPOSE: Purge CICS tasks
*      PRINT ON,Nogen
*
      DFHREGS
* Register Usage:
* 
* R2 -> task count
* R3 code base
* R4 -> task list
* R5 -> tran list corresponding
* R6 -> EIB
* R8 -> TIOA start 3
* R9 -> TIOA offset 3 optional
* RB -> TIOA end 3
* RD -> dynamic storage
* 
DFHEISTG DSECT
CVDA       DS    F
RESP       DS    F
TARGET     DS    PL4
FACILITY   DS    CL4
FACTYPE    DS    F
TASKCNT    DS    F
LOGTEXT    DS    0H
LOGPART1   DS    CL32
LOGTASK    DS    CL07
LOGPART2   DS    CL07
LOGTRAN    DS    CL04
LOGPART3   DS    CL07
LOGTERM    DS    CL04
LOGPART4   DS    CL07
LOGUSER    DS    CL08
LOGPART5   DS    CL14
LOGTYPE    DS    CL05
LOGTEXTL   EQU   *-LOGTEXT
LOGL       DS    H
TIOALEN    DS    H
TRANPARM   DS    CL4
USERPARM   DS    CL8
HOWSTART   DS    CL2
KILLFLAG   DS    C
PARMFLAG   DS    C
* 
KILLTASK DFHEIENT CODEREG=(3),DATAREG=(13),EIBREG=(6)
KILLTASK AMODE 31
KILLTASK RMODE ANY
* 
EXEC  CICS HANDLE CONDITION ERROR(DUMP)
* 
BAL   R7,LIST                     get list of tasks
L     R2,TASKCNT
LTR   R2,R2
BZ    RETURN
XC PARMFLAG,PARMFLAG  initialize to no input parms
CLC EIBTRNID,=C'KILL'
BNE INITIAL
*
EXEC CICS ASSIGN STARTCODE(HOWSTART)
*
CLC HOWSTART,=C'TD'  possible terminal input?
BNE INITIAL  n - skip receive
*
EXEC CICS RECEIVE SET(R8) +
LENGTH(TIOALEN)
*
LH R11,TIOALEN
C R11,=F'10'
BL INITIAL  n - skip parse
LA R10,1
AR R11,R8
S R11,=F'6'
LA R8,4(R8)
LA R9,4
GETPARM DS ØH
TM PARMFLAG,X'03'
BC 3,INITIAL  y - we're done parsing
TM PARMFLAG,X'01'
BC 3,GETPARM1  y - skip parse for TRAN=
CLC Ø(5,R8),=C'TRAN= '
BNE GETPARM1
LA R8,5(R8)
LA R9,5(R9)
BH LOADTRAN
LA R12,TIOALEN
SR R12,R9
BCTR R12,R0
C R12,=F'3'
BNH LOADTRAN
LA R12,3
LOADTRAN LA R7,TRANPARM
EX R12,TRANMVC
TR TRANPARM,XLTAB  blank any valid delimiter
LA R12,3
PADTRAN CLI Ø(R7),X'40'
BNE PADTRAN1
MVI 1(R7),X'40'
PADTRAN1 LA R7,1(R7)
BCT R12,PADTRAN
OI PARMFLAG,X'01'
B GETPARM1
*
TRANMVC MVC Ø(1,R7),Ø(R8)
*
GETPARM1 TM PARMFLAG,X'02'  user parm already loaded?
BC 3,GETPARM2  \( y \) - skip parse for USER=
CLC Ø(5,R8),=C'USER='
BNE GETPARM2
LA R8,5(R8)  \( \) skip over USER=
LA R9,5(R9)  \( \) bump tioa offset accordingly
LH R12,TIOALEN
SR R12,R9
BCTR R12,R0
C R12,=F'7'
BNH LOADUSER
LA R12,7
LOADUSER LA R7,USERPARAM
EX R12,USERMVC
TR USERPARAM,XLTAB  \( \) blank any valid delimiter
LA R12,7
PADUSER CLI Ø(R7),X'40'
BNE PADUSER1
MVI 1(R7),X'40'
PADUSER1 LA R7,1(R7)
BCT R12,PADUSER
OI PARMFLAG,X'02'  \( \) signal user parm input
GETPARM2 LA R9,1(R9)  \( \) increment tioa offset
BXLE R8,R10,GETPARM
B INITIAL  \( \) end parse

* USERMVC MVC Ø(1,R7),Ø(R8)
*
INITIAL DS ØH
XC KILLFLAG,KILLFLAG  \( \) initialize kill flag
MVC LOGPART1,MSGPART1  \( \) set up CSMT log msg
MVC LOGPART2,MSGPART2
MVC LOGPART3,MSGPART3
MVC LOGPART4,MSGPART4
MVC LOGPART5,MSGPART5
LA R7,LOGTEXTL  \( \) get length for TD write
STH R7,LOGL
MVC CVDA,DFHVALUE(PURGE)  \( \) set up task purge
MVC LOGTYPE,PURGE
BAL R7,KILL  \( \) do first level cancels
CLI KILLFLAG,X'FF'  \( \) were there any?
BNE RETURN  \( \) n - then we're done
EXEC CICS DELAY FOR SECONDS(3)
BAL R7,LIST  \( \) see what tasks remain
L R2,TASKCNT
LTR R2,R2
BZ RETURN
MVC CVDA,DFHVALUE(FORCEPURGE)  \( \) we're done kidding around
MVC LOGTYPE,FORCE
BAL R7,KILL  \( \) do second level cancels
B RETURN
* LIST DS ØH

* EXEC CICS INQUIRE TASK LIST +
  LISTSIZE(TASKCNT) +
  SET(R4) +
  SETTRANSID(R5)

* BR R7

* KILL DS ØH
  ZAP TARGET,Ø(4,R4) get taskno from list
  CP TARGET,EIBTASKN no suicides
  BE KILLNEXT

* EXEC CICS INQUIRE +
  TASK(TARGET) +
  FACILITY(FACILITY) +
  FACILITYTYPE(FACTYPE) +
  USERID(LOGUSER)

* CLI PARMFLAG,X'ØØ'
  BE KILLØØ
  CLI PARMFLAG,X'Ø1'
  BE KILLØ1
  CLI PARMFLAG,X'Ø2'
  BE KILLØ2
  CLI PARMFLAG,X'Ø3'
  BE KILLØ3
  B RETURN illogic - shouldn't get here

  KILLØØ LA R8,EXCLTAB check exclude table
  LA R1Ø,4
  LR R11,R8
  LA R11,EXCLLEN(R11)
  BCTR R11,RØ

  KILLØØX CLC Ø(4,R8),Ø(R5) task's tran in exclude list?
  BE KILLØØX
  BXLE R8,R1Ø,KILLØØX y - then bypass cancel
  B KILLTSK

  KILLØ1 CLC TRANPARM,Ø(R5)
  BE KILLTSK
  B KILLNEXT

  KILLØ2 CLC USERPARM,LOGUSER
  BE KILLTSK
  B KILLNEXT

  KILLØ3 CLC TRANPARM,Ø(R5)
  BNE KILLNEXT
  CLC USERPARM,LOGUSER
  BE KILLTSK
  B KILLNEXT
KILLTSK MVI KILLFLAG,X'FF' show that we fired
*
EXEC CICS SET +
  TASK(TARGET) +
  PURGETYPE(CVDA) +
  RESP(RESP)
*
CLC RESP,DFHRESP(NORMAL) BNE KILLNEXT
MVC LOGTERM,=4X'40' clear termid
CLC FACTYPE,DFHVALUE(TERM) is facility a terminal?
BNE KILLLOG n - don't log facility
MVC LOGTERM,FACILITY
*
KILLLOG UNPK LOGTASK,TARGET format log msg
OI LOGTASK+6,X'F0'
MVC LOGTRAN,Ø(R5)
*
EXEC CICS WRITEQ TD +
  QUEUE('CSMT') +
  FROM(LOGTEXT) +
  LENGTH(LOGL)
*
KILLNEXT LA R4,4(R4) bump to next task in list
LA R5,4(R5) next tran in list
BCT R2,KILL repeat kill for each task
BR R7
*
DUMP DS ØH EXEC CICS DUMP TASK DUMPCODE(EIBTRNID)
B RETURN
*
RETURN DS ØH EXEC CICS RETURN
*
MSGPART1 DC CL32'KILLTASK issued cancel for task '
MSGPART2 DC CL07', tran '
MSGPART3 DC CL07', term '
MSGPART4 DC CL07', user '
MSGPART5 DC CL14', purgetype = '
*
PURGE DC CL05'PURGE'
FORCE DC CL05'FORCE'
*
EXCLTAB DS ØH Tran Exclude Table
DC CL4'AAON' Abend-Aid
DC CL4'DSNC' DB2
DC CL4'OMEG' Omegamon
EXCLLEN EQU *-EXCLTAB
*
PROGRAM NOTES

There are a few features of KILLTASK that you may want to know about before using it at your site. First-level cancels are issued in the form of a task PURGE; only if tasks remain after a three-second interval are second-level cancels issued in the form of a task FORCE.

Any transactions you want to make ineligible for cancellation should be placed in the transaction exclude table. Our table excludes the DB2 connection, Abend-Aid/FX, and Omegamon because these products have their own entries in the PLTSD, and it is generally a good idea to exclude non-application transactions. Logging to transient data queue CSMT is performed for any tasks purged.

Finally, a good deal of freedom is allowed with respect to utility mode terminal input. Parameters ‘USER=’ and ‘TRAN=’ may be entered with either one first. Spacing and use of delimiters is free-form.

