In this issue

3  Managing SMF type 110 volume with the CICS MCT
6  CICS Performance Monitor for z/OS V1.2 enhancements
13 Loop analysis – hints and tips
28 Helpful exit for shutdown assistant users – part 2
41 Audit trail for CICS maxtask events
50 CICS news

© Xephon Inc 2004
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 $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our Notes for Contributors from www.xephon.com/nfc.
Managing SMF type 110 volume with the CICS MCT

The topic of how to manage huge volumes of SMF data seems to be raised regularly, and I have noticed that many shops are unaware that CICS offers a very simple method of significantly reducing the quantity of information that is written in the CICS monitoring records (SMF type 110, subtype 1) without, necessarily, any loss of functionality in their SMF reporting system.

As an example, a medium-sized site was producing an average of about 4.5GB of SMF data per day. A quick examination of the output from IFASMFDP, the IBM-supplied program that copies SMF records from the SYS1.MANx collection files, typically, to a daily collection file in the form of a GDG, showed that SMF record type 110 accounted for 1.9GB or 42% of that total.

A few words here on IFASMFDP – the output from this program is somewhat misleading. The report shows Records Read, Percent of Total, and Avg. Record Length. The percentage shown is simply the count of a specific record type divided by the total number of records. This does not take into account the record length, which varies enormously between record types.

Thus a typical IFASMFDP report may show type 110 records as being only about 3% of the SMF record count, but these records are usually very much larger than other SMF record types. A type 110 record is usually close to 32KB in size, whereas many record types are only a few hundred bytes long.

To get a better picture, copy and paste the IFASMFDP report into a spreadsheet program like Microsoft Excel. Multiply the record count by the average record length, sum all these values for all record types, then calculate a percentage of this sum for each record type.

The site mentioned above was producing about 80,000 type 110 records per day, with an average length of about 24KB. This quantity of data was caused by about 1.5 million CICS transactions per day. The SMF type 110 subtype 1 records were required for
accounting and billing systems, as well as for performance monitoring, tuning, and capacity planning purposes. This meant that there was no question of simply turning CICS monitoring off. Almost all of the type 110 records were subtype 1 (i.e., CICS monitoring), less than 1% being subtype 2 (CICS statistics records), and no other subtypes were present.

Each and every one of those 1.5 million CICS transactions was writing 1,288 bytes of CICS monitoring information to the CICS SMF interface. This is referred to as a ‘segment’ of SMF data. CICS buffers these segments and only writes a type 110 record to SMF when the record is full, which in this case is every 24 transactions. This value is 24 because of the maximum SMF record size being 32KB, so, allowing for the SMF record header information, only 24 of the 1,288 byte segments can fit in a record.

However, of that 1,288 bytes of information, very little (actually in total about 60 bytes) was used by the accounting and performance monitoring systems into which it was being fed at the site in question.

The solution to this problem of large quantities of unnecessary data is to implement a CICS Monitoring Control Table (MCT), which limits the contents of the type 110 subtype 1 record to just those fields that are necessary, instead of allowing CICS to use the IBM-supplied default MCT, which, in CICS TS 1.3, results in the 1,288 byte segment size.

The default MCT is used by any CICS region displaying this message at start-up time:

```
DFHMN0105I vtam_applid Using default Monitoring Control Table.
```

It is a kind of ‘everything but the kitchen sink’ version of the table, in which every field identified in the table in the Default CICS Dictionary Entries table in the chapter on CICS monitoring in the Customization Guide is included. Note that this table spans several pages in the manual.

In fact it can be even worse – if a CICS region has a specially-coded MCT incorporating Event Monitoring Points (EMPs), such
as those which the CICS sockets interface developers recommend be included, these are appended to the 1,288 bytes. Segment lengths of over 2,000 bytes can easily result.

What is required is an MCT, which uses the EXCLUDE statement on the DFHMCT TYPE=RECORD macro. If necessary, an INCLUDE statement may be used as well, but note that an INCLUDE without an EXCLUDE does nothing. The details are explained in the Resource Definition Guide, in the chapter on the MCT, so I will not labour them here.

To tailor an MCT so it doesn’t flood SMF datasets with never-used information, start by finding out exactly which fields from the SMF type 110 record segment are used in your SMF reporting and analysis system. The chances are that you will find it is a very short list.

The MCT refers to each field by a field number. Cross references to recognizable field names are listed in both the Customization Guide and the Resource Definition Guide.

Once you have a list of fields, coding the MCT is trivial. If the list of fields is small, simply code:

```
EXCLUDE=ALL, INCLUDE=(a,b,c,d,e,...)
```

Note that EXCLUDE=ALL does not actually exclude every field. There are six fields that cannot be excluded: specifically, fields 1, 2, 4, 5, 6, and 89. These correspond to the following:

- 1 = Transaction id
- 2 = Terminal id
- 89 = User id
- 4 = Transaction type
- 5 = Start time
- 6 = Stop time

This corresponds to the shortest possible SMF type 110 segment, and is 36 bytes long. These are always the first six fields in the segment, and are always in the order shown (ie user id is the third field, despite being identified as field number 89).

If, for instance, your investigation revealed that in addition to the non-excludable fields you needed just user task CPU time (field 8), program name (field 71), and logical unit name (field 111), you would code:
These additional fields total 24 bytes, for a total segment length of 60 bytes, and a saving of 1,228 bytes per CICS transaction. In the previous example of 1.5 million transactions per day, the daily quantity of type 110 SMF data drops from 1.9 gigabytes to 90 megabytes.

If it turns out that you need more of the CICS-supplied fields, they can be included using a group-level nomenclature (eg INCLUDE=(DFHTASK)). If a very large number, but still not all, of the fields are required, then specific fields, or groups of fields, can be excluded (eg EXCLUDE=(DFHTASK) instead of EXCLUDE=ALL). Again, this is all comprehensively documented in the Resource Definition Guide, including listings of which fields belong to which groups.

With a very short segment, CICS does not have to pass data to SMF as frequently, with fewer than 5% of the original SMF calls required in our example. The CPU savings resulting from the CICS region neither having to build nor write so many SMF records are very small, though, not least because CICS always collects the data internally, even if it never writes any out.

Where savings can be seen is in the dramatic reduction of the volume of SMF data, which in many sites is stored on disk and archived to tape, all of which takes time and management. Processing and analysing this data will also show healthy CPU and elapsed time reduction, all-in-all a useful measurable payback for a very small investment.

Patrick Mullen
Consultant (Canada)

CICS Performance Monitor for z/OS V1.2 enhancements

In previous articles (see CICS Update issues 213, 216, and 217) we have looked at the functions provided by CICS Performance
Monitor for z/OS V1.1. In this article we will look at some of the additional features provided by CICS Performance Monitor V1.2, namely its historical task data capabilities and exploitation of recently-shipped additional run-time data and task kill support within CICS Transaction Server.

HISTORICAL TASK DATA
In a highly-active system, many transactions have a fleeting life as a currently-active task inside CICS. Historical task data is the only way of seeing what your systems are really doing. Questions such as, “How many instances have run in the last half an hour?”, “Are the response times OK?”, and “Have I had any tasks that have abended, and are they specific transactions?” are all questions that can be answered by looking at historical task data.

CAPTURING HISTORICAL TASK DATA
Recent enhancements made to CICS Transaction Server 1.3,
2.2, and 2.3 allow the capture of historical task data. VSAM KSDSs can be allocated to each CICS region to store this data.

Figure 2: Discarding capture rules

Figure 3: History dataset activity
and CICSplex SM capabilities can be used to manage, capture, and view historical task data. By recording in this manner, the recording is independent of the CMAS, and data is available in the event of an address space or LPAR failure. This also optimizes the performance of capturing data for later on-line analysis.

What data is to be captured is specified via the CICS Performance Monitor GUI. These panels are added as extra tabs to the existing panels provided in V1.1. Here the transaction id can be specified explicitly or wild-carded. The matching tasks can also be included or excluded from capture via this method (see Figure 1). These capture rules can be installed into the run-time in the same manner as thresholds in CICS PM V1.1.

The currently active capture rules can also be observed via the run-time views. From here the capture rules may be discarded from the run-time if no longer required (see Figure 2).

Activity on the history datasets can also be observed from the GUI (see Figure 3).

Figure 4: Launch screen
When recording data to the history datasets, the datasets are flip-flopped to allow continuous recording of the most recent data. The amount of data recorded is limited only by the amount of DASD allocated to the datasets.

VIEWING CAPTURED DATASET

The CICS Performance Monitor GUI provides the ability to launch the Web User Interface to display the captured data via a comprehensive set of filter criteria. The launch screen is shown in Figure 4.

As can be seen, data can be displayed for various filter criteria like termid, tranid, file request count, etc. Various slices can also be made through the data with regard to time interval. After entering the desired filter criteria and pressing Report, the corresponding WUI view is presented (see Figure 5).

Figure 5 shows a tabular view of historical task data. Only a few fields are visible here. The actual data provided is extensive, covering all the data presented at the XMNOUT CICS exit.
Navigation as usual is integrated with all the other WUI views for simple one-click navigation through the data provided via the Web User Interface.

This support is made available via CICS PM interfaces and CICS TS base CPSM facilities via the extensions to the MODEF resource table and the provision of the HTASK and MASHIST resource tables.

ADDITIONAL RUN-TIME DATA SUPPORT

CICS Performance Monitor V1.2 also exploits data recently made available via a PTF on CICS Transaction Server. WUI views and extensions to thresholding are now also provided for:

- **XEXITGLUE** – global user exit:
  - enable/disable exit.
- **XEXITTRUE** – task related user exit:
  - enable/disable exit.
- **XSTREAM** – log stream name:
  - set keypoint frequency and log defer interval.
- **XLSRPBUF** – VSAM LSRPOOL buffers.
- **XDSPGBL** – CICS Dispatcher Global:
  - inq/set FORCEQR.
- **XDSPPOOL** – CICS Dispatcher TCB pool:
  - inq/set MAXOPENTCBS – maximum L8 open mode TCBs.
  - inq/set MAXJVMTCBS – maximum J8 mode open TCBs.
  - inq/set MAXHPTCBS – maximum H8 mode open TCBs.
- **X2TASK** – active task:
  - Correlation UOW id.
• XMONITOR – monitoring and statistics:
  – inq FILELIMIT – file resource limit.
  – inq TSQUEUELIMIT – Temporary Storage queue limit.
  – inq APPLNAMEST – application naming option.
  – inq RMISTATUS – RMI option.
• TSKSPOLS – all task subpools.
• MVSTCBGL – global MVS TCB information.
• MVSTCB – information for individual MVS TCBs.
• MVSESTG – information about a single MVS storage element.
• EXTGLORD – similar to XEXITGLUE:
  – Exit point.
  – Position of the program within the Exit point.
  – Program name.
  – Usecount and concurrency status.
• TASKRMI – Resource Manager Interface usage by individual task.
• TASKTSQ – Temporary Storage usage by individual task.
• TASKESTG – Task Storage Element.

