CICS update

209

April 2003

In this issue

CICS Update

Published by

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

North American office

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

Subscriptions and back-issues

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

CICS Update on-line

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

Editor

Trevor Eddolls

Contributions

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

Prevent 24x7 CICS regions from abending with a 722 error

We have two production TORs in a CICSPlex. These two CICS regions are available 24x7 and they are only cycled at IPL time. We have had several cases over the last few years where one of these production TORs would abend during the day (peak hours no doubt) with a 722, which is an output limit reached.

Since the regions are very rarely cycled, the number of messages written to the MSGUSR log kept growing until we hit the output limit. We made several attempts to increase the output limit, but the region kept abending with a 722 after some time. We also looked into trying to suppress various messages by coding user exit XMEOUT. This worked, but we had some problems and realized that we needed to be careful about which messages were being suppressed.

After working with IBM, they told me of a way to prevent the 722 abends while still maintaining all the CICS messages. I am not sure where, or even whether, this is documented in any of the manuals.

The first thing you need to do is edit your start-up JCL and remove the MSGUSR DD statement. If you remove this, and CICS attempts to write to CSSL, it will dynamically allocate MSGUSR for you. This is important. If you don't do this and you close and reopen the TDQ, it will just continue to write to CSSL where it left off. If you remove it from the start-up JCL and then you close it, the data is written to JES. When you reopen the TDQ, a new one is allocated.

With that said, we wrote a program that will close and re-open the CSSL transient data queue. We defined a new transaction called MSET (MSGUSR reset). The program serves two purposes. First, it goes in the PLT to start the first MSET transaction at midnight after the region is brought up. Secondly, the PCT definition for MSET will point to this program. It will close and open the CSSL TDQ and then start a new MSET to run in 24 hours. This process will continue to run until the region is brought down, several weeks later.

Life is a little easier for us also because each day’s messages are written to a separate JES DD statement. If we know on what day an error occurred, all we need to do is search that day’s log. Previously, we had to search the whole MSGUSR log, several days’ worth of messages, to find what we were looking for.

Here is a review:

1 Remove the //MSGUSR DD from your start-up JCL.

2 Add a PLT entry for SYS200 to your 24x7 regions.

3 Create a PCT entry for MSET that points to program SYS200 and install it on the region.

PROGRAM SYS200

```
IDENTIFICATION DIVISION.
       PROGRAM-ID.     SYS2ØØ.
      *****************************************************************
      *
      * THIS PROGRAM IS BOTH THE PLT PROGRAM AND THE APPLICATION
      *   PROGRAM. IT WILL CLOSE THE TDQ CSSL AT MIDNIGHT FOR THE
      *   24X7 CICS REGIONS.  THIS WILL CREATE ONE DAY'S LOG FOR
      *   THE MSGUSR QUEUE.  IT WILL THEN START ITSELF TO RUN
      *   AGAIN IN 24 HOURS.
      *
      * MSET IS THE TRANSACTION WE USE.  IF YOU WISH TO USE A
      *   DIFFERENT TRANSACTION, DO A CHANGE ALL 'MSET' TO YOUR
      *   TRANSACTION ID.
      *
      *****************************************************************

       ENVIRONMENT DIVISION.
       DATA DIVISION.

       WORKING-STORAGE SECTION.

      *****************************************************************
       PROCEDURE DIVISION.

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

```
* IF NOT STARTED BY A PLT PROGRAM, IN OTHER WORDS, IT WAS
*   STARTED BY THE MSET TRANSACTION, 1) CLOSE THE CSSL TDQ,
*   2) OPEN THE CSSL TDQ, 3) SCHEDULE MSET TO RUN IN 24 HOURS.
*
******************************************************************
     IF EIBTRNID IS EQUAL TO 'MSET'
         EXEC CICS SET
              TDQUEUE('CSSL')
              CLOSED
         END-EXEC

         EXEC CICS SET
              TDQUEUE('CSSL')
              OPEN
         END-EXEC

         EXEC CICS START
             TRANSID('MSET')
             INTERVAL(240000)
         END-EXEC
******************************************************************
*
* IF STARTED BY A PLT PROGRAM, SCHEDULE THE TRANSACTION TO RUN
*   1 MINUTE AFTER MDNIGHT.  SO WHATEVER TIME THE CICS REGION
*   WAS BROUGHT UP, THE FIRST RUN OF MSET WILL OCCUR AT 1
*   MINUTE AFTER MIDNIGHT.
*
******************************************************************
     ELSE
         EXEC CICS START
              TRANSID('MSET')
              TIME(000100)
         END-EXEC.