Russell Hunt
CICS Systems Programmer (USA)
Interpreting temporary storage behaviour

INTRODUCTION

The temporary storage component of CICS has been rewritten and restructured into a CICS Domain in CICS Transaction Server Release 1. The old DFHTSP code of CICS/ESA Release 4.1.0 and below has been replaced with a new suite of programs, written using object-oriented techniques. This enhancement has brought with it the benefits of structural design, stability, and maintainability that the Domain model provides to CICS.

With the rewrite of temporary storage for CICS Transaction Server, a number of queries against the restructured code have been reported to the CICS change team. This article addresses these observations, and explains their background and how they should be interpreted.

CICS TRANSACTION SERVER

This article makes reference to CICS Transaction Server for OS/390 Releases 1 and 2. CICS Transaction Server is a member of the OS/390 family of MVS-based software servers. IBM has integrated CICS with a set of supporting software, offering a single product in their place.

The CICS component of CICS Transaction Server Release 1 has a Release number of 0510. The CICS component of CICS Transaction Server Release 2 has a Release number of 0520. There is no separately orderable product such as ‘CICS/ESA 5.1.0’, however – it is the CICS component of CICS Transaction Server Release 1. IBM has recently shipped CICS Transaction Server Release 3. The CICS component of CICS Transaction Server Release 3 has a Release number of 0530.

BACKGROUND TO CICS TEMPORARY STORAGE MANAGEMENT

The basic concepts of temporary storage control under CICS are the same in CICS Transaction Server as they were for CICS/ESA Release 4.1.0 and prior releases. An API is provided for CICS applications to
write to, read from, and delete record data held on temporary storage queues. The queue name is a unique identifier on a given CICS system that represents the collection of records held upon the queue.

Using the EXEC CICS WRITEQ, READQ, and DELETEQ API commands, applications can store, retrieve, and delete temporary storage queues. The data on each queue can be held on one of several media. Main temporary storage resides in-core within the CICS address space. Auxiliary temporary storage is maintained in a number of Control Intervals residing on a CICS-maintained VSAM ESDS, DFHTEMP. Finally, shared temporary storage (introduced in CICS Transaction Server Release 1) allows queue data to be stored within a Coupling Facility.

Auxiliary temporary storage data is read in and written out of the CICS address space at the Control Interval level; that is, CICS will read and write entire Control Intervals to and from DFHTEMP. Within each Control Interval, record data for different auxiliary temporary storage queues may well be held contiguously. CICS maintains index information for retrieval of given records on each queue.

To optimize performance, an EXEC CICS DELETEQ of a temporary storage queue will not cause CICS to read in and rearrange every Control Interval that contains a record on the queue being deleted. Such a physical deletion of data on every DELETEQ would be expensive – a queue could have up to 32,767 records upon it, scattered across many Control Intervals on DFHTEMP.

To read in each Control Interval via I/O and then compress them to remove the redundant record data with Move Character Long (MVCL) instructions would increase path length and response time within temporary storage processing to an unacceptable level. Instead, CICS removes the references to the deleted queue from within its internal control blocks. It also updates the temporary storage byte map – this is a control block with a byte for every Control Interval on DFHTEMP.

The byte map records the amount of Free Space remaining within each Control Interval, and is used when CICS selects a target Control Interval for storing a new record, which will be added to an auxiliary temporary storage queue. By updating the bytes for the Control
Intervals containing records for a deleted queue, to reflect the fact that this space is now available once more, CICS logically deletes the data.

When a Control Interval is later selected to store a new auxiliary temporary storage record for a WRITEQ request, CICS will select a Control Interval that contains enough Free Space to accommodate the new record, by means of the byte map. CICS will then read the Control Interval into a temporary storage buffer in-core within the address space. It then has to compress the buffer to squeeze out all the redundant record space. It does this by determining the records within the Control Interval that are still required (ie still have existing queues) and moving them along into one contiguous block, starting at the beginning of the Control Interval.

This series of MVCL instructions results in the records for deleted queues being overwritten by this compression process, and the Free Space in the Control Interval being repositioned into a contiguous area at the end of the Control Interval. This Free Space can then be used to accommodate the new record being added.

As an aside, CICS uses this compression process as a good point to validate the consistency of the temporary storage control block data and Control Interval contents. If the Free Space in the buffer is not in agreement with the amount of Free Space, as calculated from the records found within the Control Interval and the temporary storage control blocks, corruption of some kind has occurred since this Control Interval was previously compressed. If such data integrity loss is detected, CICS issues a DFHTS1310 abend.

CONTROL INTERVAL SELECTION AND BUFFER COMPRESSION

A user migrated his CICS environment from CICS/ESA Release 4.1.0 to CICS Transaction Server. The workload and temporary storage definitions were kept constant between the two versions, and yet various puzzling differences in the temporary storage statistics data were observed.

Figure 1 shows some of the statistics fields produced by the temporary storage component of CICS, firstly from a CICS/ESA Release 4.1.0 region and then from a CICS Transaction Server region. As can be
seen, the workloads and temporary storage usage on both systems were very similar.

The fields that surprised the user were the ratio of ‘Number of Control Intervals available’ to the ‘Peak Control Intervals in use’. In CICS/ESA Release 4.1.0 this ratio was 75%, in CICS Transaction Server it was over 90%. Also, the ‘Number of temp storage compressions’ in CICS Transaction Server (that is, Control Interval compressions within temporary storage buffers) was only 5% of the value it used to be in CICS/ESA Release 4.1.0.

Taking the ratio of peak Control Intervals in use to total number of Control Intervals available first, it needs to be understood how CICS/ESA Release 4.1.0 managed the selection of Control Intervals to satisfy EXEC CICS WRITEQ requests. In particular, the notion of the ‘75 percent rule’ must be explained.

EXEC CICS WRITEQ requests on a cold-started CICS/ESA Release

<table>
<thead>
<tr>
<th></th>
<th>CICS/ESA</th>
<th>CICS Transaction Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of transactions</td>
<td>334,840</td>
<td>307,689</td>
</tr>
<tr>
<td>Put/Putq auxiliary storage</td>
<td>770,619</td>
<td>701,072</td>
</tr>
<tr>
<td>Get/Getq auxiliary storage</td>
<td>644,097</td>
<td>624,229</td>
</tr>
<tr>
<td>Peak temporary storage</td>
<td>5,780</td>
<td>5,619</td>
</tr>
<tr>
<td>Times queues created</td>
<td>286,558</td>
<td>272,469</td>
</tr>
<tr>
<td>Control Interval size</td>
<td>12,288</td>
<td>12,288</td>
</tr>
<tr>
<td>Longest auxiliary temp</td>
<td>8,048</td>
<td>8,024</td>
</tr>
<tr>
<td>Number of Control Intervals</td>
<td>24,000</td>
<td>23,999</td>
</tr>
<tr>
<td>Peak Control Intervals in use</td>
<td>18,006</td>
<td>21,793</td>
</tr>
<tr>
<td>Number of temp storage</td>
<td>396,505</td>
<td>20,137</td>
</tr>
</tbody>
</table>

*Figure 1: Comparison of statistics*
4.1.0 system are allocated into Control Intervals from the beginning of the DFHTEMP dataset. The first Control Interval is used for storing data for each WRITEQ command that is issued. Successive data records are written into the Control Interval contiguously from the start. This process continues until a WRITEQ request is made with data too large to fit the remaining Free Space within the Control Interval. At this point, CICS/ESA Release 4.1.0 switches to use the second Control Interval, and record data is added from the start of this. In such a way, successive Control Intervals are used to store temporary storage record data.

This process continues until 75% of the Control Intervals on DFHTEMP have been written to. EXEC CICS WRITEQ requests after this point are directed back to the start of the dataset once more. CICS uses the Free Space data from the byte map to select a Control Interval with enough Free Space to accommodate the requests. It looks back to the start of the dataset because, the theory is, old queues written earlier to CICS may well have been deleted by now. As explained above, such deleted queue data will remain in Control Intervals but no longer be required. If the byte map shows a Control Interval has sufficient Free Space for the new record being written, CICS will read it into a buffer and compress it to move all the required records to the start of the Control Interval.

The reason CICS/ESA Release 4.1.0 retained 25% of Control Intervals was to provide space for special header records. These are generated internally by CICS when handling records that are larger than the Control Interval size (‘spanned records’). Special header records require an empty Control Interval when being written. If DFHTEMP contained fragmented data in each Control Interval, a Special header record could not be stored. This is why CICS/ESA Release 4.1.0 tried to maintain a percentage of free Control Intervals for use by large items such as Special header records.

With CICS Transaction Server, temporary management still tries to maintain a percentage of empty Control Intervals for use in storing large records. However, the algorithm for providing this has been changed to only implement the ruling when DFHTEMP has been extended and cannot be enlarged any more. In other words, the total primary and secondary capacity of the dataset has been reached. The
reasoning behind this is that it is more efficient to allocate an empty Control Interval when writing a record to temporary storage than it is to read in a Control Interval via an I/O and then to compress it to generate a contiguous Free Space for use by new requests. Therefore, while it is still possible to enlarge the dataset and provide further empty Control Intervals for use by temporary storage, this action is preferable to the alternative of returning to the start of DFHTEMP and searching for an existing Control Interval with sufficient space to handle new WRITEQ requests.

With this understood, the second confusing statistic becomes understandable. Since new empty Control Intervals are being selected in preference to old ones being reused, the rate of reuse of existing Control Intervals is reduced. Therefore, the number of temporary storage buffer compressions is reduced.