KILLING TASKS IN A RUNNING CICS SYSTEM
CICS Performance Monitor for z/OS V1.2 also exploits the kill feature recently provided on CICS TS 2.2 and 2.3. The need for a kill function arose from the users’ need to purge problem tasks under all circumstances. Situations can arise where a user cannot list tasks in a region or cancel a problem task, eg a waiting or looping CICS task can bring the CICS region to a halt. It can also occur because the CPSM listener runs under QT TCB and the QT TCB may be dominated by one or more looping tasks or execution of the listener task can be delayed by a shortage of
resources, eg storage. Typical usage of kill is as a last resort before recycling the system. This may keep the system up and allow Service Level Agreements to be met.

There are two main elements to CICS TS’s solution.

It enhances the existing support for cancelling tasks via earlier detection of deferred purge. It will initiate purge (or forcepurge) for the ‘oldest’ instead of the ‘first found’ task marked for purge. Detection of deferred purge is on entry to and exit from EXEC for tasks in a tight loop between the application code and the upper EXEC layers. This reduces the need for kill.

It provides a ‘KILL’ function that is separated from CICS. A new transaction is invokable from the MVS console. A subset of inputs and outputs available with the INQUIRE and SET commands are available to identify the tasks under consideration. Requests are executed under a TCB other than the QR TCB. In this way, tasks can be enquired about and killed even when the QR TCB is blocked.

Kill is supported wherever PURGE FORCEPURGE is provided today, ie against TASK, CONNECTION, and TERMINAL resources.

Dr Paul Johnson
CICS Transaction Server Systems Management Planning/Development
IBM Hursley (UK) © IBM 2004

Loop analysis – hints and tips

CICS application program logic errors (or unexpected data or response values) can lead to loops occurring within application programs. If unbounded, these loops can have a serious effect on the throughput and performance of the CICS system where the applications are executing. This article describes the nature of the various types of loops that can occur when running application programs within CICS, and offers some guidance on
the problem determination methodologies that can be applied towards resolving them.

CLASSIFICATION OF LOOPS WITHIN CICS

Certain loop conditions within CICS applications are detectable by the CICS system – others are not. This is because the nature of a loop may appear to be a valid sequence of events as far as CICS is concerned. For example, if an application were to repeatedly update a VSAM file, it is not possible for CICS to know whether or not this is an invalid loop condition. The application may be a batch-style non-terminal background task that handles a great many updates to files. It would be wrong for CICS to abend a task in such an apparent loop, since a repeated sequence of many tens of thousands of updates may well be normal business behaviour for such an application program.

There are two key factors when classifying the types of loops within CICS. One is the extent of the loop – in other words, is it unbounded or not (ie will it continue forever if unchecked)? The other is whether CICS is able to detect the looping condition and do something about it.

Loops are a repeated sequence of instructions. At their most basic form, they constitute a branch instruction back to a label just preceding the branch. This is an example of a very tight loop. If the branch has no conditional logic to guard its execution, it is an unconditional branch and hence the loop will continue forever unless checked by some external interruption. A variant of this is if the branch is guarded by some test which (if true) avoids the branch and so causes the executing program to end the looping sequence by itself. This may be a test on some external condition, or it might be when a count value has reached some predetermined amount. Since this simple coding variation provides the ability to complete the loop by itself, without the need for external interruption, it is easy to see how an apparently looping program may in fact just be performing a great many repetitive operations before completing. Conversely, it is also easy to see how an invalid limiting value for the count (say) may
cause the application to enter into an extended loop. CICS cannot analyse the business logic within an application to know whether a loop is planned or not; all it can do is detect certain conditions that may well be the result of an invalid loop and abend the running program’s task in such cases.

CICS has the ability to detect looping conditions that do not involve particular EXEC CICS commands within the scope of the loop. In the simple branch loop described above, there are no EXEC CICS commands within the loop; it just repeatedly iterates through the application code itself. This means that once the central processor’s thread of execution has entered such a tight loop within the application logic, CICS will not get invoked again while the loop is in progress. Since CICS has a cooperative programming model, it relies on application programs to periodically return control to it. This is typically by means of the EXEC CICS API, but also includes the use of XPI calls from GLUEs and DFHRMCAL calls from TRUEs. If an application program fails to invoke CICS by these means, the CICS control programs will not get control and so CICS functionality cannot be driven. In the case of a tight loop that does not invoke EXEC CICS commands, there is the need for CICS to be able to regain control in a pre-emptive manner. This is achieved by means of a timer mechanism, which posts CICS after a particular time period has elapsed since it was last able to perform any work. CICS is able to detect that a running task has entered what appears to be such a tight loop by the fact that control was not returned to CICS from the application within a certain elapsed period of time. This time period is specified by the ICVR SIT parameter, which states (in milliseconds) for how long a task should execute without returning control to CICS in this way. The default value for ICVR is 5,000 milliseconds; it can range from (decimal) 500 through to 2,700,000, in multiples of 500. CICS will round down specified values that are not multiples of 500. The ICVR value is the runaway task timeout value as used by transactions defined with RUNAWAY=SYSTEM in their transaction definitions. CICS can purge a task (that has not returned control to it via certain EXEC CICS commands) once this ICVR timeout period has elapsed. If
a transaction’s definition specified that the RUNAWAY parameter did not specify SYSTEM (but had a numeric timeout value instead), this is used as the runaway timeout period for that particular transaction.

Note: if ICVR is set to a value of zero (or if a transaction definition’s RUNAWAY parameter is set to zero), then runaway task detection is unavailable for such a task and it will not get purged by CICS if it enters an unbounded loop.

Note also that PTF UQ72475 (APAR PQ67892) modified CICS TS 2.2 to avoid runaway detection while executing a TRUE. This change is present at base code level in CICS TS 2.3. CICS now avoids abending tasks where a TRUE retains control for longer than the runaway limit. This change makes CICS work in a manner consistent with earlier releases.

There is another type of loop that is detectable by CICS (by means of the ICVR SIT parameter or the RUNAWAY value in a looping task’s transaction definition). This is a non-yielding loop, where the scope of the loop does contain one or more EXEC CICS commands. This means that part of the loop will iterate through CICS control programs. However, only certain operations within CICS will cause the runaway time value to be reset, so preventing CICS from deciding to abend a task that may be looping. These are operations that cause the task to be suspended for some reason, or ones that have to invoke CICS dispatcher services. If an EXEC CICS command does not involve such an operation, the runaway time value for the task will not be reset. This means that CICS is therefore able to detect the loop when the task’s runaway time period reaches the system’s ICVR value (or the task’s transaction definition’s RUNAWAY value).

Examples of EXEC CICS commands that may not cause a task’s runaway timeout value to be reset include ENQ, DEQ, and ASKTIME. Typically, such commands involve a small path length within CICS, and do not typically result in suspensions or CICS dispatcher services being invoked in order to honour them.
ACTION WHEN DETECTING LOOPS WITHIN CICS

Whether a loop is tight (with no EXEC CICS commands) or non-yielding (with EXEC CICS commands that do not reset the CICS runaway task timeout value), CICS is able to detect that the loop has led to the task executing for longer than is deemed acceptable by the ICVR SIT parameter or the RUNAWAY transaction definition setting. It can therefore take steps to rectify the situation. This results in the running task being abended by CICS with an AICA abend code. The task will enter dynamic transaction backout, its recoverable changes will be backed out, and the task will complete and leave the system.

An investigation of the abended task will concentrate on the transaction dump from the AICA abend. This will reveal the transid of the task, and the program that was running under CICS at the time it was abended. The trace entries within the CICS internal trace table may offer some clues as to the scope of the loop, whether it contained such non-yielding EXEC CICS commands. (Guidance on the use of trace analysis within a looping task is given later in the article.)

Another useful problem determination tool is the Program Status Word (PSW), which is formatted within a transaction dump for a task that abends because of some form of external interrupt (such as an abend ASRA for a program check, or AICA for an interruption by CICS). The address within the PSW will be pointing at the next instruction within the CICS address space that would have been executed by the central processor had the interruption not taken place. This will be pointing somewhere within the bounds of the loop logic inside the looping application. Using the dump, the load and entry points of the running program can be determined. The offset of the instruction addressed by the PSW can then be found by using the linkage editor map to determine the failing program within the load module and then subtracting its entry point address from the PSW next instruction address. Switching attention to a compiler listing of the looping application program at this point, the section of code that contains the loop can be analysed. The sixteen general purpose register values at the time of the interrupt (also formatted out by
the transaction dump) can be used to visually ‘dry-run’ through the code at this point and, hopefully, determine why the loop occurred.