     EXEC CICS
         RETURN
     END-EXEC.
```

Jon McCabe
CICS Systems Programmer
CUNA Mutual Group (USA)

Articles for inclusion in *CICS Update* can be sent to the editor, Trevor Eddolls, at trevore@xephon.com. A copy of our *Notes for Contributors* can be downloaded from www.xephon.com/nfc.

DFHRMUTL enhancements – or can I cold restart CICS?

INTRODUCTION

One of the more common user requirements is to be able to determine whether or not a CICS system may be cold-restarted after its previous shutdown, and what effect upon the system's data integrity would result if the wrong decision were made in this regard. Over the years, many user-implemented solutions to this problem have been offered – a number of these have appeared in *CICS Update* itself.

This article describes the background to the problem, and explains the recent changes made to CICS in order to provide an official IBM solution to facilitate this need amongst the user community.

BACKGROUND TO THE PROBLEM

When a CICS system is shut down, there are a number of states that can result. If CICS is quiesced normally, via a controlled shutdown, it will take appropriate measures to ensure that the system is terminated in a structured manner. Such a quiesce is achieved via an EXEC CICS PERFORM SHUTDOWN command (or the CEMT equivalent). During a controlled shutdown, CICS will ensure that all inflight user tasks are allowed to complete normally whenever possible. This allows them to reach their final task detach syncpoint, and so complete their inflight Units Of Work (UOWs). By doing this, all changes made to recoverable resources such as files, database entries, temporary storage queues (etc) are committed or backed out in a controlled manner, and the data integrity of the system is left in a committed state. CICS also performs various housekeeping operations, such as closing VSAM files, and ensuring that journal buffers are flushed to their underlying non-volatile media (such as logstreams). In addition, CICS records the success of the controlled shutdown by writing a warm keypoint operation to the system log (DFHLOG). This reflects the state of the system at the time of the shutdown.

If CICS is terminated via an immediate shutdown, however (eg via an EXEC CICS PERFORM SHUTDOWN IMMEDIATE command), the shutdown procedure is optimized to ensure that CICS is terminated in a more rapid manner. As such, inflight UOWs are not given the opportunity to complete, and neither is a warm keypoint recorded on the CICS system log. This means that data integrity of recoverable resources is not left in a committed state at the end of such an immediate shutdown. This is a documented side-effect of using immediate shutdowns against CICS – they are not recommended, and controlled shutdowns should be used instead (whenever possible). Note that the same issue of uncommitted changes to recoverable resources will exist if CICS is terminated by means of an operating system cancel command, or if (say) some power failure or machine check were to occur.

Depending on the type of CICS shutdown, a following TYPE=AUTO restart will have differing results. If CICS was shut down normally at the end of the previous run, a TYPE=AUTO restart will result in a warm restart of CICS. The system is recovered by means of the warm keypoint information held on the system log, and the state data held on the CICS global and local catalog datasets. This results in CICS being restored to a state as near as possible to that which existed at the end of the previous run, when all the inflight task activity had completed. Conversely, if CICS had been terminated by an immediate shutdown command (or MVS cancel command), the TYPE=AUTO restart would result in an emergency restart of the system. This would necessitate CICS reading backwards along the CICS system logstream, rebuilding the UOWs that had been present within the system during the previous run, and reconstructing their locks against the recoverable resources which they had been modifying at the time CICS terminated. Having re-established these inflight UOWs, CICS would complete its emergency restart. It would then drive these reconstructed UOWs to back out their changes in parallel with new task activity entering the restarted system. (Note: this concept of parallel backout and new workload activity was introduced with CICS Transaction Server V1.1. Prior to this version of the product, emergency restart processing would perform the backout of inflight UOW recoverable resources in a single-threaded manner as part of the emergency restart itself.)

A TYPE=AUTO restart is considerably quicker than a TYPE=COLD restart. This is because CICS is able to re-establish its environment from selected state data held on the system log and the global/local catalog datasets. There is no process of retrieving resource definitions from the DFHCSD dataset and reinstalling the various groups of resources into the system, as has to be performed during a TYPE=COLD restart. However, it is the case that users may wish to reinstall their CICS system resource definitions from the DFHCSD dataset periodically (for various reasons). In order to achieve this, a TYPE=COLD restart of the CICS system needs to be performed. This reinstalls the definitions from the RDO grouplists. CICS will also perform various other initialization operations that pertain to a cold restart, such as trimming the system logstream of data relating to local resource updates, for example.

SHUNTED UNITS OF WORK

Prior to CICS Transaction Server, users would base their decision on whether or not a TYPE=COLD restart was viable by inspecting the CICS catalog dataset. This is where a warm keypoint was recorded in CICS/ESA 4.1.0 and earlier releases. If a warm keypoint had been written to the catalog at the end of the previous run, this means CICS had been shut down in a controlled manner and so all inflight UOWs had been allowed to commit their changes prior to CICS terminating. This approach was viable for CICS/ESA 4.1.0 since there was no concept of shunted activity.

With CICS Transaction Server, and the new Recovery Manager component of the product, UOWs that fail at specific phases within their processing can now enter a shunted state. This de-couples them from certain resources (eg their terminal), converts their active locks to longer-term retained ones, and makes their log data eligible for movement from the DFHLOG logstream to the DFHSHUNT logstream. DFHSHUNT is provided as a repository for log data used by long-running tasks within the CICS system. These shunting operations are provided automatically by CICS, their purpose being to allow the state of such a UOW to be put 'on ice' for a period of extended duration, until the cause of the failure can be resolved. At this time, the UOW can be unshunted, and its resolution can be completed.