TEMPORARY STORAGE PERFORMANCE CONSIDERATIONS
On migrating to CICS Transaction Server from CICS/ESA Release 4.1.0, a user noticed that the CPU costs and response times from their transactions were increasing during the run of CICS. The transactions were intensive users of auxiliary temporary storage. The system definitions for the numbers of buffers and strings, Control Interval sizes, and DFHTEMP attributes were all unchanged.

Analysis of the problem showed that the trace option TS=3 was being specified. With the restructure of temporary storage into a Domain, it now has its own trace component that can be set independently of the other traceable components of CICS. In CICS/ESA Release 4.1.0, temporary storage trace calls were made from within the Application Domain (AP).

Specifying TS=3 instructs CICS to perform consistency checking of its control blocks for any corruption that may have occurred. In effect, this is the same activity that could be carried out by a DFHTRAP in previous releases. In fact, such a DFHTRAP is available from IBM to perform consistency checking in CICS/ESA Release 4.1.0 – it is designed to capture the moment when a corruption occurs that could otherwise lead to a DFHTS1310 abend eventually occurring.
Having such validation work being carried out by CICS has a noticeable effect upon the pathlength of a temporary storage request, and the CPU consumption needed to achieve it. Because the consistency checking takes longer to complete, more auxiliary temporary storage data exists in the system – this explained the gradual degradation in performance and increase in CPU costs during a run of CICS.

The solution was to correct the trace option for the temporary storage trace component. TS=3 should only be set in either a test environment, or when a production system is known to be experiencing temporary storage corruption and a resolution to the problem requires such a pragmatic approach.

QIDERR VERSUS INVREQ
Prior to CICS Transaction Server Release 1, there was a restriction that prevented a temporary storage queue being created that had a null name (ie binary zeros, or X'000000000000000').

However, it was regarded as valid for applications to attempt to read or delete such a queue – in these circumstances, CICS/ESA Release 4.1.0 would return a qiderr response to the EXEC CICS READQ or DELETEQ API request. Qiderr is a ‘soft’ response that indicates to the application that a given queue does not exist. Applications should cater for the possibility of qiderr being returned on a READQ or DELETEQ request since it is quite likely that a queue may not exist (or did once exist but has since been deleted) when such a request is made.

With CICS Transaction Server, the decision was made to tighten up the API regarding this response. Because it is not possible to create a temporary storage queue that has a null name, applications should have no valid reason for attempting to issue EXEC CICS READQ or DELETEQ requests against such a queue.

Any such requests against a queue indicate that application logic is incorrect – most probably because the variable field used to specify the queue name value was not set up properly and so left set to its initial value (typically one of binary zeros). Therefore, a stronger API response than qiderr was deemed appropriate to be returned to the
application. CICS Transaction Server therefore returns invreq in such circumstances.

It was felt appropriate to differentiate between API calls against temporary storage queues that did not currently exist, but which could validly exist at some point under CICS (ie qiderr), as opposed to requests against queues that could never have existed (ie invreq).

SHARED TEMPORARY STORAGE CONSIDERATIONS
Shared temporary storage (or temporary storage data sharing) provides multiple MVS regions in a parallel sysplex with access to CICS temporary storage queue data. Shared temporary storage queues are held within pools; each such pool corresponds to a list structure within a coupling facility.

Shared temporary storage queues are non-recoverable. However, since they are stored within the coupling facility, they are normally preserved across a CICS restart or even an MVS re-IPL.

CICS systems that use shared temporary storage gain access to a pool of queues via a temporary storage data sharing server for a given pool. All such access is achieved via cross-memory calls to the server for the pool.

There are many benefits to using shared temporary storage for queue management. These include the performance improvement of accessing queue data from a coupling facility compared to function shipping the request to another CICS system (a Queue Owning Region or QOR).

Users considering implementing shared temporary storage can refer to further information on the CICS system definition, resource definition, and security implications from the CICS System Definition Guide, CICS Resource Definition Guide, and CICS RACF Security Guide.

CHANGES VISIBLE FROM THE CICS DUMP FORMATTER
The restructuring of temporary storage into a CICS Domain, and the implementation of object-oriented programming techniques, have led to a number of changes visible from the CICS dump formatter.
TS=2 formats the various temporary storage control blocks as before. In CICS Transaction Server, however, this now starts with the Domain and class anchors. The TSA is the Domain anchor block, and each class within the object-oriented code has its own anchor block too. These are followed by the DTN (Digital Tree Nodes) control blocks. These represent the tree structure used to maintain temporary storage queue nodes on a CICS system.

Each queue is then broken down into its component control blocks. In CICS Transaction Server, a queue is represented by a TSQ control block. Each record on a queue has a TSI control block to describe the item. For main queues, a TSI addresses a TSM control block, and for each auxiliary queue there is a TSX.

Additional control blocks (such as those specific to auxiliary and shared queue management) are also displayed if appropriate to the CICS system that generated the dump.

The various temporary storage control blocks visible when formatting a CICS Transaction Server system dump using the ‘TS’ verb exit are documented in the *CICS Problem Determination Guide*.

**SUMMARY AND CONCLUSIONS**

I hope that this article has helped explain the background to the CICS temporary storage Domain in CICS Transaction Server, and also given an indication of some of the variations that may be encountered when comparing temporary storage activity and statistics between CICS/ESA Release 4.1.0 and CICS Transaction Server.

*Editor’s note: readers wishing to discuss the material in this article can contact the author via e-mail at andy_wright@uk.ibm.com. CICS is a registered trademark of International Business Machines Corporation.*

---

*Andy Wright  
*CICS Change Team Programmer  
*IBM (UK)*

© IBM Corporation 1999
This month we conclude the redesigned system to record CEMT output to the CSMT transient data queue.

******************************************************************
P-PARSE-CEMT-OUTPUT.
* this routine handles up to 100 lines of output.
******************************************************************
MOVE L-WORKAREA-TIOALEN TO W-WORK-LENGTH.
MOVE L-WORKAREA-TIOA-DATA(1:W-WORK-LENGTH) TO W-TIOA-DATA.
MOVE +0 TO W-UNSTRING-COUNT.
UNSTRING W-TIOA-DATA
   DELIMITED BY C-SBA-CHAR
   INTO W-CEMT-OUTPUT-LINE(1)
                  W-CEMT-OUTPUT-LINE(2)
                  W-CEMT-OUTPUT-LINE(3)
                  W-CEMT-OUTPUT-LINE(4)
                  W-CEMT-OUTPUT-LINE(5)
                  W-CEMT-OUTPUT-LINE(6)
                  W-CEMT-OUTPUT-LINE(7)
                  W-CEMT-OUTPUT-LINE(8)
                  W-CEMT-OUTPUT-LINE(9)
                  W-CEMT-OUTPUT-LINE(11)
                  W-CEMT-OUTPUT-LINE(12)
                  W-CEMT-OUTPUT-LINE(13)
                  W-CEMT-OUTPUT-LINE(14)
                  W-CEMT-OUTPUT-LINE(15)
                  W-CEMT-OUTPUT-LINE(16)
                  W-CEMT-OUTPUT-LINE(17)
                  W-CEMT-OUTPUT-LINE(18)
                  W-CEMT-OUTPUT-LINE(19)
                  W-CEMT-OUTPUT-LINE(20)
                  W-CEMT-OUTPUT-LINE(21)
                  W-CEMT-OUTPUT-LINE(22)
                  W-CEMT-OUTPUT-LINE(23)
                  W-CEMT-OUTPUT-LINE(24)
                  W-CEMT-OUTPUT-LINE(25)
                  W-CEMT-OUTPUT-LINE(26)
                  W-CEMT-OUTPUT-LINE(27)
                  W-CEMT-OUTPUT-LINE(28)
                  W-CEMT-OUTPUT-LINE(29)
                  W-CEMT-OUTPUT-LINE(30)
                  W-CEMT-OUTPUT-LINE(31)
                  W-CEMT-OUTPUT-LINE(32)
                  W-CEMT-OUTPUT-LINE(33)
                  W-CEMT-OUTPUT-LINE(34)
                  W-CEMT-OUTPUT-LINE(35)
                  W-CEMT-OUTPUT-LINE(36)
TALLYING IN W-UNSTRING-COUNT
ON OVERFLOW
  MOVE 'CEMT OUTPUT TRUNCATED' TO W-MSG-TEXT
  MOVE '1Ø' TO W-MSG-NO
  PERFORM P-HANDLE-ERROR
END-UNSTRING.
PERFORM P-WRITE-TIOA
  VARYING K FROM +1 BY +1 UNTIL K > W-UNSTRING-COUNT.