An example transaction dump for an AICA abend is shown below:

```
CICSTEST   -- CICS TRANSACTION DUMP -- CODE=AICA TRAN=TCC1

PSW & REGISTERS AT TIME OF INTERRUPT
PSW                Ø78DØØØØ   95DØØ326   ØØØ2ØØØ1   ØØØØØØØØ
REGS 0-7           151Ø9614   151Ø858Ø   151Ø96C8   151Ø9678
                     15DØØ58   151Ø81DC   ØØØ7424Ø   14E52FFF
REGS 8-15          151Ø315Ø   151Ø94D8   15DØØ128   15DØØ252
                     15DØØ11C   151Ø84EØ   95DØØ3Ø6   ØØØØØØØØ

Transaction environment for transaction_number(0000029)
transaction_id(TCC1) orig_transaction_id(TCC1)
initial_program(PROGCC1 ) current_program(PROGCC1 )

PROGRAM INFORMATION FOR THE CURRENT TRANSACTION

Number of Levels 00000001

INFORMATION FOR PROGRAM AT LEVEL 00000001 of 00000001

Program Name     PROGCC1      Invoking Program CICS
Load Point       15D00000      Program Length        0001170
Entry Point      95D00020      Addressing Mode AMODE 31
Language Defined COBOL        Language Deduced COBOL II

The (edited) dump information shows that an abend AICA occurred for transaction identifier TCC1, running on the CICSTEST CICS region. The PSW address is 95D00326. (Note: the top bit in the address is actually the addressing mode indicator bit. Since this is set to 1, the application was executing in amode 31 at the time of the interrupt. Had the top bit been set to 0, this would have signified amode 24. The actual address is shown by the remaining 31 bits within the word, ie 15D00326). The task number was 00029, and the running program at the time of the
abend was PROGCC1. This was at link level 1, invoked by CICS and not an EXEC CICS LINK command. The dump formatter shows the load point of PROGCC1 to be 15D00000, and the entry point to be 95D00020 (remember, the top bit is not part of the real address, so the actual entry point is 15D00020). The hex 20 byte difference between the load point and entry point values is caused by the CICS command level stub, link-edited to the application program.

A subtraction of the entry point address from the PSW address shows that the next instruction that would have been executed within PROGCC1 (had the interrupt and abend AICA not occurred) would have been at offset hex 306 into the program. 15D00326 - 15D00020 = 306. An analysis of the application program listing for PROGCC1 (below) shows that this is within the confines of an unbounded loop:

```
000057 PERFORM
0002F2
000059 MOVE
0002F2 5820 912C L 2,300(0,9)
0002F6 D219 2000 A018 MVC 0(26,2),24(10) MSG1
0002FC 9240 201A MVI 26(2),X'40' MSG1+26
000300 D220 2018 201A MVC 27(33,2),26(2) MSG1+27
000306 47F0 B0C0 BC 15,192(0,11) GN=8(0002F2)
```

There is an unconditional branch instruction (opcode 47, condition F) at offset hex 306, which returns control back to the PERFORM verb at offset hex 2F2. Since the intervening code has no means of exiting this path (it just contains load and move instructions), the application will loop forever unless ended by an external means (such as the abend AICA).

**CONSIDERATIONS FOR ABENDING A ‘LOOPING’ TASK**

Abending the looping task will resolve the run-time effects of the loop, which would typically have included poor performance or an increase in CPU usage while the loop was in progress. However, there are some considerations to be made when such a looping task is abended. For example, was the loop actually invalid (ie could the abended task have in fact been an example
of a long-running background-style task that was just performing a great many repetitive operations, as described earlier in this article)? If this is the case, it may be advisable to review the CICS system’s ICVR value (or perhaps more pertinent to adjust the RUNAWAY parameter value for the abended task’s transaction definition). This would avoid CICS abending such tasks performing valid long-running work in the future.

Another consideration is the length of time that the dynamic transaction backout of an abended looping task can take to perform. If the task had been repeatedly updating a recoverable resource (or resources), there could be a great many before-images to be reinstated from the CICS system log as part of the backout operations. This would mean that the act of abending a looping task can itself take a considerable time to achieve. The corollary to this is when a CICS system is cancelled while such a long-running series of recoverable changes are being made by a looping task and has to be emergency restarted as a result. The emergency restart can take a long time to complete because CICS needs to traverse back along the system log stream in order to reconstruct the task’s unit of work (UOW) environment before the inflight task can be backed out by CICS. This is indicated by a series of DFHLG-prefixed messages during the restart operation. Message DFHLG0745 shows when the backwards scan of the system log has begun. DFHLG0747 messages are then periodically issued by CICS to reveal the number of log records that have been processed so far by the restart. These messages are issued when a specific number of log records have been retrieved from the CICS system log; this number is the greater of half of the CICS AKPFREQ value or (decimal) 500. During the backwards scan of the log, a DFHLG0748 message may be issued. This means that CICS was able to optimize the backwards scanning process at this stage. Eventually, at the completion of the backwards log scan, CICS will issue message DFHLG0749, indicating that all the log records of interest have been retrieved from the system log, and any UOWs that needed reconstruction have been rebuilt from their logged data. Once control has been given to CICS (message
ANALYSING YIELDING LOOPS WITHIN CICS

The above example (or a loop containing recoverable work within a CICS system) is an example of a yielding loop. This defines a loop that does involve the suspension of the task as part of the loop's scope of operations, or else involves the use of CICS dispatcher services. A yielding loop means that the looping task’s runaway timeout value is reset by CICS and so never reaches the purgeability threshold specified by the ICVR or RUNAWAY settings. The result is that CICS does not abend the task, since it is unaware that a loop has occurred. In the case of a recoverable operation against a resource such as a VSAM file, CICS has to log the before-image of the data to the DFHLOG system log as part of the EXEC CICS command, in order to be able to undo the recoverable operation if the task were later to abend or issue an EXEC CICS SYNCPNT ROLLBACK command. Since the updates involve hardening of log data to the MVS System Logger, and I/O of the updated records to VSAM, CICS dispatcher services will be resetting the runaway time for the executing task.

Although a yielding loop is not detectable by CICS, and so will not result in the looping task being abended with an AICA abend code, the symptoms of the loop will typically result in the task’s anti-social behaviour being detected by the users of the system, monitoring tools, or systems management software. Symptoms can include poor response times, transaction throughput degradation, increased waits because of enqueues, sympathy sickness with other CICS regions, and hangs for terminal end users. Once this has been detected, the looping task needs to be identified if this is not yet known. Action can then be taken against the task (eg by purging it from the system to remove the loop). Before it is purged, however, it is sensible to try to capture a short piece of trace activity from the running task, for the subsequent problem determination of the cause of the yielding loop. Since the purge may not result in a CICS transaction or system dump being generated (containing the looping application program’s
trace entries within the CICS internal trace table), a short section of auxiliary trace recording could be taken before the task is removed from the system. CETR selective tracing can be used to trace data from just one particular transaction or terminal. See the *Transaction and Terminal Trace* display, produced by pressing PF5 on the main CETR display, for more details about this.

Since the looping task was not abended by CICS, it contains commands that yield control to CICS dispatcher services. As such, it is sensible to concentrate on the EXEC CICS trace entries produced by the looping task when analysing the trace. There is no point in looking at any other trace data at this stage.

An example of the trace from a looping task (number 00119) is shown below:

```
ØØ119 QR  AP 00E1 EIP ENTRY ASKTIME-ABSTIME 001404A8
...y, 08004A02 ...

ØØ119 QR  AP 00E1 EIP EXIT ASKTIME-ABSTIME OK 00000000
.....,00004A02 ...

ØØ119 QR  AP 00E1 EIP ENTRY FORMATTIME 001404A8
...y,08004A04 ...

ØØ119 QR  AP 00E1 EIP EXIT FORMATTIME OK 00000000
.....,00004A04 ...

ØØ119 QR  AP 00E1 EIP ENTRY LINK 001404A8
...y,08000E02 ....

ØØ119 QR  AP 00E1 EIP ENTRY HANDLE-CONDITION 00140648
.....,08000204 ....

ØØ119 QR  AP 00E1 EIP EXIT HANDLE-CONDITION OK 00000000
.....,00000204 ....

ØØ119 QR  AP 00E1 EIP ENTRY WRITEQ-TS 00140648
.....,08000A02 ....

ØØ119 QR  AP 00E1 EIP EXIT WRITEQ-TS OK 00000000
.....,00000A02 ....

ØØ119 QR  AP 00E1 EIP ENTRY READQ-TS 00140648
.....,08000A04 ....

ØØ119 QR  AP 00E1 EIP EXIT READQ-TS QIDERR 00000000
```
& 0320A04

00119 QR AP 00E1 EIP ENTRY HANDLE-CONDITION 00140648

00119 QR AP 00E1 EIP EXIT HANDLE-CONDITION OK 00000000

00119 QR AP 00E1 EIP ENTRY RETURN 00140648

00119 QR AP 00E1 EIP EXIT LINK OK 00000000

00119 QR AP 00E1 EIP ENTRY DELAY 001404A8

00119 QR AP 00E1 EIP EXIT DELAY OK 00000000

00119 QR AP 00E1 EIP ENTRY ASKTIME-ABSTIME 001404A8

00119 QR AP 00E1 EIP EXIT ASKTIME-ABSTIME OK 00000000

00119 QR AP 00E1 EIP ENTRY FORMATTIME 001404A8

00119 QR AP 00E1 EIP EXIT FORMATTIME OK 00000000

00119 QR AP 00E1 EIP ENTRY LINK 001404A8

00119 QR AP 00E1 EIP EXIT HANDLE-CONDITION 00140648

00119 QR AP 00E1 EIP EXIT HANDLE-CONDITION OK 00000000

00119 QR AP 00E1 EIP ENTRY WRITEQ-TS 00140648

00119 QR AP 00E1 EIP EXIT WRITEQ-TS OK 00000000

00119 QR AP 00E1 EIP ENTRY READQ-TS 00140648

The trace has been formatted using selective trace options for the DFHTU620 off-line auxiliary trace formatting utility. The DFHTU620 job was submitted with the following selective trace print parameters:

```
ABBREV, TASKID=(00119), TYPETR=(AP00E1)
```

This causes only those EXEC CICS trace entries for task 00119 to be returned. (AP00E1 is the unique trace point identifier for EXEC CICS ENTRY and EXEC CICS EXIT trace points.) Abbreviated trace (one line per entry) is chosen because it makes analysis of the overall flow of events easier to read.

The trace shows a repeating sequence of EXEC CICS commands, issued by two applications running within the task. As can easily be seen, an EXEC CICS READQ of a temporary storage queue fails with a QIDERR response. The task has issued an EXEC CICS HANDLE CONDITION and so this failure is handled and the task allowed to continue. Another EXEC CICS HANDLE CONDITION is issued, and the flow of events repeats itself.

Just by examining this selective series of trace entries, it is possible to determine a number of facts about the looping environment. There are two programs involved; notice the use of
the EXEC CICS LINK and EXEC CICS RETURN commands to traverse between them. Note also that the Field A value varies between these two programs; it changes from 001404A8 to 00140648 and back again. Field A on an EXEC CICS ENTRY trace entry denotes the address of the high-level language save area for the application program that issued the command. For PL/I this is the DSA, for Assembler the DFHEISTG, for COBOL the working storage.

Selecting only the EXEC CICS trace entries makes it very easy to spot atypical events within the bounds of the loop; in this case, it is the QIDERR response rather than OK. It also allows a quick identification of the scope of the loop itself, including which commands are contained within it. In this example, it can be seen that the pattern of commands repeats itself after the EXEC CICS DELAY.

At this stage it would be useful to further analyse the failing EXEC CICS command. This can be achieved by using a different trace selection. If DFHTU620 is executed with:

**SHORT, TASKID=(00119), TYPETR=(AP00E1)**

then the SHORT formatting option will return additional information to that given by the abbreviated trace entries. In particular, attention can be paid to the return address of the commands. This is the RET field in the trace entries, and shows the address within the CICS address space of the next instruction to be executed after the flow of control has returned to the application program from CICS, on completion of the EXEC CICS command.

An example of the use of the SHORT trace entry formatting option is shown below:

```
00119 QR AP 00E1 EIP ENTRY WRITEQ-TS FIELD-A(00140648) FIELD-B(08000A02)

RET-500C226E

16:19:25.254436257 =000414=
```

```
00119 QR AP 00E1 EIP EXIT WRITEQ-TS OK FIELD-A(00000000) FIELD-B(00000A02)
```
For ease of reference, this shows just the two temporary storage commands being traced. For successful EXEC CICS commands, the return address values are consistent between the EIP ENTRY and EIP EXIT trace points. For those commands that are handled by an EXEC CICS HANDLE CONDITION environment, however, the return address will be different on the EIP EXIT trace point. This is because the address now denotes the place where the flow of control will return when the condition is handled; ie the handle routine established by the earlier EXEC CICS HANDLE CONDITION command. In the example above, it can be seen that the EXEC CICS READQ command has a different RET address on exit to the one that it had when the command entered CICS. This indicates that the QIDERR condition was handled.

This example demonstrates one possible cause of a loop within CICS (a handled failure condition that allows the failing environment to retry the request). Of course, each loop situation will have its own cause and will generate its own particular EXEC CICS trace entry information. However, the same basic principles apply when investigating the problem. What are the EXEC CICS commands being issued? What is the scope of the loop (that is, the pattern of the repeating commands)? What are the programs that constitute the looping environment? Are any of the commands failing? Are any handle routines in effect, or has NOHANDLE been used? Either could inadvertently lead to a loop occurring if conditions or abends are not handled properly. Also, are any
unexpected commands contained within the scope of the loop? In the example given above, an EXEC CICS DELAY command can be seen. It may be appropriate to the application’s logic; conversely, it may be redundant coding from the original test phase of the application program. An investigation of the commands within the scope of the loop should be performed to confirm their attributes and relevance to the logic at the time the loop was being executed.

In addition to focusing on just the EXEC CICS trace entries issued by a looping task, other types of trace entry may warrant attention too. For example, a short on storage (SOS) condition produced by a looping application program will generate CICS Storage Manager trace entries of interest. As such, it is worth investigating these too, with DFHTU620 option:

ABBREV, TASKID=(123), TYPETR=(AP0OE1, SM0C01, SM0C02)

REFERENCES

The CICS Resource Definition Guide has further information on the RUNAWAY attribute specified for transaction definitions. The CICS System Definition Guide gives details about use of the ICVR system initialization parameter to control runaway detection within CICS.

CONCLUSION

I hope this article has helped give a background to the types of loop that may be encountered when running applications within CICS and offered some guidance on ways of approaching them from the problem determination perspective.

Andy Wright (andy_wright@uk.ibm.com)
CICS Change Team
IBM (UK) © IBM 2004
Helpful exit for shutdown assistant users – part 2

This month we conclude the code for the shutdown assistant exit.

CSSHUT

*ASM XOPTS(CICS, SP)

CSSHUT TITLE 'PERFORM CICS-SHUTDOWN-PROCESSING
SPACE 1

*---------------------------------------------------------------------*
*                C S S H U T                                          *
*---------------------------------------------------------------------*
*        THIS ROUTINE WILL BE AVAILABLE WITH THE                      *
*        TRANSACTION-ID 'SHUT'.                                       *
*                                                                     *
*        THE TA 'SHUT' MUST BE STARTED VIA CONSOLE:                   *
*                                                                     *
*        (/)F cicsname, SHUT                                         *
*                                                                     *
*---------------------------------------------------------------------*

EJECT

*---------------------------------------------------------------------*
*        PROGRAM CHANGES                                              *
*---------------------------------------------------------------------*

*      INCLUDE++ CSCWAA       INCLUDE CICS NLV CWA ASSEMBLER
EJECT

*---------------------------------------------------------------------*
*        C I C S       C W A - B E R E I C H                         *
*        INCLUDE-ELEMENT FUER ASM PROGRAMME CSCWAA
*---------------------------------------------------------------------*

ADDRESSIERUNG: EXEC CICS ADDRESS CWA(CWAPTR)
ACHTUNG : KEINE AENDERUNG ERLAUBT - NUR LESEN
CWAPTR MUSS NOCH DEFINIERT WERDEN

SPACE 3
USING CWADSECT, CWAPTR
SPACE 3

CWADSECT DSECT
SPACE 3

CWAAREA DS 0CL1536 CWA-BEREICH
SPACE 1

CWAEEYCA DC CL4' ' EYECATCHER
CWAWSYS DC CL4' ' ZUGEORDNETE VSAM-SYSID (CWACICID)
CWATSYS DC CL4' ' ZUGEORDNETE TERM-SYSID (CWACICID)
CWAWSYSD DC CL4' ' ORIGINAL SYSTEM-ID
CWAPPLID DC CL8' ' ORIGINAL APPLICATION-ID
CWANCVT DC AL4(Ø) POUN TER NLV-CVT
CWA_PTR_CUATR DC AL4(Ø) ADRESSE D. CUA-TRANSAKTIONSTABELLE
CWA_CI CSLEVEL DC OC4' ' CICS-LEVEL '0311' OR '0330'
CWA_CI CSLEV DC CL1' ' CICS-LEVEL
CWA_CI CSVER DC CL1' ' CICS-VERSION
CWA_CI CSREL DC CL1' ' CICS-RELEASE
CWA_CI CSMOD DC CL1' ' CICS-MODIFICATION
CWA_CMFSTOP DC PL4' Ø' STOP-TIME FOR CMF-EVENTS HHMMSSTC
DS XL56Ø .......... FREI ...........
DS XL12Ø .......... FREI ...........
DS XL22 .......... FREI ...........
CWA_CI CTSX DC CL4' ' CICS-ID-BESCHREIBUNG
CWA_CI CID DC CL1' ' CICS-ID
CWA$PROD EQU C'P' .. PROD
CWA$TEST EQU C'T' .. TEST
CWA$VPRD EQU C'V' .. VORPROD
CWA$SYST EQU C'S' .. SYSTEM-CICS
CWA_CI CNR DC CL1' ' CICS-NR
CWA$TERM EQU C'T' .. TERMINAL
CWA$VSAM EQU C'V' .. DATASET VSAM
CWA$PAIS EQU C'P' .. PAISY
CWA$ODM EQU C'O' .. ODM
CWA$INFO EQU C'I' .. INFO
CWA$APPL EQU C'O' .. APPLICATION Ø0-Ø9
* $APPL EQU ??? .. APPLICATION A-C
* $APPL EQU ??? .. APPLICATION E-O
* $APPL EQU ??? .. APPLICATION Q-S
* $APPL EQU ??? .. APPLICATION U-Z
CWA$DATUM DC CL8' ' DATUM FORMAT TT.MM.JJ
CWA$MT DC CL6' ' DATUM TTMMJJ
CWA$MT DC PL4' Ø' DATUM ØTTMMJJC
CWA$MT DC CL6' ' DATUM JJJMMT
CWA$MT DC PL4' Ø' DATUM ØJJMMTTC
CWA$MT4 DC CL8' ' DATUM TTMMJJJ
CWA$MT4 DC PL5' Ø' DATUM ØTTMMJJJ
CWA$MT4 DC CL8' ' DATUM JJJJJM
CWA$MT4 DC PL5' Ø' DATUM ØJJJJMM
CWA$MT3 DC CL6' ' DATUM MMJ
CWA$MT3 DC PL3' Ø' DATUM ØMMJJC
CWA$MT3 DC CL6' ' DATUM JJJMM
CWA$MT3 DC PL3' Ø' DATUM ØJJJMMC
CWA$MT4 DC CL6' ' DATUM MMJJ
CWA$MT4 DC PL4' Ø' DATUM ØMMJJJC
CWA$MT4 DC CL6' ' DATUM JJJJM
CWA$MT4 DC PL4' Ø' DATUM ØJJJM
CWA$MT3 DC CL5' ' DATUM TTTJ
CWA$MT3 DC PL3' Ø' DATUM TTTJJ
CWA$MT3 DC CL5' ' DATUM JJTT
CWA$MT3 DC PL3' Ø' DATUM JJTTT
CWA$MT3 DC CL7' ' DATUM TTTJJJ
CWA_FTT_TEXT DS CL207 BE SCHREIBUNG DER AKTION
CWAFTTDSECT EQU *
CWAFTTANZAHL EQU 30 ANZAHL TABELLENEINTRAEGE FTT
SPACE 2
*---------------------------------------------------------------------*
* ENDE DER DSECT FUER FUNKTIONSTASTENTABELLE *
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
* BEGINN DER DSECT FUER AKTIONSNAMETABELLE *
* ADRESSIERUNG UEBER "CWA_PTR_ANT" *
*---------------------------------------------------------------------*
CWAANTDSECT DSECT
CWA_ANT_HILFE DS CL16 HILFETEXT
CWA_ANT_TASTEN DS CL16 ANZEIGE DER TASTENBELEGUNG
CWA_ANT_AUSGANG DS CL16 BEENDEN EINER FUNKTION
CWA_ANT_REFRESH DS CL16 WIEDERHERSTELLEN
CWA_ANT_UPDATE DS CL16 DATEN SPEICHERN