CICS will drive shunting processing for those UOWs that fail at certain points within their syncpoint operations. The more common point at which shunting can occur is when a UOW is 'indoubt' with respect to the resolution of the syncpoint. That is, the coordinating CICS system has instructed the UOW on a subordinate CICS system to prepare itself. Having done this, the UOW is 'indoubt' as to whether to continue to commit its work, or whether the syncpoint coordinator has since determined that the UOW should be rolled back and the work be undone (ie a backwards commit). Until the appropriate syncpoint flow is received on the subordinate CICS system, the UOW is in an indoubt state (it is said to be in the 'indoubt window'). If a communications failure between the CICS systems occurs at this point, the UOW needs to be retained pending re-establishment of communications and receipt of the appropriate syncpoint completion flow. This is achieved by shunting the UOW until this point in time. (Prior to CICS Transaction Server, a heuristic decision as to whether to commit forwards or backwards had to be made, with the corresponding possibility of a mismatch of syncpoint resolutions when communications were eventually re-established.) In addition to shunting owing to a failure whilst indoubt, CICS can also shunt UOWs that fail during their commit or backout processing. Again, such UOWs are shunted until such time as the cause of the failure can be corrected.

Shunted UOW log data is retained on the system logstreams until the cause of the shunted UOW can be determined. This has the potential to take a long time. As stated above, the data can be held on either the DFHLOG or DFHSHUNT logstreams. It is possible that such shunted UOWs will still exist at the time that CICS is to be shut down. Clearly, this is an issue when considering whether CICS may be cold restarted after such a shutdown. Since a TYPE=COLD restart causes log data pertaining to local resources to be discarded by CICS during the restart, any such data for shunted UOWs would be lost too. In other words, the existence of such shunted UOWs is a reason why a TYPE=COLD restart is not applicable in such a case.

While this was not an issue in CICS/ESA 4.1.0 (when shunting of UOWs could not occur), it is a fact to be considered when determining whether or not to cold restart a CICS Transaction Server system.

DFHRM0203 AND DFHRM0204 MESSAGES

To help give guidance in this area, CICS Transaction Server issues one of two Recovery Manager messages while processing a controlled shutdown. These are DFHRM0203 or DFHRM0204. Their explanations are as follows:

DFHRM0203: There are *x* indoubt, *y* commit-failed, and *z* backout-failed UOWs.

Explanation: this message displays the numbers of indoubt units of work (UOWs), backout-failed UOWs, and commit-failed UOWs in the CICS system at the time of the normal shutdown. It is issued only if there is at least one such UOW. If there are none, message DFHRM0204 is issued instead. Messages DFHRM0203 and DFHRM0204 can be used to determine whether or not it is safe to cold start CICS following a normal shutdown without losing resynchronization information. See DFHRM0204 for more information.

DFHRM0204: There are no indoubt, commit-failed, or backout-failed UOWs.

Explanation: there are no indoubt, commit-failed, or backout-failed units of work in the CICS system at the time of the normal shutdown. (If there are any such units of work, message DFHRM0203 is issued instead.) This message indicates that it is safe to do a cold start of CICS without losing any resynchronization information.

These messages are useful when inspecting the joblog of the CICS system being shut down, as they reveal the existence and numbers of shunted UOWs at the time of the shutdown, and may be intercepted/utilized by an intelligent operator package when determining the type of following restart. However, it was felt that the product could be enhanced to provide a more programmable interface for this area, and also to improve the diagnostic information pertaining to shunted UOWs.

CICS MODIFICATIONS TO ADDRESS THE REQUIREMENT

CICS Transaction Server Versions 1.3 and 2.2 have been enhanced to provide new functionality, to help address this requirement. Runtime CICS has been enhanced to create a new type of global catalog record. This is created during a controlled shutdown of CICS. It is also deleted during a restart of CICS. Therefore, existence of the record on the global catalog signifies that a controlled shutdown was used to end the previous run. If the record is not present on the catalog, CICS is either active, or did not terminate in a controlled manner. Either way, determining that a subsequent cold-restart can occur is not viable at this stage.

The new catalog record contains the counts of the number of shunted UOWs present at the time of the shutdown. These are held as three fullword values – the number of indoubt UOWs, the number of commit-failed UOWs, and the number of backout-failed UOWs. The VSAM key of the new global catalog record is a (decimal) 28 byte hexadecimal value: 00000011C4C6C8D9D4C4D440C4C6C8D9D4C4D46DD9C5E2E3C1D9E340.

The new record is held on the global catalog dataset, and so can be examined by user-written programs to determine whether or not a cold restart of CICS is viable. If the record is not found on the catalog then CICS was not shut down in a controlled manner, and so a cold restart is not viable for data integrity reasons. (CICS may of course be active, in which case the record would not exist either.) If the record is found, then the existence of any shunted UOWs at the time of the controlled shutdown will be represented by non-zero values in one or more of the three count fields. If any of these count fields are non-zero, a cold-restart is not viable. Only if the record exists, and its three count fields are all zeros, is it safe for the user to perform a cold-restart of the system.

A record with zero count values corresponds to a DFHRM0204 message issued during the controlled shutdown. Likewise, a record with any non-zero values in its counts corresponds to a DFHRM0203 message. (These diagnostic messages are still produced.)

New language DSECTs are provided to help with programmable analysis of the new catalog record. These are called DFHRMRED, DFHRMREH, DFHRMREL, and DFHRMREO, and are provided for Assembler, C, PL/I, and COBOL respectively. The Assembler DSECT structure is shown below:

```
* -------------------------------------------------------------------
*               Licensed Materials - Property of IBM
*               "Restricted Materials of IBM"
*               5655-147
*               (C) Copyright IBM Corp. 2ØØ2,
*                CICS 5.3.Ø
* The DFHRMREC declaration maps the record held on the global
* catalog for use in determining what type of restart may be
* overridden by DFHRMUTL.
*
* The rmdm_restart_key is a (decimal) 28 byte field of value:
* ØØØØØØ11C4C6C8D9D4C4D44ØC4C6C8D9D4C4D46DD9C5E2E3C1D9E34Ø
* -------------------------------------------------------------------
DFHRMREC DSECT
DFHRMREC__DUMMY  DS ØCL4Ø               Restart record
RMDM_RESTART_KEY DS CL28
RMDM_RESTART_COUNTS DS ØCL12
RMDM_RESTART_COUNTS@RMDM_INDOUBT_UOWS DS FL4 Indoubt UOWs
RMDM_RESTART_COUNTS@RMDM_CFAIL_UOWS DS FL4 Commit fail UOWs
RMDM_RESTART_COUNTS@RMDM_BFAIL_UOWS DS FL4 Backout fail UOWs
DFHRMREC__LEN EQU L'DFHRMREC__DUMMY    Structure length
```

In addition to the run-time changes made to CICS, DFHRMUTL has been enhanced. DFHRMUTL is the CICS-supplied batch program that allows analysis and modification of Recovery Manager restart attributes. For example, the auto restart override can be set by DFHRMUTL. This is how users may implement a change of restart type (by overriding TYPE=AUTO restarts to make CICS start up cold instead). This is achieved by setting the SET_AUTO_START override option to AUTOCOLD.

DFHRMUTL now analyses the catalog record data and includes additional information in its report. The next restart type (eg WARM) is shown and the count fields for the three possible types of shunted UOWs are given. If any of the counts are non-zero, DFHRMUTL issues the new message DFHRM0315, stating that the AUTOCOLD option should not be used. An example of this new DFHRMUTL output is shown below:

```
--DFHRMUTL:   DFHGCD information
   Recovery manager auto-start override   : AUTOASIS
   Recovery manager next start type       : WARM
   UOW information from Global Catalog:
     Indoubt UOWs      : 00000000
     Commit fail UOWs  : 00000000
     Backout fail UOWs : 00000000
```

Note: a CICS system that was shut down warm, and which has no indoubt, commit-failed, or backout-failed Units Of Work keypointed at that time, can safely be restarted cold without loss of data integrity.

In this case, CICS had been shut down in a controlled manner, and no indoubt UOWs existed at the time. As such, a cold restart would be viable.

CICS Transaction Server 1.3 and 2.2 have been enhanced by PTF service to provide support for this new functionality. PTF UQ72907 (APAR PQ60109) has been provided for CICS Transaction Server Version 1.3, and PTF UQ72639 (APAR PQ60089) for Version 2.2.

FURTHER READING

These enhancements to CICS are documented in various manuals, including the *CICS Messages and Codes Manual*, *Customization Guide*, *Supplementary Data Areas Manual*, and *Operations and Utilities Guide*. This last manual describes DFHRMUTL and the various options that can be specified when running it against the CICS catalog. For further information on CICS start-up and shutdown in general, the *CICS Recovery and Restart Guide* gives a detailed description of these operations.

I hope that this article has helped explain the background to using the DFHRMUTL utility for determining whether or not a cold restart of CICS is viable, and the issues involved in CICS making this determination.

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

New CPSMCONN SIT parameter

The DB2CONN, DBCTLCON, and MQCONN System Initialization Table (SIT) parameters introduced in CICS Transaction Server Version 1, Release 2 certainly made administration easier for users of DB2, DBCTL, and MQSeries.

Two recently closed IBM APARs, which will make administration easier for CICSPlex SM users or even ease the implementation of CICSPlex SM, are:

- CICS Transaction Server for OS/390 Version 1, Release3, PQ65166, UQ71532.
- CICS Transaction Server for z/OS Version 2, Release 2, PQ65168, UQ71534.

The APARs introduce a new CPSMCONN SIT parameter, so that CICSPlex SM users no longer need to modify the Program List Tables (PLT).

The CPSMCONN parameter has four options:

- NO – the default.
- LMAS – link to EYU9NXLM (LMAS initialization) during start-up.
- CMAS – link to EYU9XLCS (CMAS initialization) during start- up.
- WUI – link to EYU9NXLM (LMAS initialization) and then EYU9VKIT (WUI server initialization) during start-up. EYU9VKIT will also be linked to during normal shut down.

The APARs also add the system abend codes 052, 053, 067, 0D4, 0D6, 0D7, 0D8, 0E0, and 202 to the pregenerated System Recovery Table (SRT) DFHSRT1$ so that CICSPlex SM users will no longer have to modify the SRT either.

Carl Wade McBurnie
IT Consultant (Germany)

CICS Transaction Gateway Version 5 and CICS Transaction Server Version 2.2: Part 1 – Architecture for connecting a workstation CICS Transaction Gateway to CICS

INTRODUCTION

The new release of the CICS Transaction Gateway, Version 5, enables many connectional options to CICS Transaction Server Version 2.2. This flexibility permits numerous architectural options – making decisions difficult. This article is the first of a series of three dealing with the CICS Transaction Gateway Version 5 (referred to as CTG in this article) and CICS Transaction Server Version 2.2 (CICS in this article), which demonstrate a reliable and secure arrangement.

In this first article, I discuss how to configure the CICS Transaction Gateway residing on a workstation (in other words, not on an OS/390 or z/OS platform) to communicate with CICS via TCP/IP. I will describe a reliable and secure arrangement for a workstation application to access a CICS program.

The second article will discuss arrangements where the CICS Transaction Gateway is residing in MVS (OS/390 or z/OS) and communicating with CICS. The emphasis in this part will be how to arrange things so that an application using WebSphere facilities can use CICS function.

The final article supplies some code that can be used within an Excel spreadsheet to contact CICS.

New communication function introduced with CICS Transaction Server Version 2.2 permits a non-MVS CTG to communicate with a host CICS (running in OS/390 or z/OS) using TCP/IP. This does away with TCP62 functionality and allows an easy configuration of the communication path. The advent of this protocol permits an easier network configuration of the CTG and enables reliable and secure functionality. One can now contact CICS TS 2.2 using TCP/IP without using an intermediate instance of CICS (running on, say, AIX).