******************************************************************
P-WRITE-TIOA.
******************************************************************
  MOVE W-CEMT-OUTPUT-DATA(K) TO W-TDQ-BUFFER.
  MOVE '***' TO W-TDQ-BUFFER(1:2).
  SUBTRACT +1 FROM LENGTH OF W-TDQ-BUFFER GIVING W-WORK-LENGTH.
  PERFORM VARYING J FROM +3 BY +1 UNTIL J > W-WORK-LENGTH
  EVALUATE TRUE
    WHEN W-TDQ-BUFFER(J:1) = C-INSERT-CURSOR
      MOVE SPACE TO W-TDQ-BUFFER(J:1)
    WHEN W-TDQ-BUFFER(J:1) = X'ØØ'
      MOVE SPACE TO W-TDQ-BUFFER(J:1)
    WHEN W-TDQ-BUFFER(J:1) = C-START-FIELD-CHAR
      MOVE SPACES TO W-TDQ-BUFFER(J:2)
      ADD +1 TO J
    WHEN W-TDQ-BUFFER(J:1) = LOW-VALUE
      MOVE SPACE TO W-TDQ-BUFFER(J:1)
    WHEN OTHER
      CONTINUE
  END-EVALUATE
END-PERFORM.
EVALUATE TRUE
  ******* DON'T SHOW ANYTHING AFTER THE 'STATUS:' LINE
    WHEN W-TDQ-BUFFER(3:7) = 'STATUS:'
      PERFORM P-WRITE-TDQ
      ADD W-UNSTRING-COUNT TO K
    WHEN W-TDQ-BUFFER(3:77) = SPACES
      CONTINUE
    WHEN W-TDQ-BUFFER(3:77) = LOW-VALUES

CONTINUE
WHEN OTHER
PERFORM P-WRITE-TDQ
END-EVALUATE.

******************************************************************
P-INQUIRE-REQID.
******************************************************************
MOVE C-REQID TO W-REQID.

EXEC CICS INQUIRE
  REQID(W-REQID)
  NOHANDLE
END-EXEC.

******************************************************************
P-CANCEL-REQID.
******************************************************************
PERFORM WITH TEST AFTER UNTIL EIBRESP NOT = DFHRESP(NORMAL)
  EXEC CICS CANCEL
    REQID(C-REQID)
    NOHANDLE
END-EXEC
END-PERFORM.

******************************************************************
P-START-AGAIN.
******************************************************************
**** FIRST CANCEL ANY OUTSTANDING REQIDS WITH THE SAME NAME.  
**** THIS PREVENTS INADVERTENT 'SPAWNING' PROBLEMS
PERFORM P-CANCEL-REQID.
EXEC CICS START
  TRANSID(C-MONITOR-TRANS-ID)
  AFTER
  MINUTES(W-INTERVAL-BIN-MINS)
  REQID(C-REQID)
  RESP(W-RESP)
  RESP2(W-RESP2)
END-EXEC.
IF W-RESP NOT = DFHRESP(NORMAL) THEN
  MOVE 'START TRANSID ERROR' TO W-MSG-TEXT
  MOVE '11' TO W-MSG-NO
  PERFORM P-HANDLE-ERROR
  GO TO ØØØØ-CICS-RETURN
END-IF.

******************************************************************
P-HANDLE-ERROR.
******************************************************************
IF W-RESP NOT = DFHRESP(NORMAL) THEN
  MOVE W-RESP TO W-RESP-PIC
  MOVE W-RESP2 TO W-RESP2-PIC
STRING
   W-MSG-TEXT DELIMITED BY ' '
   ',RESP=' W-RESP-PIC DELIMITED BY SIZE
   ',RESP2=' W-RESP2-PIC DELIMITED BY SIZE
   INTO W-MSG-TEXT
END-STRING
END-IF.
PERFORM P-WRITE-MSG.

******************************************************************
P-WRITE-TDQ.
******************************************************************
EXEC CICS WRITEQ TD QUEUE(C-MSG-QUEUE)
   FROM(W-TDQ-BUFFER)
   LENGTH(LENGTH OF W-TDQ-BUFFER)
   NOHANDLE
END-EXEC.
MOVE SPACES TO W-TDQ-BUFFER.

******************************************************************
P-WRITE-MSG.
******************************************************************
PERFORM P-GET-TIMESTAMP.
MOVE W-TIME TO W-MSG-TIME.
MOVE W-MSG TO W-TDQ-BUFFER.
PERFORM P-WRITE-TDQ.
MOVE SPACES TO W-MSG-TEXT.

******************************************************************
P-GET-TIMESTAMP.
******************************************************************
EXEC CICS ASKTIME
   ABSTIME(W-ABSTIME)
   NOHANDLE
END-EXEC.
EXEC CICS FORMATTIME
   ABSTIME(W-ABSTIME)
   DDMMYYYY(W-DDMMYYYY)
   DATESEP('/')
   TIME(W-TIME)
   TIMESEP(':')
   NOHANDLE
END-EXEC.

ZZZZCOUT
**********************************************************************
*                                      *
*   MODULE NAME = ZZZZCOUT         *
*                                      *

* DESCRIPTIVE NAME = CICS/ESA XZCOUT GLOBAL USER EXIT PROGRAM

* FUNCTION - CAPTURE CEMT OUTPUT FOR EVENTUAL WRITING TO CSMT
* LOG BY ZZZCEMT (Q.V.).

* GWA LENGTH = 91 BYTES

* STATUS = 4.1.Ø

* ERROR CONDITION CODES ARE SAVED IN THE GWA.
* PROGRAM ZZZCEMT CAN CHECK THIS CODE AND TAKE APPROPRIATE
* NOTIFICATION ACTION.

* ————————————————————————————————————————————*
* ————————————————————————————————————————————*
* USER EXIT INTERFACE FOR EXIT POINT XZCOUT
DFHUEXIT TYPE=EP,ID=(XZCOUT)

* TERMINAL INPUT/OUTPUT AREA (USED BY CEMT)
COPY DFHTIOA

* TERMINAL CONTROL TABLE
PRINT OFF,NOGEN
COPY DFHTCTTE
PRINT ON,GEN

* XPI DSECT FOR INQUIRE TRANSACTION CALL
COPY DFHXMIQY

* GWA AREA
GWA DSECT
GWA_DWORD DS D WORK
GWA_WORKPTR1 DS F -> WORKAREA 1
GWA_WORKPTR2 DS F -> WORKAREA 2
GWA_OFFSET DS F OFFSET IN WORKAREA
GWA_TIOALEN DS F TIOA LENGTH
GWA_TERMD IS CL4 TERMINAL-ID
GWA_TRANID DS CL4 TRANSACTION-ID
GWA_USERID DS CL8 USER-ID
GWAERRNO DS XL1 ERROR NO.
ALL_OK EQU X'ØØ' NO WORRIES
ERR_NO_WORKAREA EQU X'Ø1' NO VALID WORKAREA
ERR_XPI_CALL EQU X'Ø2' XPI CALL FAILED (INQ TRANS)
ERR_BUFFER_FULL EQU X'Ø3' TIOA WON'T FIT IN BUFFER
ERR_TCTTE_NULL EQU X'Ø4' TCTTE INVALID
GWAMSG DS CL49 FREE FOR ERROR MESSAGE
GWA_MAX_LEN EQU 65536 SIZE OF EACH WORKAREA
GWA_STATUS DS XL1 WORKAREA STATUS
GWA_INITIAL EQU X'ØØ' INITIAL STATE - NOT READY
GWA_WORK1 EQU C'1' WORKAREA1 ACTIVE
GWA_WORK2 EQU C'2' WORKAREA2 ACTIVE

GWALEN EQU *-GWA LENGTH OF GWA

* LENGTH OF HEADER FIELDS FOR SAVED CEMT DATA
* USERID + TERMID + TIOA LENGTH (PACKED)
HDRLEN EQU L'GWA_USERID+L'GWA_TERMID+3

**********************************************************************
* REGISTER EQUATES                                              *
**********************************************************************
*R1 EQU 1 UEP BASE ON ENTRY, WORK
UEPBAR EQU 2 -> USER EXIT PARMS
GWABAR EQU 3 -> GLOBAL WORK AREA
*R4 EQU 4 WORK
*R5 EQU 5 WORK
*R6 EQU 6 WORK
*R7 EQU 7 WORK
WORKPTR EQU 8 WORKAREA PTR
TIOABAR EQU 9 -> TIOA
TCTTEAR EQU 1Ø -> TCTTE
BASEREG EQU 11 PROGRAM BASE
*R12 EQU 12 WORK, UEPXSTOR BASE
*R13 EQU 13 SAVEAREA POINTER
*R14 EQU 14 WORK
LINKREG EQU 14 SUBROUTINE LINKAGE
*R15 EQU 15 ENTRY ADDRESS, RETURN CODE

EJECT

**********************************************************************
ZZZZCOUT AMODE 31
ZZZZCOUT RMODE ANY
ZZZZCOUT CSECT

SAVE (14,12) SAVE REGISTERS
LR BASEREG,R15 USING ZZZZCOUT,BASEREG SET UP PROGRAM BASE REGISTER
B START GO AROUND EYECATCHER
DC C'*ZZZZCOUT'*
DC C'*&SYSDATE*'
DC C'*&SYSTIME*'
DC C'*VERSION 1.Ø'*

START DS ØH
LR UEPBAR,R1 USING DFHUEPAR,UEPBAR -> USER EXIT PARAMETER LIST

* CHECK GWA
L R1,UEPGAL -> GWA LENGTH
CLC Ø(2,R1),=AL2(GWALEN) GWA LENGTH OK ?
* CHECK THAT THE WORKAREA POINTER IS VALID
  CLI  GWA_STATUS,GWA_INITIAL  OPEN FOR BUSINESS ?
  BE   RETURN                 NO, EXIT ->
  CLI  GWA_STATUS,GWA_WORK1   WORKAREA 1 ?
  BE   WORKAREA1             YES ->
  CLI  GWA_STATUS,GWA_WORK2   WORKAREA 2 ?
  BE   WORKAREA2             YES ->
  B    NO_WORKAREA           ELSE PROBLEM ->