CWA_ANT_RUECKWAERTS DS CL16 RUECKWARTS BLATTERN
CWA_ANT_VORWAERTS DS CL16 VORWARTS BLATTERN
CWA_ANT_AKTION DS CL16 AKTIVIERUNG Aktionbar
CWA_ANT_UNTERBRECHEN DS CL16 VORGANGSUNTERBRECHUNG
CWA_ANT_ABBRUCH DS CL16 ABBRUCH
CWA_ANT_EINSTIEG DS CL16 ZURUECK ZUM EINSTIEGBILD
CWA_ANT_AUSWAHL DS CL16 ZURUECK ZUM AUSWAHLBILD
CWA_ANT_SICHERN DS CL16 EINFRIEREN VON DATEN
CWA_ANT_LINKS DS CL16 BLATTERN NACH LINKS
CWA_ANT_RECHTS DS CL16 BLATTERN NACH RECHTS
CWA_ANT_ANFANG DS CL16 ANZEIGE DER ERSTEN SEITE
CWA_ANT_SCHLUSS DS CL16 ANZEIGE DER LETZTEN SEITE
CWA_ANT_ABMELDEN DS CL16 ZSS-ABMELDUNG
CWA_ANT_DRUCKEN DS CL16 DRUCKEN (PA2)
CWA_ANT_LOESCHEN DS CL16 LOESCHEN BILDSCHIRM
CWA_ANT_DATENFREIGABE DS CL16 DATENFREIGABE
CWA_ANT_HILFE_ANLEGEN DS CL16 BOSHELP HILFE ANLEGEN
CWA_ANT_SUCHEN DS CL16 SUCHEN
CWA_ANT_EURODM DS CL16 UMRECHNEN EURO/DM
CWAANTDSECTE OQU *
*---------------------------------------------------------------------*
* ENDE DER DSECT FUER AKTIONSNAMETABELLE *
*---------------------------------------------------------------------*
END INCLUDE++
EJECT

*---------------------------------------------------------------------*
* NETVIEW-COMMAREA *
*---------------------------------------------------------------------*

SPACE
NETVIEW_COMMAREA DSECT
NETVIEW_COMMAREA_LENGTH DC AL2(NETVIEW_COMMAREA_T_LENGTH)
NETVIEW_COMMAREA_TEXT DC CL8' NETVIEW'

NETVIEW_COMMAREA_T_LENGTH EQU *NETVIEW_COMMAREA_EJECT

* WORKFIELDS IN EIS

* COMMAREA DEFINITION START

* CMON DS OCL10 23-01 . COMMAREA FOR PROGRAM CSMON01
* CMONLEN DS AL2 23-01 . COMMAREA LENGTH
* CMONFUN DS CL1 23-01 . COMMAREA FUNCTION-CODE
* CMONRET DS XL1 23-01 . COMMAREA RETURN-CODE
* CMONPNR DS CL6 23-01 . COMMAREA PERSONAL-NUMBER
* CMONLGTH DS AL2 23-01 COMMAREA-LENGTH FOR E.C. LINK
* COMMAREA DEFINITION END

EIS_RESP DC F'0' RESPONSE-FIELD FROM E.C. REQ.
EISRESOURCE DC CL4' ' RESOURCE-FIELD FOR ENQ
EIS_RTRANSID DC CL4' ' RTRANSID FIELD FROM RETRIEVE
EIS_RTERMID DC CL4' ' RTERMID FIELD FROM RETRIEVE
EIS_QUEUE DC CL8' ' QUEUE FIELD FROM RETRIEVE
EIS_LENGTH DC AL2(Ø) LENGTH FIELD FROM RETRIEVE
EIS_MSG DC CL100' ' MESSAGE FIELD FOR WTO COMMAND

EXEC CICS ADDRESS CWA (CWAPTR)

OC EIBTRMI D, EIBTRMI D TRANSID-START FROM A TERMINAL ?
BNZ NORMAL YES...PERFORM SHUTDOWN

EXEC CICS RETRIEVE SET (COMPTR)
LENGTH (EIS_LENGTH)
RTRANSID (EIS_RTRANSID)
RTERMID (EIS_RTERMID)
QUEUE (EIS_QUEUE)
RESP (EIS_RESP)

CLC EIS_RESP, DFHRESP(NORMAL) ANY ERRORS DETECTED ?
BNE NETVIEW_MSG13 NO...ISSUE ERROR-MESSAGE

CLC EIS_RTRANSID, =C'SHUT' VALID COMMUNICATION ?
BNE NETVIEW_MSG13 NO...ISSUE ERROR-MESSAGE
CLC EIS_RTERMID, =C'CN99' VALID COMMUNICATION ?
BNE NETVIEW_MSG13 NO...ISSUE ERROR-MESSAGE
CLC EIS_QUEUE, =CL8'NETVIEW' VALID COMMUNICATION ?
BNE NETVIEW_MSG13 NO...ISSUE ERROR-MESSAGE
* CLI  CMONRET, Ø
  ANY ERRORS OCCURRED IN PROG. CSMONØ1
* BE  STAT#OK
  NO...SKIP ABOUT ERROR-MESSAGE
* STAT#ERR DS ØH
* MVC EIS_MSG, BLANK
* MVC EIS_MSG(L' MSGØ04), MSGØ04
* BAL R8, MESSAGE
  WRITE MESSAGE
* STAT#OK DS ØH
* EXEC  CICS HANDLE ABEND RESET
  EJECT
* NLV-NTB-MONITOR PROCESSING
* SPACE
CLI  CWACICNR, CWA$TERM
  TERMINAL-CICS INDICATOR ?
BE  NTB#OK
  YES...DON'T CALL NTB-MONITOR
CLI  CWACICNR, CWA$VSAM
  DATASET-CICS VSAM INDICATOR ?
BE  NTB#OK
  YES...DON'T CALL NTB-MONITOR
CLI  CWACICNR, CWA$PAIS
  PAISY-CICS INDICATOR ?
BE  NTB#OK
  YES...DON'T CALL NTB-MONITOR
CLI  CWACICNR, CWA$ODM
  ODM-CICS INDICATOR ?
BE  NTB#OK
  YES...DON'T CALL NTB-MONITOR
EXEC  CICS HANDLE ABEND LABEL(NTB#NOK)
EXEC  CICS LINK PROGRAM ('CSMONØ2')
  *  RESP  (EIS_RESP)
* CLC  EIS_RESP, DFHRESP(NORMAL)
  ANY ERRORS DETECTED ?
BE  NTB#OK
  NO...SKIP ABOUT ERROR-MESSAGE
* NTB#NOK DS ØH
* MVC EIS_MSG, BLANK
* MVC EIS_MSG(L' MSGØ05), MSGØ05
* BAL R8, MESSAGE
  WRITE MESSAGE
NTB#OK DS ØH
* EXEC  CICS HANDLE ABEND RESET
  EJECT
* DATAPACKER-II-MONITOR-PROCESSING
* SPACE
* DPII#PLT DS ØH
* EXEC  CICS HANDLE ABEND LABEL(DPII#ERR)
* EXEC  CICS LINK PROGRAM ('DPIIZ8ØØ')
  *  RESP  (EIS_RESP)
* CLC  EIS_RESP, DFHRESP(NORMAL)
  ANY ERRORS DETECTED ?
BE  DPII#OK
  NO...BYPASS ERROR-MESSAGE
* DPII#ERR DS ØH
* MVC EIS_MSG, BLANK
* MVC EIS_MSG(L’MSGØØ6), MSGØØ6
* BAL R8, MESSAGE       WRITE MESSAGE
* DPII#OK DS ØH
* EXEC CICS HANDLE ABEND RESET
* EJECT
* EJECT
*
* STOP DB2-ATTACHMENT- FACILITY IF IT’S CICS/ESA 4.1 *
* *
*---------------------------------------------------------------------*
*              STOP DB2-ATTACHMENT-FACILITY IF IT’S CICS/ESA 4.1      *
*---------------------------------------------------------------------*
* SPACE
* CLI CWA_CICSVER, C’4’ IS CICS/ESA VERSION 4 ACTIVE ?
BNE INFOSETU NO: DON’T STOP DB2
MVC EIS_MSG, BLANK
MVC EIS_MSG(L’MSGØØ7), MSGØØ7
BAL R8, MESSAGE       WRITE MESSAGE
EXEC CICS HANDLE ABEND LABEL(DB2#ERR)
EXEC CICS LINK PROGRAM (‘CSMONØ3’)                           
   RESP (EIS_RESP)
*

* CLI EIS_RESP, DFHRESP(NORMAL) ANY ERRORS DETECTED ?
BNE DB2#ERR YES...ISSUE ERROR-MESSAGE
MVC EIS_MSG, BLANK
MVC EIS_MSG(L’MSGØØ8), MSGØØ8
BAL R8, MESSAGE       WRITE MESSAGE
B DB2#OK DB2-ATTACHMENT SUCCESSFULLY STOPPED
*

DB2#ERR DS ØH
MVC EIS_MSG, BLANK
MVC EIS_MSG(L’MSGØØ9), MSGØØ9
BAL R8, MESSAGE       WRITE MESSAGE
EJECT 1

* INFO-CICS SETUP *
* *
*---------------------------------------------------------------------*
* INFOSETU DS ØH
*
* CLI CWA_INFOCICS, CWA_INFOCICS_Y IS IT ALREADY INFO-CICS ?
BE INFO#OK YES...BYPASS INFO-CICS SETUP
*

* CLI CWACICID, CWA$PROD PROD-CICS INDICATOR ?
BE INFO#SET YES...INFO-CICS SETUP
CLI CWACICID, CWA$SYST SYSTEM-CICS INDICATOR ?
BNE INFO#OK NO...BYPASS INFO-CICS SETUP
INFO#SET DS ØH
CLI CWACICNR, CWA$TERM TERMINAL-CICS INDICATOR ?
BNE INFO#OK NO...BYPASS INFO-CICS SETUP
MVC EIS_MSG, BLANK
MVC EIS_MSG(L’MSGØØ10), MSGØØ10

BAL R8, MESSAGE WRITE MESSAGE
EXEC CICS HANDLE ABEND LABEL(INFO#ERR) *  
EXEC CICS LINK PROGRAM ('CSI INFO')  
  RESP  (EIS_RESP) *  
  ------------------------------------  
  CLC EIS_RESP, DFHRESP(NORMAL) ANY ERRORS DETECTED ?  
  BNE INFO#ERR YES... ISSUE ERROR-MESSAGE  
  MVC EIS_MSG, BLANK  
  MVC EIS_MSG(L' MSG011), MSG011  
  BAL R8, MESSAGE WRITE MESSAGE  
  B INFO#OK INFO-CICS SETUP SUCCESSFULL  
  ------------------------------------  
  INFO#ERR DS ØH  
  MVC EIS_MSG, BLANK  
  MVC EIS_MSG(L' MSG012), MSG012  
  BAL R8, MESSAGE WRITE MESSAGE  
  INFO#OK DS ØH  
  EXEC CICS HANDLE ABEND RESET  
  EJECT  
*  
*                              INTERTEST-STOP PROCESSING  
*  
*  SPACE  
EXEC CICS INQUIRE PROGRAM('IN25PGM2')  
  RESCOUNT(RESCOUNT)  
  RESP  (EIS_RESP) *  
  ------------------------------------  
  CLC EIS_RESP, DFHRESP(PGMIDERR)  
  BE IN25#OK PGMIDERR: INTERTEST IS NOT ACTIVE  
  CLC EIS_RESP, DFHRESP(NORMAL) ANY OTHER ERRORS DETECTED ?  
  BNE IN25#NOK NO... BYPASS ERROR-MESSAGE *  
  CLC RESCOUNT, =F'Ø' IS INTERTEST ACTIVE ?  
  BE IN25#OK NO... DON'T CALL INTERTEST-SHUTDWON  
*  
EXEC CICS HANDLE ABEND LABEL(IN25#NOK)  
*  
EXEC CICS LINK PROGRAM ('IN25PLTE')  
  RESP  (EIS_RESP) *  
  ------------------------------------  
  CLC EIS_RESP, DFHRESP(NORMAL) ANY ERRORS DETECTED ?  
  BE IN25#OK NO... BYPASS ERROR-MESSAGE  
*  
IN25#OK DS ØH  
  MVC EIS_MSG, BLANK  
  MVC EIS_MSG(L' MSG015), MSG015  
  BAL R8, MESSAGE WRITE MESSAGE  
  IN25#OK DS ØH  
  EXEC CICS HANDLE ABEND RESET  
  EXEC CICS DELAY INTERVAL(1)  
  EJECT  

* -------------------------------------------------------------------- *
*              IMS-STATISTIC                                          *
* --------------------------------------------------------------------- *

EXEC CICS HANDLE ABEND LABEL(IMSS#NOK)
CLI CWACICNR,C'I' IS IT INFO-CICS
BE IMSS#PRO YES.. WRITE STATISTIC
CLI CWACICNR,C'Ø' IS IT AN APPL.-CICS
BNL IMSS#PRO YES.. WRITE STATISTIC