I am going to refer to OS/390 and z/OS operating systems generically as MVS; and a CICS Transaction Gateway instance residing outside of MVS as a workstation CTG (this term covers a CICS Transaction Gateway placed on a server or any distributed environment that is not MVS). Similarly, I use the term workstation application to mean a program executing on a workstation, server, or other non-MVS distributed environment.

WHY READ THIS SET OF ARTICLES?

You will discover my favourite way to arrange CICS Transaction Gateways and CICS regions to ensure maximum availability (no critical failure points) with the best performance for a production setup. The arrangements are secure and reliable.

Even better, applications using the CICS Transaction Gateway to access CICS functions do not need to have any knowledge of the architectural configuration.

Worth a read? I hope so!

SOME BACKGROUND

The CICS Transaction Gateway permits applications running outside of the CICS environment to run functions within a CICS region. This can be a screen emulation (using the External Presentation Interface) or, more usually, a CICS application program (via the External CICS Interface).

This series of articles assumes use of ECI functionality – but exactly the same architectural considerations apply to EPI usage (where the CTG supports EPI access).

Applications that access CICS-sourced data can reside within WebSphere (on MVS), and can be Windows-type programs (written in Visual Basic or C++) running on a workstation, programs residing on a Unix/Linux/AIX server, or Java applications. The challenge is to arrange all these disparate environments to provide the maximum availability in a secure fashion.

The CTG usually resides in the same execution environment (Windows or MVS etc) as the application invoking the CICS function. However, Java applications can communicate via TCP/IP to a CTG residing elsewhere. This series of articles does not consider this latter case: the CTG is deemed to be residing on the same box as the application using its facilities (so I'm not getting into, for example, JNDI look-up for the CTG).

The general arrangement is shown in Figure 1:

- The workstation (in general, a non-MVS platform) contains a CICS Transaction Gateway.
- A workstation application wants to execute a module running in a CICS region.

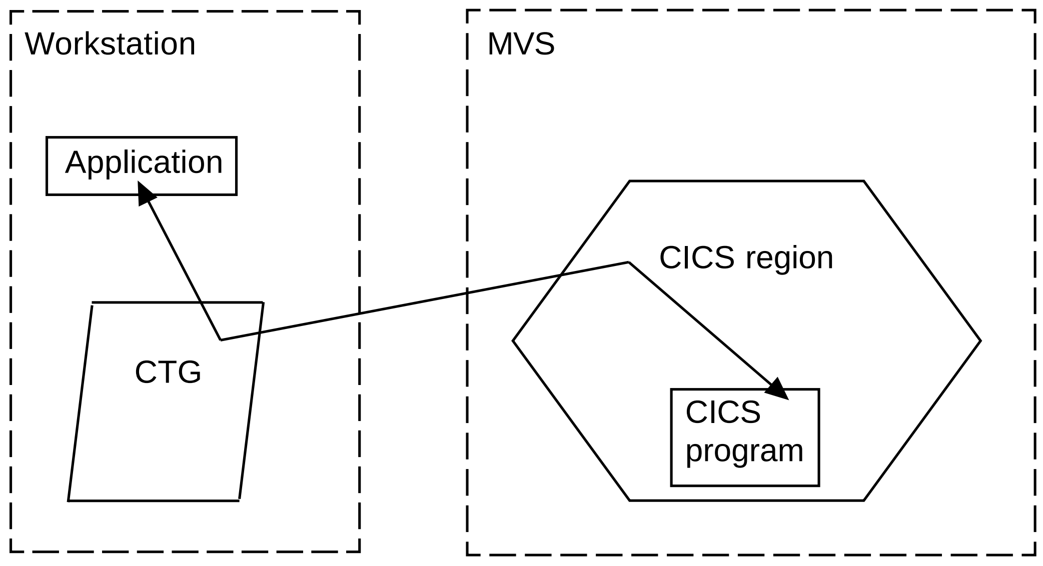

Figure 1: Idealized CTG and CICS workststation configuration

- The request flows from the CTG to CICS and a response is returned.
- COMMAREAs are used to convey the request and response.
- CodePage conversion is required between the two environments (EBCDIC in CICS, ASCII/Unicode on the workstation).

TORS AND AORS

TOR and AOR arrangement

The usual CICS arrangement is to have a Terminal Owning Region talking to many Application Owning Regions. The TOR knows which AOR is to run the transaction and routes a request accordingly. Each AOR can be cloned for workload balancing. The TOR and the AORs can reside within the same or different MVS images. Figure 2 shows this common arrangement.

As far as CTG access is concerned, the CICS program to run resides within an AOR. The question is, how to get to this AOR?

Using the CTG as a TOR

An obvious first arrangement for configuring a workstation CTG to access CICS is to avoid the TOR and use TCP/IP connections directly into the relevant AOR. The CTG is acting as a sort of TOR, but routing is not enabled – paths to the AOR are fixed and static.

This arrangement is not good enough! Do not set up things as shown in Figure 3 – this arrangement is acceptable for a test environment, but not for a production configuration.

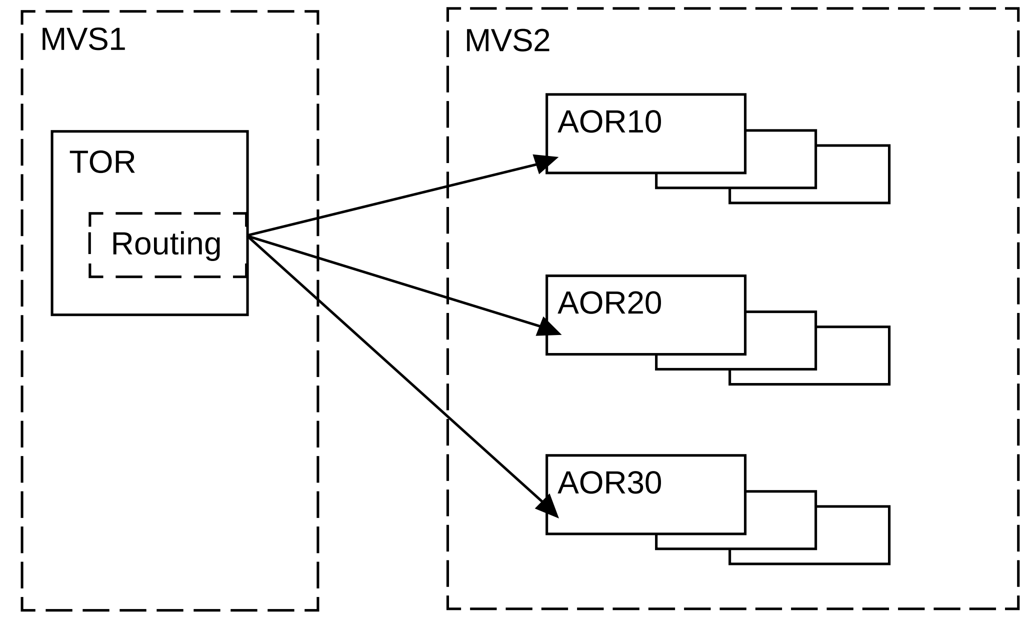

Figure 2: TORs and AORs

WHY IS THIS BAD NEWS?

There are several things wrong with this arrangement (skip ahead to Figure 4 if you want to see my preferred arrangement).

The first thing to observe for Figure 3 is that you are going to have CTG configuration entries for each AOR (three in this case). Each workstation application that wants to use a CICS program will have to know in which AOR the module is available. Consequently, knowledge of the CICS configuration is required

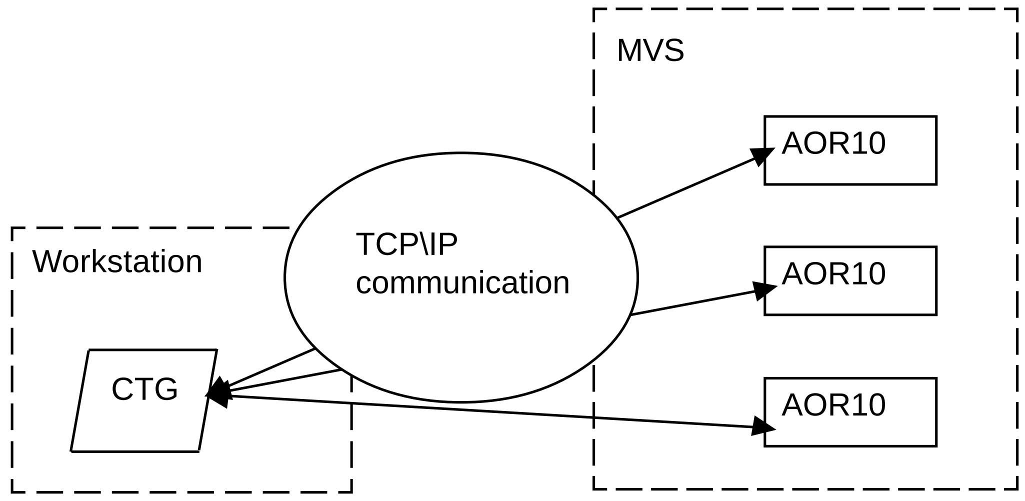

Figure 3: A bad CTG and AOR arrangement

within the workstation application that wants to use the CICS function.

This is bad enough when viewed in security terms, but what happens if the CICS configuration changes, or the AORs are moved to a different MVS image? Not only will CTG definitions have to change, but maybe also the workstation application programs.