* WORKAREA1 DS ØH
  ICM  WORKPTR,B'1111',GWA_WORKPTR1  GET WORKAREA 1 ADDR
  BZ   NO_WORKAREA            BAD ADDR ?? ? ->
  B    FIND_TERMINAL          ELSE CONTINUE ->

* WORKAREA2 DS ØH
  ICM  WORKPTR,B'1111',GWA_WORKPTR2  GET WORKAREA 2 ADDR
  BZ   NO_WORKAREA            BAD ADDR ?? ? ->

* FIND_TERMINAL DS ØH
  ICM  TCTTEAR,B'1111',UEPTCTTE  -> TCTTE
  BZ   NO_TCTTE                EXIT IF NONE ->
  MVC  GWA_TERMID,TCTTETI      SAVE TERMID

* FIND THE USER-ID USING XPI CALL
  L    R12,UEPXSTOR          -> XPI STORAGE
  LA   R12,4(R12)             LEAVE 1ST 4 BYTES
  USING DFHXMIQ_ARG,R12
  L    R13,UEPSTACK          -> KERNEL STACK
  DFHXMIQX CALL,CLEAR,        XPI CALL INQ TRANACTN +
     IN,                      INPUT PARMS +
     FUNCTION(INQUIRE_TRANSACTION),
     OUT,                      OUTPUT PARMS +
     TRANSACTION_ID(GWA_TRANID),
     USERID(GWA_USERID),
     RESPONSE(*),
     REASON(*)
  CLI  XMIQ_RESPONSE,XMIQ_OK  XPI CALL OK ?
  BNE  XMIQ_ERROR            NO ->

* IF TRANSACTION IS NOT 'CEMT', EXIT
  CLI  GWA_TRANID,C'C'        C TRANSID ?
  BNE  RETURN                NO ->
  CLC  GWA_TRANID+1(3),=CL3'EMT'  'CEMT' ?
  BNE  RETURN                NO ->

* COPY THE TIOA DATA TO THE ACTIVE WORKAREA. 1ST 8 BYTES IS USER-ID,
* THEN TIOA DATA LENGTH (2 BYTES), THEN TIOA DATA.
ICM TIOABAR,B'1111',UEPTIOA -> TIOA
BZ RETURN GET OUT IF NO TIOA
XR R7,R7
ICM R7,B'0011',TIOATDL GET TIOA DATA LENGTH
BZ RETURN GET OUT IF NO DATA
ST R7,GWA_TIOALEN SAVE TIOA LENGTH
L R1,GWA_OFFSET GET OFFSET
AR R1,R7 OFFSET + TIOA LENGTH
LA R1,HDRLEN(R1) + USERID + LEN
C R1,=AL4(GWA_MAX_LEN) > BUFFER SIZE ?
BH BUFFER_FULL YES ->
*
LR R5,R7 SET LENGTH
LA R6,TIOADBA -> TIOA DATA
LR R4,WORKPTR -> BUFFER
A R4,GWA_OFFSET -> WRITE AREA
MVC Ø(R4),GWA_USERID SAVE USER-ID
MVC Ø(R4),GWA_TERMID SAVE TERMINAL-ID
L R1,GWA_TIOALEN GET TIOA LENGTH
XC GWA_DWORD,GWA_DWORD CLEAR WORK FIELD
CVD R1,GWA_DWORD CONVERT TO DECIMAL
MVC Ø(R4)+4,R4,GWA_DWORD+8-3 SAVE LAST 3 BYTES
LA R4,HDRLEN(R4) BUMP PTR
MVCL R4,R6 COPY TIOA
*
* RECALCULATE NEW OFFSET
L R4,GWA_OFFSET GET OFFSET
LA R4,HDRLEN(R4) ADD LEN + USERID ETC
A R4,GWA_TIOALEN + TIOA LEN = OFFSET
ST R4,GWA_OFFSET SAVE IT
 *
B RETURN AND EXIT ->
EJECT
*
******************************************************************************
* RETURN TO CICS
******************************************************************************
RETURN DS ØH
L R13,UEPEPSA RESTORE R13
LA R15,UERCNORM SET RC = OK
RETURN (14,12),RC=(15)
EJECT
*
******************************************************************************
* ERROR HANDLING
******************************************************************************
NO_TCTTE DS ØH TCTTE ADDRESS ZERO
MVI GWAERRNO,ERR_TCTTE_NULL FLAG ERROR
MVC GWAMSG,=CL(L'GWAMSG')NO TCTTE'
B RETURN AND EXIT ->
*
BUFFER_FULL DS ØH TIOA WON'T FIT
Displaying CPU usage by TCB – revisited


The following amendments should be noted:

• In Issue 163, page 26, after the ‘Important: pictures should not be modified’ statement in the IPPCDTCB program, the line:

  Ø1 IPPCGTCB  PIC X(Ø8) VALUE 'IPPCDTCB'

should read:

  Ø1 IPPCGTCB  PIC X(Ø8) VALUE 'IPPCGTCB'

• In Issue 164, pages 39 and 40, the same program has two ‘PERFORM ACCESS-CALTAB’ lines. These should be ‘PERFORM ACCESS-GTCB’.

© Xephon 1999
Accessing CICS control blocks in COBOL

Many recently trained programmers and other IT industry workers often treat COBOL with contempt. However, on the mainframe the majority of existing applications are written in COBOL. More significant is the fact that new applications continue to be developed in COBOL – so the language must have some good points!

Recently, I have found myself at various client sites where I needed to examine some of the data in CICS control blocks. The main problem of being a consultant is that each environment you encounter is different – you cannot rely on the availability of an Independent Software Vendor (ISV) tool. However, almost always the installation has COBOL. So I decided to write a program I could take with me to sites to use for the purpose of examining internal CICS structures in COBOL that I knew would usually be available wherever I went. I found this to be much easier than might be thought.

THE SOURCE

Since the run-time environment available at different places is also different, I decided that it would be best to take the program in source code form. The application is extremely simple, consisting of a BMS mapset, a program, and three resource definitions. Theoretically, the program and mapset definitions are not required if program auto-install is being used (PGAIPGM=ACTIVE in the SIT), but it should be noted that the program must run in CICS key since it references areas of CICS key storage which are fetch-protected.

I am currently expanding the program and mapset in order to display more information, but the code included with this article could easily be modified and extended for your own purposes. Note that the code is written to be able to run in any CICS/ESA environment from Version 4.1 through to the latest available release, Version 5.3 (the CICS component of Transaction Server 1.3). It has been tested on all of those releases with the exception of 5.1, although I do not anticipate any problems in that version.
THE RESOURCE DEFINITIONS
The names used for these resources can be anything that conforms to your naming conventions. The illustrated definitions only show the relevant parameters; the normal CICS defaults can be assumed for all other values. All definitions must, of course, be placed in an appropriate group and installed.

Transaction ADDR
TRANSaction ==> addr
DEscription ==> control block display
PROGram ==> addrdisp
TASKDATAloc ==> Any
SPurge ==> Yes
TPUrgE ==> Yes

Program ADDRDISP
PROGram ==> addrdisp
Description ==> control block display
DATalocation ==> Any
EXECKey ==> CICS

Mapset ADDRMAP
Mapset ==> ADDRMAP
Description ==> Control Block Maps

ADDRDISP
IDENTIFICATION DIVISION.
PROGRAM-ID. ADDRDISP.
*
* Address manipulation — MVS & CICS control blocks
*
* This program will work in all versions of CICS/ESA from
* V4.1 through 5.3 (CICS Transaction Server 1.3). It is
* constructed in such a manner as to allow it to be
* adapted to later releases without great difficulty.
* All release-specific data names and values have a prefix
* of 'Vvr-' where 'v' is the Version number and 'm' the
* Release level, eg 'V41-'. Note that the Modification
* level is ignored when interrogating CICS as to its RELEASE
* via the INQUIRE SYSTEM command; this is based on the
* assumption that no general fix will change the basic
* number of Domains nor the number of gates allowed per
* Domain. However, newer releases may change some of the
* control structures referenced by the program, so careful
* review of the LINKAGE SECTION would need to be performed
* as part of any adaptation to a specific release.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.

* The following is used for determining which characters
* are "printable" when displaying data areas.

    CLASS PUNCTUATION IS '§' '.' '<' '(' '+ ' '|' ' & ' '!' '£'
    '∗ ' '} ' : ' > ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' ' '

DATA DIVISION.
WORKING-STORAGE SECTION.

Ø1 FILLER.

* NB. If the number of Domains ever exceeds 40, then the
* ADDRMAP, the following value, and the definition of
* KCB-VECTOR in the LINKAGE SECTION below must change.

Ø3 MAX-IN-MAP PIC 9(04) COMP VALUE 40.
Ø3 MAX-NO-OF-DOMAINS PIC 9(04) COMP VALUE 26.
88 V41-NO-DOMAINS VALUE 26.
88 V51-NO-DOMAINS VALUE 29.
88 V52-NO-DOMAINS VALUE 29.
88 V53-NO-DOMAINS VALUE 36.
Ø3 WHAT-VERSION.
Ø5 WV PIC X(Ø3).
88 V41 VALUE 'Ø41'.
88 V51 VALUE 'Ø51'.
88 V52 VALUE 'Ø52'.
88 V53 VALUE 'Ø53'.
Ø5 FILLER PIC X(Ø1).

* The following is used to format an entry on the main
* Domain information display panel.

Ø1 OUT-LINE VALUE SPACES.
Ø3 DD-ID PIC X(Ø2).
Ø3 FILLER PIC X(Ø1).
Ø3 DD-IX PIC 9(Ø2).
Ø3 FILLER PIC X(Ø1).
Ø3 DD-ADDR PIC X(Ø8).
Ø3 FILLER PIC X(Ø1).