B IMSS#OK NO.. DON'T WRITE STATISTIC
IMSS#PRO DS ØH
EXEC CICS LINK PROGRAM ('CSMON05')

RESP (EIS_RESP)

* -------------------------------------------------------------------- *
*                  CLC EIS_RESP,DFHRESP(NORMAL) ANY ERRORS DETECTED ? *
BE IMSS#OK NO.. BYPASS ERROR-MESSAGE
* -------------------------------------------------------------------- *

IMSS#NOK DS ØH
MVC EIS_MSG,BLANK
MVC EIS_MSG(L'MSG016),MSG016
BAL R8,MESSAGE WRITE MESSAGE
IMSS#OK DS ØH
EXEC CICS HANDLE ABEND RESET
EJECT

* -------------------------------------------------------------------- *
* STOP CAU IN SCICS, TCICS AND VCICS *
* -------------------------------------------------------------------- *

SPACE 1
* CLI CWACICID,CWA$SYST SYSTEM-CICS INDICATOR ?
* BE CAU_STOP YES.. STOP CAU
* CLI CWACICID,CWA$VPRD VORPROD-CICS INDICATOR ?
* BE CAU_STOP YES.. STOP CAU
* CLI CWACICID,CWA$TEST TEST-CICS INDICATOR ?
* BE CAU_STOP YES.. STOP CAU
B CAU_END NO.. BYPASS CAU STOP
CAU_STOP DS ØH

EXEC CICS START TRANSID ('CAFF')
FROM (CAUSTOP)
RESP (EIS_RESP)

* -------------------------------------------------------------------- *
* CAU_DELAY DS ØH
EXEC CICS DELAY INTERVAL (ØØØØØ1)
* -------------------------------------------------------------------- *

MVC EIS_RESOURCE,=C'CAF8'
EXEC CICS ENQ RESOURCE (EIS_RESOURCE)
LENGTH (=Y(L' EIS_RESOURCE))
NOSUSPEND
RESP (EIS_RESP)
* .................................................................
  CLC  EIS_RESP, DFHRESP(ENQBUSY)
  BE   CAU_DELAY
* .................................................................
  MVC  EIS_RESOURCE, =C'CAF'B'
  EXEC CICS DEQ RESOURCE (EIS_RESOURCE)    *
      LENGTH (=Y(L'EIS_RESOURCE))          *
      RESP (EIS_RESP)
  CAU_END  DS  ØH
  *   B  SHUTNO      ......NORMAL    SHUTDOWN
  *   EJECT
  *---------------------------------------------------------------------*
  *              DECIDE THE PROPER CICS-SHUTDOWN                        *
  *              -------------------------------                        *
  *---------------------------------------------------------------------*
  *        SPACE
  *  CLI   CWACICNR, CWA$TERM   TERMINAL-CICS INDICATOR ?
  BE    SHUTYES     YES...NORMAL    SHUTDOWN
  B     SHUTNO
  *  CLI   CWACICNR, CWA$VSAM   DATASET-CICS INDI CATOR VSAM ?
  *    BE    SHUTNO     YES...NORMAL    SHUTDOWN
  *  CLI   CWACICNR, CWA$PAIS   APPLICATION-CICS PAISY INDICATOR ?
  *    BE    SHUTNO     YES...NORMAL    SHUTDOWN
  *    B    SHUTYES    NO....IMMEDIATE SHUTDOWN
  EJECT
  *---------------------------------------------------------------------*
  *              PERFORM SHUTDOWN NORMAL ( WITHOUT IMMEDIATE )            *
  *              -------------------------------------------            *
  *              ACTIVE FOR DATASET-CICS                                *
  *              ACTIVE FOR APPLICATION-PAISY-CICS                      *
  *---------------------------------------------------------------------*
  SPACE 1
  SHUTNO  DS  ØH
  MVC  EIS_MSG, BLANK
  MVC  EIS_MSG(L'MSG003), MSG003
  BAL  R8, MESSAGE       WRITE MESSAGE
  EXEC CICS HANDLE ABEND LABEL(CEMN#NOK)
  *
  EXEC CICS PERFORM SHUTDOWN
      RESP (EIS_RESP)
  *
  CLC  EIS_RESP, DFHRESP(NORMAL)  ANY ERRORS DETECTED ?
  BE   RETURN   NO...BYPASS ERROR-MESSAGE
  *
  CEMN#NOK  DS  ØH
  EXEC CICS HANDLE ABEND RESET
  MVC  EIS_MSG, BLANK
  MVC  EIS_MSG(L'MSG001), MSG001
  BAL  R8, MESSAGE       WRITE MESSAGE
  B   RETURN
  EJECT
PERFORM SHUTDOWN IMMEDIATE

ACTIVE FOR TERMINAL-CICS
ACTIVE FOR APPLICATION-CICS (WITHOUT PAISY)

SPACE 1

SHUTYES DS ØH
MVC EIS_MSG, BLANK
MVC EIS_MSG(L'MSGØ02),MSGØ02
BAL R8, MESSAGE WRITE MESSAGE
EXEC CICS HANDLE ABEND LABEL(CEMI#NOK)
EXEC CICS PERFORM SHUTDOWN IMMEDIATE
* RESP (EIS_RESP)

CLC EIS_RESP,DFHRESP(NORMAL) ANY ERRORS DETECTED ?
BE RETURN NO...BYPASS ERROR-MESSAGE

CEMI#NOK DS ØH
EXEC CICS HANDLE ABEND RESET
MVC EIS_MSG, BLANK
MVC EIS_MSG(L'MSGØ01),MSGØ01
BAL R8, MESSAGE WRITE MESSAGE
B RETURN
EJECT

NETVIEW COMMUNICATION MESSAGES

SPACE

NETVIEW_MSG13 DS ØH
MVC EIS_MSG, BLANK
MVC EIS_MSG(L'MSGØ13),MSGØ13
BAL R8, MESSAGE WRITE MESSAGE
B RETURN

NETVIEW_MSG14 DS ØH
MVC EIS_MSG, BLANK
MVC EIS_MSG(L'MSGØ14),MSGØ14
BAL R8, MESSAGE WRITE MESSAGE
B RETURN
EJECT

RETURN TO CALLER

SPACE

RETURN DC ØH Ø'
EXEC CICS RETURN
EJECT

MESSAGE WRITE ROUTINE

EXEC CICS WRITE OPERATOR TEXT (EIS_MSG) *
RESP (EIS_RESP) *
BR R8 RETURN TO CALLER EJECT
* DEFINITIONS AND LITERALS *
* *
SPACE 1

COMPTR EQU R6 REGISTER FOR COMMAREA
CWAPTR EQU R10 REGISTER FOR CWA-DSECT

DS OD

MSGØØ1 DC C'CSSHUT-ØØ1 shutdown-failure'
MSGØØ2 DC C'CSSHUT-ØØ2 immediate shutdown-processing'
MSGØØ3 DC C'CSSHUT-ØØ3 normal shutdown-processing'
MSGØØ4 DC C'CSSHUT-ØØ4 program-failure for ECSA-Statistics'
MSGØØ5 DC C'CSSHUT-ØØ5 program-failure for NTB-Monitor'
MSGØØ6 DC C'CSSHUT-ØØ6 program-failure for DPII-Monitor'
MSGØØ7 DC C'CSSHUT-ØØ7 stop CICS-DB2 Attachment-Facility is in progress'
MSGØØ8 DC C'CSSHUT-ØØ8 stop CICS-DB2 Attachment-Facility terminated'
MSGØØ9 DC C'CSSHUT-ØØ9 program-failure for DB2-Attachment-Facility'

MSGØ1Ø DC C'CSSHUT-Ø1Ø INFO-CICS setup is in progress'
MSGØ11 DC C'CSSHUT-Ø11 INFO-CICS setup terminated'
MSGØ12 DC C'CSSHUT-Ø12 INFO-CICS setup error'
MSGØ13 DC C'CSSHUT-Ø13 Illegal program communication detected'
MSGØ14 DC C'CSSHUT-Ø14 -- SHUTDOWN -- possible via NetView'
MSGØ15 DC C'CSSHUT-Ø15 Inter-test-stop failure'
MSGØ16 DC C'CSSHUT-Ø16 Error during execution program CSMONØ5'
MSGØ17 DC C'CSSHUT-Ø17 XXMATT-Exit cannot be enabled'
MSGØ18 DC C'CSSHUT-Ø18 Stop IP-Socketinterface impossible'
CAUSTOP DC C'STOP'
CSXXMATT DC C'CSXXMATT'
XXMATT DC C'XXMATT'
BLANK DC CL256'

EJECT EQUIREG REGISTER EQUATES
EJECT LTORG
DC C'

Claus Reis
CICS Systems Programmer
Nuernberger Lebensversicherung AG (Germany) © Xephon 2004

Audit trail for CICS maxtask events

Some problems that cause short-lived maxtask events in CICS can be difficult to debug. The following two programs can help to show what is happening both within and across systems when maxtask occurs.

COLLECTION PROGRAM

The first program, CZSMXTRP, runs at the XEIOUT exit point because there is no exit point supplied specifically for maxtask events and XEIOUT gives good results for this purpose. It collects data about which tasks are active and why they are suspended (resource name and resource type) and writes the data to the exit global work area (GWA).

CZSMXTRP will also cause a snap dump (MXT001) unless this is suppressed. If a maxtask event lasts for any length of time there is potential for collecting and printing an excess of data, so the program checks for an elapsed time of at least one minute before it records data again. This gives a snapshot, minute by minute, of what was happening during a long maxtask event.

Because of a lack of supplied DSECTS for certain control blocks, some offsets are hard-coded, and so they may change between releases. The programs work as written for CICS 2.2. For earlier releases, see notes in the code. There is only one change for CICS/TS 1.3.

PRINT PROGRAM

The second program, CZSMXTPR, runs at PLT time and then reschedules itself every three minutes. It takes any data written to the GWA since it last executed and prints it via a TD queue (MXTP) to a sysout dataset. The active task at data collection time is highlighted (asterisks) on the output. The GWA storage is flagged for re-use once the data is printed.
IMPLEMENTATION

There is negligible overhead in running these programs.

CZSMXTPR should be added to the PLT in all systems where the exit program is active.

To activate the exit we use the following command in a program in the PLT:

```
EXEC CICS ENABLE PROGRAM('CZSMXTRP') EXIT('XEIOUT')
          GALENGTH(32767) START
```

The output looks something like the following:

```
MXT SUMMARY FROM Ø3329 AT Ø9:33:58
TASK# TRAN TYPE NAME     TERMID

26353 PG55 DSTSKDEF :f::ARL1
26351 STØØ LMQUEUE -AGE
26350 SC02 LMQUEUE 2176
26348 STØØ IRLINK CIDC>AAM AHZ ****
26347 STØØ ZCIOWAIT DFHZARR1 AFZ
26339 STØØ ALLOCATE CIC2 AG5
26338 STØØ IRLINK CIC2>ABT AH1
26337 STØØ LMQUEUE AHW
26336 STØØ LMQUEUE AGY
26333 STØØ ALLOCATE CIC2 AH7
26332 STØØ AG0
26327 STØØ ZCIOWAIT DFHZARL1 AF4
26325 STØØ IRLINK CIDC>AAA AGB
26316 STØØ IRLINK CIDC>ABB AG7
26315 STØØ IRLINK CIDC>ABB AFK
26291 STØØ IRLINK CIC2>ABQ AHB
26264 PF94 2385
00050 OMEG USERWAIT SR2WORK
00045 CKTI MQSeries GETWAIT
00044 CKAM
00038 AAON ECWAIT SINGLE
00037 OMEG USERWAIT SRVWORK
00035 CSNE ZC DFHZNAC1
00033 CSHQ SHSYSTEM
00020 CSZ1 FEPRM SZRDP
00019 CSNC CSNC MROQUEUE
00008 CSTP TCP_NORM DFHZDSP
00006 CSSY ICEXPIRY DFHAPTX
00005 CSSY ICMIDNTU DFHAPTX
```
THE PROGRAMS

CZSMXTRP

TITLE 'CZSMXTRP - XEIOUT GLOBAL USER EXIT PROGRAM'

*=====================================================================*
* DESCRIPTION:                                                      *
* IF SYSTEM IS AT MAXTASK THEN STORE INFO ON EACH TASK              *
* (TASK NUM/TRAN-ID/WAIT REASONS) INTO THE GWA FOR POST-PROCESSING   *
* BY CZSMXTPR WHICH PRINTS THE DATA VIA A TD QUEUE.                  *
* THE MAX GWA SIZE IS 32K. THE PROGRAM IS CODED TO BE ABLE TO        *
* STORE UP TO 10 MXT EVENTS WITH A CICS MXT VALUE OF UP TO 100.      *
* RELEASE DEPENDENCIES:                                             *
* THE PROGRAM WORKS AT CICS/TS V2.2                                 *
* SEE CODE AT LABEL 'GETTCA' FOR CHANGE TO XMTxn OFFSET FOR          *
* EARLIER RELEASES.                                                 *
*=====================================================================*

PRINT GEN
DFHREGS  REGISTER EQUATES
COPY  DFHCSADS  DSECT FOR CSA
DFHUEXIT  TYPE=EP, I D= XEIOUT  STANDARD UE PARAMETERS
DFHUEXIT  TYPE=XPIENV  SET UP XPI ENVIRONMENT
DFHAFCD  TYPE=DSECT  DSECT FOR AFCB
COPY  DFHDUDUY  DSECT FOR XPI DUMP CALL

* G ADSECT  DSECT
* GALIT  DS  CL8  EYECATCHER 'CZSMXTRP'
* GALPRT  DS  CL8  DATE/TIME LAST PRINT
* GALMAX  DS  CL8  DATE/TIME LAST MXT
* GA WORK  DS  CL8  WORK DATE/TIME
* GALMENT  DS  10CL28Ø8  UP TO 10 MXT EVENTS TO BE STORED

* GAENTRY  DSECT
* GADATE  DS  CL4  DATE
* GATIME  DS  CL4  TIME
* GATRANS  DS  100CL28
* GATRANS COVERS A MAXIMUM OF 100 TASKS WHERE
* EACH 28 BYTE ENTRY CONSISTS OF TASK NUMBER (4)
* TRAN-ID  (4)
* RESOURCE TYPE (8)
* RESOURCE NAME (8)
* TERM-ID  (4)
* INITIALISATION - SAVE REGS ETC. *

CZSMXTRP CSECT
CZSMXTRP AMODE 31
CZSMXTRP RMODE ANY

SAVE (14,12)
LR R11, R15 SET UP BASE REGISTER
USING CZSMXTRP, R11
LR R3, R1 ADDRESS STANDARD PARAMETERS
USING DFHUEPAR, R3

* GET CSA ADDR AND CHECK IF SYSTEM IS AT MAXTASK *

DFHAFCD TYPE=LOCATE, REG=R9 FIND THE AFCB
L R9, AFCSA-DFHAFCB(R9) ADDR CSA IN R9
DROP R13
USING DFHCSDS, R9
TM CSAKCMI, CSAMXTON ARE WE AT MAXTASKS?
BNO RETOK NO - GOBACK RC=Ø

* WE ARE AT MAXTASK - ADDRESS GLOBAL WORK AREA *

L R4, UEPCAA ADDRESS GLOBAL AREA
USING GADSECT, R4 GLOBAL AREA ENTRY DSECT
CLC GALIT, 'C' CZSMXTRP CHECK IT HAS BEEN INITIALISED
BE GETDATE YES - GET THE DATE AND TIME
MVC GALIT, 'C' CZSMXTRP MOVE LITERAL IN

* GET CURRENT DATE / TIME AND CHECK IF AT LEAST 1 MINUTE HAS ELAPSED *
* SINCE LAST MXT EVENT. IF NOT, JUST IGNORE THIS EVENT. *

GETDATE EQU *
MVC GAWORK(4), CSAJYDP MOVE DATE TO GWA HEADER AREA
L R5, CSACTODB CURRENT TIME IN 100THS SEC
S R5, '6000' SUBTRACT A MINUTE
ST R5, GAWORK+4 STORE TIME LESS 1 MINUTE IN GWA
CLC GAWORK, GALMAX IS IT A MINUTE SINCE LAST MXT
BL RETOK NO - TOO SOON SO IGNORE

* IT'S BEEN > 1 MINUTE - SAVE THE DATE/TIME THEN FIND A FREE AREA *
* (SLOT) IN THE GWA ARRAY TO STORE INFO *

MVC GALMAX(4), CSAJYDP YES - STORE CURRENT DATE
MVC GALMAX+4(4), CSACTODB AND TIME
LA R5, GALMNT GET 1ST SLOT ADDRESS
L R6, '28080' SIZE OF SLOTS
AR R6, R5 END OF SLOTS ADDRESS
USING GAENTRY, R5 ADDRESS ELEMENTS
ELLOOPEQU *
CLC GADATE(8),=D'0'          IS IT EMPTY?
BE USEIT                    YES - USE THIS ONE
CLC GADATE(8),GALPRT       IS DATE < LAST PRINTED DATE
BL USEIT                   YES
LA R5,2808(,R5)            NEXT SLOT ADDRESS
CR R5,R6                   HAVE WE CHECKED ALL SLOTS?
BL ELLOOP                  NO - CHECK THE NEXT ONE
B RETOK                    YES - NO ROOM SO RETURN

*--------------------------------------------------------------------*
* GOT A SLOT TO USE - CLEAR THE SLOT THEN GET DATA               *
* - CHECK OFFSETS TO ALLOW FOR DIFF CICS RELEASES                *
*--------------------------------------------------------------------*

USEIT EQU *
MVC GAWORK,=D'0'           CLEAR OUT GAWORK FOR FLAGS
MVC GADATE(8),GALMAX      STORE DATE/TIME IN THE ENTRY
LA R6,GATRANS            ADDRESS START OF ARRAY
LA R7,20                 LOOP COUNTER
INITL EQU *
MVC Ø(140,R6),=CL140'     CLEAR THIS SLOT
LA R6,140(,R6)           NEXT 5
BCT R7,INITL
LA R6,GATRANS            ADDRESS START OF ARRAY

*---------------------------------------------------------------------*
* KERNEL ANCHOR BLOCK - FIXED AT X'6000' OR X'7000'                *
*---------------------------------------------------------------------*
L R7,=F'24576'            KERNEL ANCHOR BLOCK X'6000'
CLC 2(9,R7),=C>'DFHKEKCB' CHECK EYECATCHER
BE GOTKE                  YES - DO DISPATCHER
L R7,=F'28672'            KERNEL ANCHOR BLOCK X'7000'
CLC 2(9,R7),=C>'DFHKEKCB' CHECK EYECATCHER
BNE ERRORKE               WRITE ERROR MSG AND QUIT

*---------------------------------------------------------------------*
* DISPATCHER ANCHOR BLOCK - KE + X'184' = ADDR OF DFHDS            *
* - DFHDS + X10' = ADDR OF DS ANCHOR                               *
*---------------------------------------------------------------------*
GOTKE EQU *
L R7,X'184'(,R7)          DFHDS ADDR
CLC Ø(8,R7),=C>'DFHDS'   CHECK EYECATCHER
BNE ERRORDS               WRITE ERROR MSG AND QUIT
L R7,X'10'(,R7)           DISPATCHER ANCHOR BLOCK ADDR
CLC 2(12,R7),=C>'DFHSDSANCHOR' CHECK EYECATCHER
BNE ERRORDSA              WRITE ERROR MSG AND QUIT

*---------------------------------------------------------------------*
* EXECUTABLE CHAIN - DS ANCHOR + X'AC' = FIRST DTA                 *
* - DTA + X'80' = XMT BLOCK                                       *
*---------------------------------------------------------------------*
PROCDTA EQU *
L R7,X'AC'(,R7)           FIRST DTA ADDRESS
L R8,=F'0'                XMT ADDR PRESENT??

BE NEXTDTA  NO - IGNORE AND GET NEXT
CLC 2(7,R8),=C' >DFHXMT' CHECK EYECATCHER
BNE ERRORXMT WRITE ERROR MSG AND QUIT

* -------------------------------------------------------------------- *
* MOVE TRANSACTION DATA TO SLOT (R6 ADDRESSES THE TRANSACTION ENTRY) *
* -------------------------------------------------------------------- *
MVC Ù(4,R6),X'3C'(R8) TASK NUMBER
MVC 4(4,R6),X'48'(R8) TRANSACTION ID
MVC 8(8,R6),X'C'(R7) RESOURCE TYPE
MVC 16(8,R6),X'1C'(R7) RESOURCE NAME

GETTCA EQU *
* Exclude one of next 2 statements depending on your CICS/TS release *
* L R8,X'88'(R8) TCA ADDRESS (CICS V1.3)
  L R8,X'E8'(R8) TCA ADDRESS (CICS V2.2)
C R8,CSACDTA CURRENTLY DISPATCHED TASK?
BNE NOTCURR NO