These things are not to be encouraged.

Even more bad news

Another major problem for the arrangement in Figure 4 occurs when one of the AORs is not available. Does a workstation application know that the function is available in one of the other AORs? And if so, is the relevant CICS module actually available in that region? How is a workstation application to be told to switch contacts?

And what about the case in which the AOR is too busy to service the request, but one of the other AORs could service the function more rapidly?

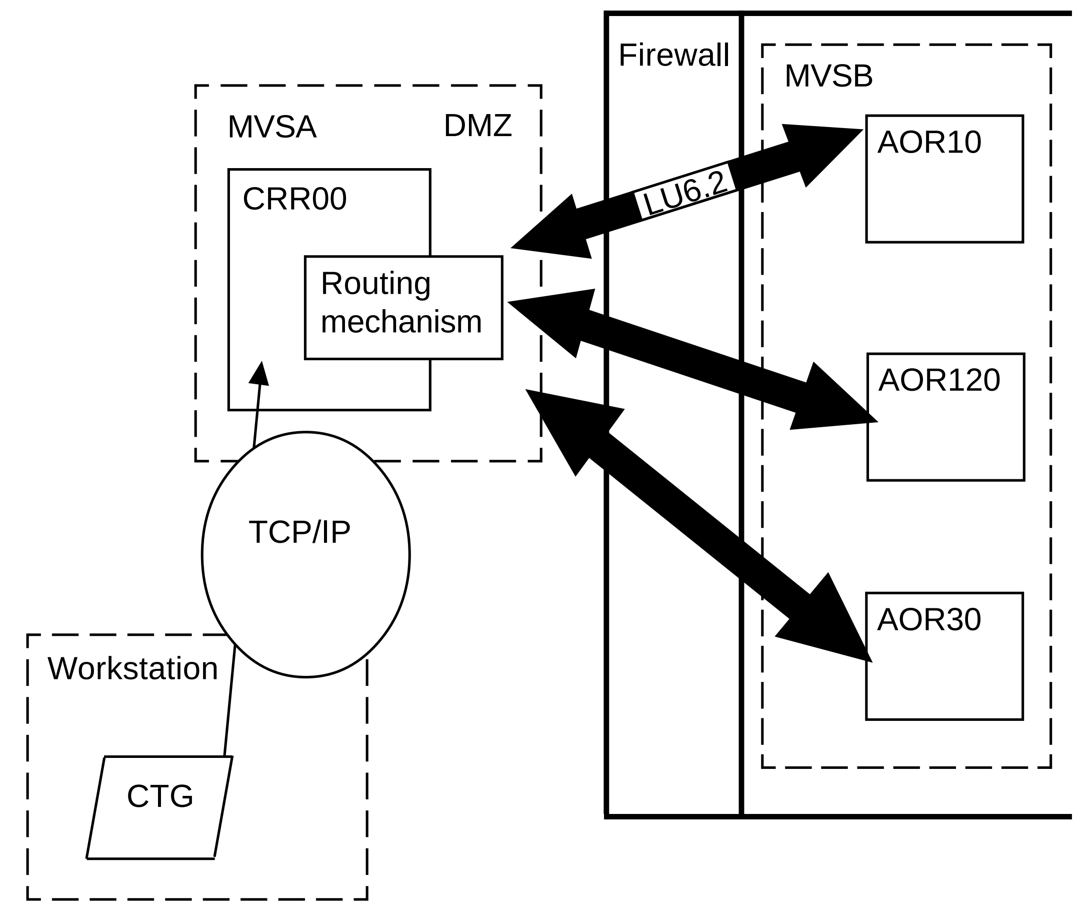

Figure 4: A better way to connect to CICS

USING A CRR

What is a CRR?

These problems are simply solved by insisting that a workstation CTG always communicate to a given CICS region. It is this CICS region that decides where the required function resides and routes the request accordingly. This region is functioning like a TOR, but for CTG requests. I have called this type of region a CTG Routing Region (CRR). Figure 4 shows this arrangement.

The CRR can reside on an MVS image that is different from that used by the AORs. This is to ensure that any possible security problems can be localized and minimized. The CRRs MVS image would reside outside of the company firewall, but the AORs are securely inside it. Communication between the CRR and the AOR is via standard CICS protocols, and is secure.

This arrangement is often called a De-Militarized Zone (DMZ): MVSA is the DMZ because nothing apart from trusted items execute within. No application code is available to be misused in the CRR. In the unlikely event of the CRR being compromised, the firewall will prevent unauthorized activity from being initiated in MVSB (where the real applications reside).

If this is not a concern, the CRR and the AORs will be within the same MVS image and a firewall will not be present. Indeed, a firewall (and its configuration) is tightly bound to an installation configuration – I'm just showing one aspect for illustrative purposes.

Once the request has been sent from the workstation CTG and has arrived at the CRR, standard CICS-based routing mechanisms can decide on the destination for the requested program. In other words, a suitable AOR is selected in which the requested CICS program is executed. This routing can be done statically via RDO facilities, or fluidly (better) using the standard Dynamic Routing Program function. The MVS Workload Manager and CICSPlex SM can be involved in this decision.

Communication with the AORs from the CRR can be via LU6.2 (if in different MVS images) or the EXCI flavour of MRO (if in the same MVS image).

The arrangement in Figure 4 is better than Figure 3 for a flow from the CTG to invoke a CICS application. This is an acceptable arrangement, but I am going to describe a much better configuration.

The CTG is only ever contacting a single CICS region, called the CRR. Consequently, there will only ever be one definition in the CTG configuration file – that of the CRR. This means that the workstation applications invoking CICS modules via the CTG have to know about only a fixed and static name (which refers to the CTG Configuration Entry for communication with the CRR). There is no code in the workstation application concerned with selecting a communication pathway.

CRR location and MVS image

I have located my CRR in a different MVS image from that of the AORs (which are doing the work). I've done this deliberately, because this arrangement gives you the opportunity to locate all TCP/IP activity inside a DMZ, whilst leaving all sensitive applications and data safety behind a firewall. Other firewall placements and configurations are available – I'm just using a specific arrangement for illustrative purposes.

The CRR itself is secure, because there are no application programs running in the region (and so no data that could be hacked or application programs that could be misused). There is only a single TCP/IP port active, which is being used for communication with the CTG. You cannot get rogue programs into a CRR, so the CRR is itself a secure entity.

The CRR will communicate through the firewall using, for example, LU6.2 protocols, which are secure. If neither a firewall nor DMZ is needed, the CRR can happily reside in the same MVS image as the AORs. In this case, the communication between the CRR and the AORs will be via MRO (using EXCI).

Routing requests using the CRR

As all requests from the CTG go into the same CRR, it is the responsibility of the CRR to correctly assign the AOR to run the required program (the name of which is sent from the workstation application). This routing can be static (information defined via RDO) or dynamic. Dynamic routing is better as static implies a single AOR (which may be unavailable).