* The following is used to format a line of data on
* the detail display panel.

Ø1 OUT-ADDR VALUE SPACES.
Ø3 OA-ADDR PIC X(Ø8).
Ø3 FILLER PIC X(Ø3).
Ø3 OA-OFF PIC X(Ø6).
Ø3 FILLER PIC X(Ø3).
Ø3 OA-CORE OCCURS 4.
Ø5 OA-DATA PIC X(Ø8).
Ø5 FILLER PIC X(Ø1).
Ø3 FILLER PIC X(Ø2).
Ø3 OA-EBCDIC PIC X(16).
Ø3 FILLER PIC X(Ø5).

* These are the various work areas required by the logic.

Ø1 FILLER.
Ø3 LINE-LIMIT PIC 9(Ø4) BINARY.
Ø3 SEG-LIMIT PIC 9(Ø4) BINARY.
Ø3 BYTE-LIMIT PIC 9(Ø4) BINARY.
Ø3 WORK-LIMIT PIC 9(Ø4) BINARY.
Ø3 DATA-LIMIT PIC 9(Ø4) BINARY.
Ø3 LENGTH-3 PIC 9(Ø4) BINARY VALUE 3.
Ø3 LENGTH-4 PIC 9(Ø4) BINARY VALUE 4.
Ø3 WORK-OFF PIC 9(Ø8) BINARY.
Ø3 WORK-OFF-X REDEFINES WORK-OFF.
Ø5 FILLER PIC X(Ø1).
Ø5 WO-LOW.
Ø7 WOL-BYTE PIC X(Ø1) OCCURS 3.
Ø3 WORK-LTH PIC 9(Ø04) BINARY.
Ø3 WORK-PTR PIC 9(Ø08) BINARY.
Ø3 WORK-PTR-X REDEFINES WORK-PTR.
Ø7 WP-BYTE PIC X(Ø1) OCCURS 4.
Ø3 ADDR-PTR REDEFINES WORK-PTR POINTERS.
Ø3 VECTOR-IX PIC 9(Ø4) COMP.
Ø3 IX PIC 9(Ø4) COMP.
Ø3 IW PIC 9(Ø4) COMP.
Ø3 MSGNO PIC 9(Ø04) BINARY.
Ø3 MSGS.
Ø5 FILLER PIC X(6Ø) VALUE 'Only <PF3>, <PF7> & <PF8> are valid.'.
Ø5 FILLER PIC X(6Ø) VALUE 'At last page.'.
Ø5 FILLER PIC X(6Ø) VALUE 'At first page.'.
Ø3 MSG REDEFINES MSGS PIC X(6Ø) OCCURS 3.
Ø3 FOUND-IND PIC X(Ø1) VALUE 'N'.
Ø8 FOUND-ONE VALUE 'Y'.

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
* As per usual, this program requires a BMS mapset.

COPY ADDRMAP.
COPY DFHBMSCA.
COPY DFHAID.

LINKAGE SECTION.

* Since two different maps are used for the two different types of data displayed, some info needs to be saved between the pseudo-conversational tasks. Also when data is being displayed, other data is required in order to control the paging.

Ø1 DFHCOMMAREA.
Ø3 LAST-DISPLAYED PIC X.
   88 MAIN-SHOWN VALUE 'M'.
   88 DETAIL-SHOWN VALUE 'D'.
Ø3 DOING PIC X.
   88 INPUT-MAIN VALUE 'M'.
   88 INPUT-DETAIL VALUE 'D'.
Ø3 DOMAIN PIC X(ØØ2).
Ø3 LAST-CURSOR PIC 9(ØØ4) BINARY.
Ø3 START-WORK PIC 9(ØØ8) BINARY.
Ø3 START-ADDR REDEFINES START-WORK POINTER.
Ø3 TOTAL-LTH PIC 9(ØØ4) BINARY.
Ø3 CURRENT-PAGE PIC 9(ØØ4) BINARY.
Ø3 TOTAL-PAGES PIC 9(ØØ4) BINARY.

* The following fields describe various MVS (OS/390) and CICS/ESA (CICS/TS) control structures. These are release-dependent, so need to be reviewed as part of any adaptation to any newer release of either.

* The PSA is a description of the MVS (OS/390) Prefixed Storage Area (low storage). This area is most unlikely to change since it is used by all CICS programs to locate DFHEIP.

Ø1 PSA.
Ø3 FILLER PIC X(54Ø).
Ø3 PSA-TCB-PTR POINTER.
Ø3 FILLER PIC X(ØØ4).
Ø3 PSA-ASCB-PTR POINTER.

* The ASCB is a description of the MVS (OS/390) Address Space Control Block.
* The ASSB is a description of the MVS (OS/390) Address Space Secondary Block.

* The JSAB is a description of the MVS (OS/390) Job Scheduler Address space Block.

* The TCB is a description of the MVS (OS/390) Task Control Block. This area is most unlikely to change since it is used by all CICS programs to locate DFHEIP.

* The TCBEXT is a description of the MVS (OS/390) Task Control Block EXTension. This area is most unlikely to change since it is used by all CICS programs to locate DFHEIP.

* The AFCB is a description of the CICS Authorized Function Control Block. The structure actually consists of a prefix, a vector list (a set of addresses), and a trailer. There is no direct pointer to the trailer but it can be found by adding the lengths of the prefix and the vector list to the address of the AFCB itself. The structure of this area is most unlikely to change since it is used by all CICS programs to locate DFHEIP.
* The AFT is a description of the CICS Authorized Function Trailer. This area is most unlikely to change.

Ø1 AFT.
Ø3 FILLER      PIC X(Ø04).
Ø3 AFT-AFCS-PTR POINTER.

* The AFCS is a description of the CICS Authorized Function Common Structure. This area is most unlikely to change.

Ø1 AFCS.
Ø3 FILLER      PIC X(Ø08).
Ø3 AFCS-KCB-PTR POINTER.

* The KCB is a description of the CICS/ESA Kernel Anchor Block.
* Although this structure is essentially the same in the releases on which this program was developed, note that the size of the array of vectors (addresses) to the Domain table entries is release-dependent. Also note that the offset (FILLER) to the array did change between Version 3 and Version 4.

Ø1 KCB.
Ø3 FILLER      PIC X(376).
Ø3 KCB-ERROR-VECTOR POINTER.
Ø3 KCB-VECTOR   POINTER
   OCCURS 26 TO 40
   DEPENDING ON MAX-NO-OF-DOMAINS.

* The DOMAIN-TABLE is a description of the CICS/ESA Kernel Domain table entry essential to the linkage architecture.
* The part of this area described is most unlikely to change.

Ø1 DOMAIN-TABLE.
Ø3 FILLER      PIC X(Ø03).
Ø3 DT-ID        PIC X(Ø02).
Ø3 FILLER      PIC X(Ø03).
Ø3 DT-IX        PIC 9(Ø08) BINARY.
Ø3 FILLER      PIC X(Ø04).
Ø3 DT-ANCHOR-PTR POINTER.
Ø3 DT-ANCHOR-PTR-X REDEFINES DT-ANCHOR-PTR
   PIC 9(Ø08) BINARY.

* The ANCHOR is a generic description of 256 bytes of any Domain's anchor block. The standard used by Domains is for the length of the control block to be placed in the first halfword of the block itself. There are, however a couple of exceptions to this rule.

* First the AP (Application) Domain's anchor block has an
* historical structure based on the original design of CICS
* in the late 1960s. This area is more commonly known as the
* Common System Area (CSA). For CICS/ESA this has been adapted
* so that it now contains a length in the second halfword of
* the CSA. However the length does not actually reflect the
* length as described by the data areas manual. This program
* uses the length found in the CSA.
*
* Second the RX (Recovery Services - introduced in 5.3)
* Domain's anchor block does not follow the standard
* convention. There is apparently no length contained within
* the control block itself, so the program assumes a size of
* 4096 (4K) bytes.
*
01 ANCHOR.
  03 A-BLOCK.
    05 A-LENGTH   PIC 9(004) BINARY.
    05 A-AP-LTH   PIC 9(004) BINARY.
    05 FILLER     PIC X(252).
  03 FILLER REDEFINES A-BLOCK.
    05 A-LINE     OCCURS 16.
    07 A-WORD     OCCURS 4.
    09 A-BYTE     PIC X(01)     OCCURS 4.