MVI Ù(R6),X'99' YES - FLAG IT FOR LATER

NOTCURR EQU *
CLI X'7'(R8),X'Ø1' TERMINAL FACILITY ADDRESS?
BNE BUMPIT NO - NON-TERMINAL TASK
L R8,X'8'(R8) TCTTE ADDRESS
C R8,=F'Ø' DO WE HAVE ONE?
BE BUMPIT NO - CHECK FOR IRLINK
MVC 24(4,R6),Ø(R8) GET TERMINAL ID

* --------------------------------------------------------------------- *
* BUMP UP TO NEXT TRANSACTION ENTRY AND GET THE NEXT DTA.             *
* CHECK WE DON'T PASS THE END OF BOTH CHAINS                          *
* --------------------------------------------------------------------- *
BUMPIT EQU *
LA R6,28(,R6) STEP UP TO NEXT ENTRY
LA R8,GATRANS+28ØØ END OF SLOT
CR R6,R8 HAVE WE HIT THE END (100 TRANS)
BNL RETOK YES - NO ROOM AT THE INN

NEXTDTA EQU *
L R7,X'2C'(,R7) STEP UP NEXT DTA
C R7,=F'Ø' HAVE WE DONE THE LAST DTA?
BNE PROCDTA NO - PROCESS DTA
B BUMPIT YES - TAKE SNAP DUMP AND RETURN

* --------------------------------------------------------------------- *
* ADDRESSING ERRORS - WRITE MSG INTO GLOBAL AREA AND QUIT             *
* --------------------------------------------------------------------- *
ERRORKE MVC Ù(24,R6),=C'ERROR KERNEL ADDR '
  B RETOK
ERRORDS MVC Ù(24,R6),=C'ERROR DFHDS ADDR '
  B RETOK
ERRORDSA MVC Ù(24,R6),=C'ERROR DS ANCHOR ADDR '
  B RETOK
ERRORXMT MVC Ù(24,R6),=C'ERROR DFHXMT ADDR '
  B BUMPIT
* OPTIONALLY TAKE A REGION DUMP WHEN MXT OCCURS.                     *
* TO SUPPRESS DUMP EITHER COMMENT OUT THE FOLLOWING CODE UP TO      *
* 'RETOK' LABEL, OR USE 'CEMT S SYD(MXTØØ1) MAX(Ø)' IN CICS REGION. *

DUMPIT EQU *
    L R5,UEPXSTOR
    USING DFHDUDU_ARG,R5
    LR R12,R13
    L R13,UEPSTACK
    DFHDUDUX CALL,
        CLEAR,
        IN,
        FUNCTION(SYSTEM_DUMP),
        SYSTEM_DUMPCODE('MXTØØ1'),
        OUT,
        DUMPID(*),
        RESPONSE(*),
        REASON(*)
    LR R13,R12

* SET RETURN CODE AND GOBACK                                         *

RETOK EQU *
    L R13,UEPEPSA
    RETURN (14,12),RC=UERCNORM
    LTORG
    END

CZSMXTPR

TITLE 'CZSMXTPR - PRINT GWA DATA COLLECTED BY CZSMXTRP'

*==================================================================*
* DESCRIPTION:                                                      *
* THIS PROGRAM SHOULD BE ADDED TO THE PLT. IT WILL CHECK FOR ANY    *
* DATA COLLECTED BY THE CZSMXTRP EXIT PROGRAM INTO THE GWA WHEN     *
* A MAXTASK EVENT OCCURS.                                           *
* ANY DATA IS FORMATTED OUT ON TO TD QUEUE 'MXTP' WHICH IS           *
* DIRECTED TO A JES SYSOUT DATASET. THIS PROVIDES AN AUDIT TRAIL    *
* OF MAXTASK EVENTS.                                                *
* THIS PROGRAM WILL RESCHEDULE ITSELF TO RUN AT 3 MINUTE INTERVALS. *
*==================================================================*

DFHREGS    REGISTER EQUATES

* GADSECT DSECT
GALIT DS CL8                     EYECATCHER 'CZSMXTRP'
GALPRT DS CL8                     DATE/TIME LAST PRINT
GALMAX DS CL8         DATE/TIME LAST MXT
GAWORK DS CL8         WORK DATE/TIME
GALMNT DS 10CL2808    10 * TRANSACTION ENTRIES

* --------------- * * GLOBAL WORK AREA MAX TASK ENTRY * * --------------- *
GAENTRY DSECT
GADATE DS CL4         DATE
GATIME DS CL4         TIME
GATRANS DS 100CL28    TASK#/TRANID/RETYPE/RENAME

* * DFHEISTG - DYNAMIC STORAGE * *

DFHEISTG DSECT
LABEL DS 6F
RESP1 DS F
RESP2 DS F
CDATE DS F          CURRENT DATE PACKDEC 0CYYDDD+
CTIME DS F          CURRENT TIME BIN 100TH SECS
WORKTIME DS D
WORKMSS DS F
WORKMASK DS CL6
LENGTH DS H
TDLEN DS H
TDREC DS CL133

* * PROGRAM ENTRY * *

CZSMXTPR CSECT
MVC LABEL(24),LABEL1   INIT EYECATCHER
MVC TDLEN,=H'133'      SET OUTPUT LENGTH
MVC CDATE,EBDATE       SET CURRENT DATE

* * CONVERT TIME FROM PACKED DECIMAL 0HHMMSS+ TO BINARY 100TH SECONDS * *

L R6,=F'0'            WORK REGISTER FOR TIME
MVC WORKTIME,=D'0'    INITIALISE WORK TIME FIELD
MVC WORKTIME+4(4),EITIME MOVE IN CURRENT TIME

* HOURS
L R4,=F'0'            ZERO REMAINDER FOR DIVIDE
CVB R5,WORKTIME      CONVERT TIME TO BINARY
D R4,=F'10000'       GET JUST THE HOURS IN R5
ST R4,WORKMSS        SAVE MMS FOR LATER
M R4,=F'360000'      GET HOURS IN 100THS IN R5
AR R6,R5             SUM HOURS

* MINUTES
L R4,=F'0'            ZERO REMAINDER FOR DIVIDE
L R5,WORKMSS         LOAD UP MINS AND SECS
D R4,=F'100'         GET JUST THE MINS IN R5
ST R4,WORKMSS        SAVE SECONDS FOR NOW
M     R4, =F' 6000'
AR    R6, R5

* SECONDS
L     R4, =F' 0'
L     R5, WORKMMSS
M     R4, =F' 100'
AR    R6, R5
ST    R6, CTIME

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

Peter Duvall
Designer/Developer
Cooperative Bank (UK)
BMC Software has announced Version 5.7 of MAINVIEW for CICS and Version 4.4 of Energizer for CICS.

MAINVIEW for CICS optimizes CICS transactions, monitoring and automating CICS functions and ensuring that a company’s CICS regions are tuned and performing.

Version 5.7 adds real-time system maintenance capabilities to enable MAINVIEW for CICS maintenance upgrades to be done without affecting service of key CICS applications, and application delay analysis (real-time and historical) to enable customers to quickly pinpoint CICS applications that are being delayed (potentially affecting critical business services).

Energizer for CICS improves CICS performance management and transaction throughput by optimizing the way z/OS dispatches CICS transactions. The product dynamically optimizes and tunes the CICS environment during peak processing. It eliminates bottlenecks, lockouts, and sympathetic outages, and reduces CPU utilization and resources for all workloads, says the company.

For further information contact: BMC Software, 2101 City West Blvd, Houston, TX 77042-2827, USA. Tel: (713) 918 8800. URL: http://www.bmc.com/products/proddocview/0,1236,19052_19429_28260_8578,00.html.

---

IONA has announced Artix Mainframe, which is intended for enterprises with extensive investments in either IMS or CICS, or both. The product extends the life and value of these mainframe systems by making it easy for mainframe developers to expose IMS and CICS COMMAREA transactions as Web services in a larger service-oriented architecture. These reusable business-level services are accessible to other developers or applications, which means that mainframe transactions can be combined and reused in new enterprise applications operating on other platforms.

Artix Mainframe uses existing mainframe security facilities to authenticate and authorize access to mainframe-based services. It runs as a native mainframe process, which gives systems programmers control over its administration.

Artix Mainframe is available in two versions. Artix Mainframe Developer is intended for users of IONA’s CORBA-based Orbix Mainframe product. Artix Mainframe Transformer is intended for those organizations that have decided to use service-oriented architectures to extend their legacy applications.

Like the Developer version, Artix Mainframe Transformer provides a suite of simple Windows-based GUI tools for automating and managing the creation of new Web services interfaces to IMS and CICS transactions, without the need for coding or code generation.

For further information contact: IONA Technologies, 200 West Street, 4th Floor, Waltham, MA 02451, USA. Tel: (781) 902 8000. URL: http://www.iona.com/pressroom/2004/20040315.htm.