PROCEDURE DIVISION.
*
* First set the release-dependent values.
*
EXEC CICS INQUIRE SYSTEM
  RELEASE(WHAT-VERSION)
  NOHANDLE
END-EXEC
EVALUATE TRUE
  WHEN V41
    SET V41-NO-DOMAINS TO TRUE
  WHEN V51
    SET V51-NO-DOMAINS TO TRUE
  WHEN V52
    SET V52-NO-DOMAINS TO TRUE
  WHEN V53
    SET V53-NO-DOMAINS TO TRUE
END-EVALUATE
*
* The PSA is the start of virtual storage.
*
SET ADDRESS OF PSA TO NULL
IF EIBCALEN = 0
*
* Initially display the Main Domain Information.
*
EXEC CICS GETMAIN
  FLENGTH(LENGTH OF DFHCOMMAREA)
  SET (ADDRESS OF DFHCOMMAREA)
END-EXEC
MOVE LOW-VALUES TO DFHCOMMAREA
MOVE 1 TO LAST-CURSOR
PERFORM SEND-MAIN
ELSE
  * Remember what we last did.
  * MOVE LAST-DISPLAYED TO DOING
  *
  * Only certain Attention Identifiers (keys) are acceptable. So we take appropriate action.
  *
  EVALUATE EIBAID
    WHEN DFHENTER
      *
      <Enter> is only allowed on the main display to specify which anchor block is to be shown.
      *
      IF MAIN-SHOWN
        EXEC CICS RECEIVE
          MAP('MAINMAP')
          MAPSET('ADDRMAP')
          NOHANDLE
        END-EXEC
      *
      We need to get BMS to tell us where the cursor was left so we can display the corresponding anchor block.
      *
      EVALUATE EIBRESP
        WHEN DFHRESP(NORMAL)
          *
          Find which was requested.
          *
          PERFORM VARYING IX FROM 1 BY 1
          UNTIL IX > MAX-NO-OF-DOMAINS OR FOUND-ONE
            IF ANCHLNF(IX) = DFHBMCUR
              MOVE IX TO VECTOR-IX
              SET FOUND-ONE TO TRUE
            END-IF
          END-PERFORM
          IF FOUND-ONE
            PERFORM SEND-DETAIL
          ELSE
            PERFORM SEND-CONTROL
BEGIN-EXEC
  WHEN DFHRESP(MAPFAIL)
    PERFORM SEND-CONTROL
  WHEN OTHER
    *
    * This should never happen, but...
    *
    EXEC CICS ABEND
      ABOCODE('OOPS')
      NOHANDLE
  END-EXEC
  END-EVALUATE
ELSE
  MOVE 1 TO MSGNO
  PERFORM SEND-CONTROL
END-IF
WHEN DFHPF3
  *
  * <PF3> is used for the standard exit function.
  *
  IF MAIN-SHOWN
    EXEC CICS SEND
      FROM(END-MSG)
      ERASE
      NOHANDLE
  END-EXEC
  EXEC CICS RETURN
END-EXEC
ELSE
  PERFORM SEND-MAIN
END-IF
WHEN DFHPF8
  *
  * <PF8> is used to scroll forward if detail data
  * is displayed and there is more to show.
  *
  IF MAIN-SHOWN
    PERFORM SEND-CONTROL
END-IF
  IF CURRENT-PAGE = TOTAL-PAGES
    MOVE 2 TO MSGNO
    PERFORM SEND-CONTROL
END-IF
  ADD 1 TO CURRENT-PAGE
  MOVE LOW-VALUES TO DETLMAPO
  MOVE SPACES TO DMSGO
  PERFORM BUILD-IT
END-EXEC
WHEN DFHPF7
  *
  * <PF7> is used to scroll backward if detail data
  * is displayed and we have previously scrolled
* forward.
* 
IF MAIN-SHOWN
   PERFORM SEND-CONTROL
END-IF
IF CURRENT-PAGE = 1
   MOVE 3 TO MSGNO
   PERFORM SEND-CONTROL
END-IF
SUBTRACT 1 FROM CURRENT-PAGE
MOVE LOW-VALUES TO DETLMAP0
MOVE SPACES TO DMSGO
PERFORM BUILD-IT
WHEN DFHCLEAR
*
* <Clear> simply causes a screen refresh.
* 
IF MAIN-SHOWN
   PERFORM SEND-MAIN
ELSE
   MOVE LAST-CURSOR TO VECTOR-IX
   PERFORM BUILD-IT
END-IF
WHEN OTHER
*
* Any other key simply generates an error message.
* 
MOVE 1 TO MSGNO
PERFORM SEND-CONTROL
END-EVALUATE
END-IF
EXEC CICS RETURN
   TRANSID(EIBTRNID)
   COMMAREA(DFHCOMMAREA)
END-EXEC
.
SEND-MAIN.
*
* Indicate what we have done.
* 
   SET MAIN-SHOWN TO TRUE
*
* Clear the output map area and set where we want the cursor.
*
   MOVE LOW-VALUES TO MAINMAPO
   MOVE -1 TO ANCHLN(LAST-CURSOR)
*
* Place the Job ID, Job Name, and CICS release in the output map.
* 
   PERFORM ADDRESS-JSAB
   MOVE JSAB-JOBID TO MJOBIDO
MOVE JSAB-JOBNAME TO MJOBNMO
MOVE WHAT-VERSION TO MRELO

* Fill in the Domain data.

* PERFORM ADDRESS-KCB
PERFORM VARYING VECTOR-IX FROM 1 BY 1
UNTIL VECTOR-IX > MAX-NO-OF-DOMAINS
SET ADDRESS OF DOMAIN-TABLE TO KCB-VECTOR(VECTOR-IX)
IF DT-ID ALPHABETIC
MOVE DT-ID TO DD-ID
MOVE DT-IX TO DD-IX
SET ADDR-PTR TO DT-ANCHOR-PTR
CALL 'HEXMANIP' USING DT-ANCHOR-PTR,
LENGTH-4,
DD-ADDR
MOVE OUT-LINE TO ANCHLNO(VECTOR-IX)
IF WORK-PTR = ZERO
MOVE DFHBMASK TO ANCHLNA(VECTOR-IX)
END-IF
ELSE
* NB The display fields are unprotected in order to allow
* the user to use the <Tab> key to position the cursor.
* So if the table entry is unused, we protect it.
* MOVE DFHBMASK TO ANCHLNA(VECTOR-IX)
END-IF
END-PERFORM

* In order to allow the map to be used across all of the
* releases (and possibly for those in the future as well),
* the excess positions on the map are protected.
* IF MAX-NO-OF-DOMAINS < MAX-IN-MAP
PERFORM VARYING VECTOR-IX FROM MAX-NO-OF-DOMAINS
BY 1
UNTIL VECTOR-IX > MAX-IN-MAP
MOVE DFHBMASK TO ANCHLNA(VECTOR-IX)
END-PERFORM
END-IF
EXEC CICS SEND
MAP('MAINMAP')
MAPSET('ADDRMAP')
ERASE
CURSOR
NOHANDLE
END-EXEC
.
SEND-DETAIL.
* Clear the output map area.
  * 
  MOVE LOW-VALUES TO DETLMAPO
  MOVE SPACES TO DMSGO
  *
* Address the requested anchor block.
  *
  PERFORM ADDRESS-KCB
  SET ADDRESS OF DOMAIN-TABLE TO KCB-VECTOR(VECTOR-IX)
  IF (DT-ID ALPHABETIC)
  AND (DT-ANCHOR-PTR-X NOT = ZERO)
  *
  For valid entries, display the requested data.
  *
  MOVE DT-ID TO DOMAIN, DDOMIDO
  MOVE VECTOR-IX TO LAST-CURSOR
  *
  Indicate what we have done.
  *
  SET DETAIL-SHOWN TO TRUE
  SET ADDRESS OF ANCHOR TO DT-ANCHOR-PTR
  SET START-ADDR TO DT-ANCHOR-PTR
  MOVE 1 TO CURRENT-PAGE
  EVALUATE DT-ID
  *
  Allow for the exceptions to the length convention.
  *
  WHEN 'AP'
  MOVE A-AP-LTH TO TOTAL-LTH
  WHEN 'RX'
  MOVE 4Ø96 TO TOTAL-LTH
  WHEN OTHER
  MOVE A-LENGTH TO TOTAL-LTH
  END-EVALUATE
  COMPUTE TOTAL-PAGES = ((TOTAL-LTH - 1) / 256) + 1
  PERFORM BUILD-IT
  ELSE
  PERFORM SEND-CONTROL
  END-IF
  *
  BUILD-IT.

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

Jerry Ozaniec
Circle Computer Group (UK)  © Xephon 1999
Listing the TMONCICS control file RLC definitions

Landmark’s The Monitor for CICS/ESA is a widely used third-party monitoring software package for CICS installations. Amongst the facilities offered by The Monitor is the Resource Level Cancel (RLC) facility. This enables CICS performance staff to set limiting values for the critical processor resources consumed by a CICS transaction. If the transaction exceeds a defined limit, The Monitor attempts to cancel the transaction in order to protect the rest of the CICS system from the effects of the runaway transaction.

The resources are CPU, storage, and I/O operations. These are further broken down into task CPU time consumed and task elapsed time, task storage above and below 16MB, and I/O to DL/I, DB2, and to a user-defined database such as IDMS or ADABAS. In addition, The Monitor allows every transaction to have a limit on the number of individual EXEC CICS operations that it is permitted to execute.

To define these RLC values, Landmark offers a set of screens within the on-line portion of The Monitor, known as the Cross System Monitor or CSM, and they are stored in the control file. This is VSAM-based. There is, however, no way to list the definitions, other than by using the same set of screens.

To bypass having to use the CSM screens, and to quickly generate a listing of all the definitions currently in place, I have written the REXX program, TMCERLC, which formats the contents of the control file and writes a report as shown in Figure 1.

The report is mostly self-explanatory, the fields being exactly as defined in The Monitor documentation. The EIBFN lines show where transactions have been limited to the number of EXEC CICS functions they can execute, and show the internal CICS code, the number of that particular function that is allowed, and a description of the CICS function. The table of these descriptions does not contain all possible EXEC CICS functions, but it can be easily modified by adding function descriptions if desired. When the description ‘Type not defined’ appears, simply add an entry with the code shown and a description, either from The Monitor CSM RLC definition screens, or from the documentation in the relevant CICS Diagnosis Reference manual.
```
parse pull tmnrec
  call proc_rec
end
else
  done = 'y'
end
exit Ø
/*—————————————————————————————————*/
/* Process a record */
/*—————————————————————————————————*/
proc_rec:
type = substr(tmnrec,1,1)
if type = 'K' then
do
  jobn = substr(tmnrec,2,8)
  if jobn ≠ sjob then
    say ''
    sjob = jobn
  tran = substr(tmnrec,18,4)
  select
    when tran = '00000000'x then return
    when trlc = '7FFFFFFF'x then return
    otherwise
      do
        rcpu = substr(tmnrec,49,4)
        rsa = substr(tmnrec,53,4)
        rsb = substr(tmnrec,57,4)
        rrun = substr(tmnrec,61,4)
        rdb2 = substr(tmnrec,65,4)
        rusr = substr(tmnrec,73,4)
        xhi = substr(tmnrec,77,1)
        xlog = substr(tmnrec,78,1)
        xwt = substr(tmnrec,79,1)
        if rcpu ≠ '7FFFFFFF'x then
          rcpu = c2d(rcpu) * 64 / 1000000
        else
          rcpu = Ø
        if rsb ≠ '7FFFFFFF'x then
          rsb = c2d(rsb) / 1024
        else
          rsb = Ø
        if rsa ≠ '7FFFFFFF'x then
          rsa = c2d(rsa) / 1024
        else
          rsa = Ø
        if rrun ≠ '7FFFFFFF'x then
          rrun = c2d(rrun) * 64 / 1000000
        else
          rrun = Ø
        if rdb2 ≠ '7FFFFFFF'x then
          rdb2 = c2d(rdb2) / 1024
        else
          rdb2 = Ø
        if rusr ≠ '7FFFFFFF'x then
          rusr = c2d(rusr) / 1024
        else
          rusr = Ø
        if xhi ≠ '7FFFFFFF'x then
          xhi = c2d(xhi) / 1024
        else
          xhi = Ø
        if xlog ≠ '7FFFFFFF'x then
          xlog = c2d(xlog) / 1024
        else
          xlog = Ø
        if xwt ≠ '7FFFFFFF'x then
          xwt = c2d(xwt) / 1024
        else
          xwt = Ø
        when tran = '00000000'x then return
      end
  end
end
rdli   = c2d(rdli)
else
  rdli   = Ø
if rdb2 ¬= '7FFFFFFF'x then
  rdb2   = c2d(rdb2)
else
  rdb2   = Ø
if rusr ¬= '7FFFFFFF'x then
  rusr   = c2d(rusr)
else
  rusr   = Ø
say jobn tran format(rcpu,4,Ø) format(rsb,6) format(rsa,6),
   format(rrun,4,Ø) format(rdli,8) format(rdb2,8),
   format(rusr,8) '  ' xhi '  ' xlog '  ' xwt
j = 8Ø
do i = 1 to 32
  eibrec = substr(tmnrec,j,6)
  eib = substr(eibrec,1,2)
  select
    when eib = 'ØØØØ'x then nop
    when eib = 'FFFF'x then nop
    otherwise do
      k = c2x(eib)
      say '   EIBFN' c2x(eib) '    ',
           format(c2d(substr(eibrec,3,4)),6) '    ' eibtyp.k
    end
  end
  j = j + 6
end
end
end
return
/*———————————————————————————————————————————————————————————————————*/
/* Initialize EIB descriptions - not an exhaustive list, add */
/* additional descriptions as required, see CICS Diagnosis Reference */
/* for a list of command descriptions. */
/*———————————————————————————————————————————————————————————————————*/
init_eib:
eibtyp.     = 'Type not defined'
eibtyp.Ø2Ø2 = 'Address'
eibtyp.Ø2Ø4 = 'Handle condition'
eibtyp.Ø2Ø6 = 'Handle aid'
eibtyp.Ø2Ø8 = 'Assign'
eibtyp.Ø2ØA = 'Ignore condition'
eibtyp.Ø2ØC = 'Push'
eibtyp.Ø2ØE = 'Pop'
eibtyp.Ø4Ø2 = 'Address set'
eibtyp.Ø4Ø4 = 'Receive'
eibtyp.Ø4Ø6 = 'Send'
eibtyp.Ø4Ø8 = 'Converse'
eibttyp.040C = 'Wait terminal'
eibttyp.0410 = 'Wait signal'
eibttyp.0420 = 'Allocate'
eibttyp.0422 = 'Free'
eibttyp.042C = 'Wait convid'
eibttyp.043E = 'Extract process'
eibttyp.0430 = 'Issue abend'
eibttyp.0432 = 'Connect process'
eibttyp.0602 = 'Read'
eibttyp.0604 = 'Write'
eibttyp.0606 = 'Rewrite'
eibttyp.0608 = 'Delete'
eibttyp.060A = 'Unlock'
eibttyp.060C = 'Startbr'
eibttyp.060E = 'Readnext'
eibttyp.0610 = 'Readprev'
eibttyp.0612 = 'Endbr'
eibttyp.0614 = 'Resetbr'
eibttyp.0802 = 'Writeq TD'
eibttyp.0804 = 'Readq TD'
eibttyp.0806 = 'Deleteq TD'
eibttyp.0A02 = 'Writeq TS'
eibttyp.0A04 = 'Readq TS'
eibttyp.0A06 = 'Deleteq TS'
eibttyp.0C02 = 'Getmain'
eibttyp.0C04 = 'Freemain'
eibttyp.0E02 = 'Link'
eibttyp.0E04 = 'Xctl'
eibttyp.0E06 = 'Load'
eibttyp.0E08 = 'Return'
eibttyp.0E0A = 'Release'
eibttyp.0E0C = 'Abend'
eibttyp.0E0E = 'Handle abend'
eibttyp.1002 = 'Asktime'
eibttyp.1004 = 'Delay'
eibttyp.1006 = 'Post'
eibttyp.1008 = 'Start'
eibttyp.100A = 'Retrieve'
eibttyp.100C = 'Cancel'
eibttyp.1202 = 'Wait event'
eibttyp.1204 = 'Enqueue'
eibttyp.1206 = 'Dequeue'
eibttyp.1208 = 'Suspend'
eibttyp.1402 = 'Write journalnum'
eibttyp.1404 = 'Wait journalnum'
eibttyp.1602 = 'Syncpoint'
eibttyp.1802 = 'Receive map'
eibttyp.1804 = 'Send map'
eibttyp.1806 = 'Send text'
eibttyp.1808 = 'Send page'
eibttyp.180A = 'Purge message'
JCL TO RUN TMCERLC

Because REXX cannot read a VSAM file directly, it is necessary to reproduce the control file data into a sequential dataset before running TMCERLC. The following JCL achieves this:

```
//TMCERLC  JOB
//*
//EXTRCNTL EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//IN       DD DSN=TMCE.V20.VTCECNTL,DISP=SHR
//OUT      DD DSN=&&CNTL,DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(1,1)),
//          DCB=(RECFM=VB,LRECL=27994,BLKSIZE=27998)
//SYSIN    DD *
//        REPRO INFILE(IN) OUTFILE(OUT) FROMKEY(K) TOKEY(K) /* RLC RECS */
//*
//TMCERLC  EXEC PGM=IRXJCL,PARM='TMCERLC'
//SYSEXEC  DD DSN=SYS1.SYSEXEC,DISP=SHR
//TMNDAT   DD DSN=&&CNTL,DISP=(OLD,DELETE)
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=* 
//*
```

Patrick Mullen  
System Software Consultant (Canada)  
© Xephon 1999
InfoSpinner has announced CICS support for its ForeSite Application Server. ForeSite Transaction Server, a new System Integration Module (SIM), has been released as part of a suite of application integration products.

Transaction Server SIM is an add-on to ForeSite Application Server 3.0, and can be installed on ForeSite’s PageServers to run concurrently with CICS software. It allows ForeSite to interface with transaction server software and update large databases as transactions occur on a Web site. It also provides OLTP management for applications on both IBM and non-IBM platforms.

For further information contact:
InfoSpinner, 1601 North Glenville Drive, Suite 108, Richardson, TX 75081, USA.
Tel: (972) 479 0135.

CICS users can benefit from Software AG’s announcement of support for Java applications on OS/390 mainframes for its Bolero Application Factory.

Although Bolero creates Java Byte Code that can theoretically run on all platforms that have a Java Virtual Machine, Java implementations are different on mainframes, NT, and Unix. The OS/390 version of Bolero supports mainframes specifically, and allows Bolero applications to be used in the CICS Open Transaction Environment and to store persistent objects in DB2 databases.

For further information contact:
Software AG (UK), Charter Court, 74/78 Victoria Street, St Albans, Herts, AL1 3XH, UK.
Tel: (01727) 844 455.
Software AG of America, 11190 Sunrise Valley Drive, Reston, VA 22091, USA.
Tel: (703) 860 5050.

IBM has announced CICS Transaction Server for VSE/ESA Release 1. The product includes a new CICS Server, CICS Universal Clients, CICS Transaction Gateway, REXX for CICS, CICS Distributed Data Management, and CICS/ VSE Version 2.3, including Report Controller Facility (RCF) for both levels of CICS server. Future additions will include a CICS Web Interface (CWI), and the CICS 3270 Bridge.

Release 1 includes a new facility for viewing CSMT messages in the coexistent CICS/VSE Version 2.3. Translated versions of CICS Transaction Server for VSE/ESA messages and RCF panels are provided in Japanese, simplified Chinese, and German.

For further information contact your local IBM representative.