If the CRR is using the Dynamic Routing Program, the name of the module to execute within an AOR does not have to be the same as that sent from the workstation application, so providing an additional level of security.

In effect, the workstation application does not have to know anything about the arrangement of the CICS regions nor even what is actually being run. This is an important security function in addition to the benefits of not having to alter anything on the CTG or workstation or workstation application whenever anything changes.

This indirectness is an important architectural property for a CTG configuration. My preferred arrangement (coming up) relies on this indirectness.

AOR load balancing and failures

Because the CRR is handling routing, using standard CICS facilities, all the benefits of existing CICS mechanisms are available to manage workload.

For example, if a given AOR has failed, Dynamic Routing can schedule work to a back-up cloned region. If too many requests are flowing to one AOR, the MVS Workload Manager can increase resources to that region. CICSPlex SM can be deployed to manage availability, configuration, and performance.

Consequently, using a CRR enables all existing CICS mechanisms to control how the request from the workstation application is processed. This means that such requests have mainframe-like performance and availability characteristics.

USING MORE THAN ONE CRR

The arrangement in Figure 4 still has a single point of failure – the CRR. For a highly available and performing arrangement, you need another CRR around in order to provide access if the first one fails.

Multiple CRRs

My preferred arrangement is to have two CRRs in place. If one fails, the other can take over communication without any drastic reconfiguration or huge loss of availability.

The key to this is the TCP/IP Port Sharing feature of MVS. This permits two activities:

- Traffic can be balanced between the two CRRs for maximum performance, capacity, and availability.
- One CRR acts as a back-up for the other – so ensuring availability.

All you do is start up two cloned CRR regions, both listening on the same TCP/IP port (they have identical TCPIPSERVICE definitions active with a SHAREPORT statement in the TCP/IP profile), and let MVS manage the rest.

My preferred arrangement

The arrangement in Figure 5 is my preferred solution. It provides:

- Failover if one of the CRRs fails (traffic is directed into the other CRR until it restarts).
- Load balancing of traffic into the CRR from applications (via the CTG).
- Protection against failure of an AOR (work is directed into another AOR if one fails).
- Load balancing of AOR workloads.

A request is sent from a workstation application to its CTG. This forwards the request to what it thinks is a single CICS region, which runs the requested program.

In fact, the request from the CTG gets looked at by the MVS TCP/IP Port Sharing function, which decides on the basis of workload and availability to transmit it to a CRR. In fact, this selection is made on the number of connections (including backlogged) active – the region with the fewest connections gets the flow. The CRR examines the request and, using its knowledge of where the required program resides and the availability of AORs, sends the request to a picked AOR.

This picked AOR is secure (potentially), sitting behind a firewall

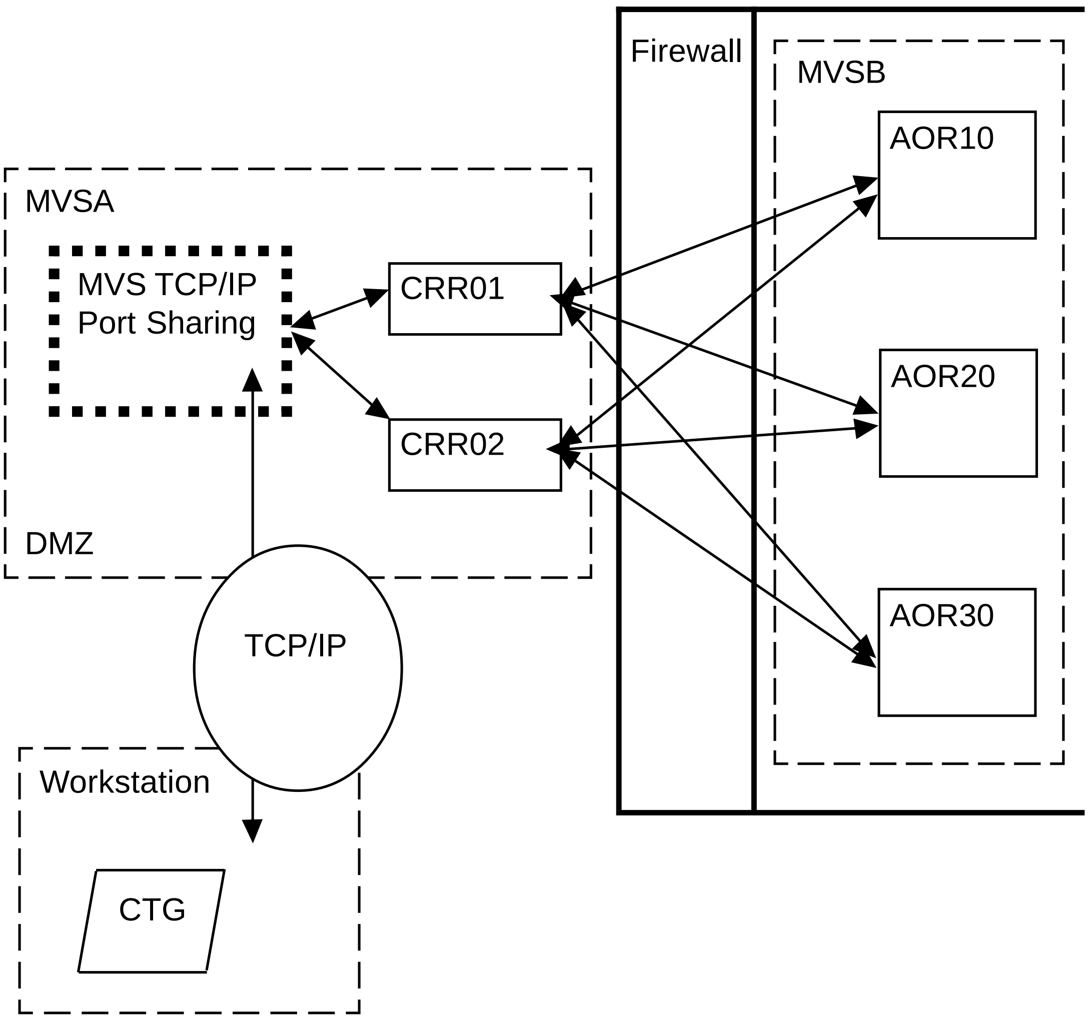

Figure 5: My preferred CTG to CICS framework

– rogue communication cannot occur because the issuing CRR is in a protected DMZ. But, more particularly, security is provided by the type of communication between the CRR and the AOR. EXCI or LU6.2 communication is secure and reliable. The userid and password supplied by the workstation application is RACF (or other ESM) protected. A request will not get past the CRR if an invalid security identity is presented. It is usually the case that this security identity is used to run function in the AOR, which is itself security authorized.

After executing the desired function, the result is sent back to the relevant CRR, which then sends it (directly without any involvement of the MVS TCP/IP Port Sharing function) back to the CTG. The CTG then returns this information to the calling workstation application.

The selection of CRR occurs when the TCP/IP connection is established. Consequently, low activity within the CTG will cause all requests to flow to the initially selected CRR. If traffic is sufficient to require another connection from the CTG to a CRR, the MVS TCP/IP Port Sharing function again selects a suitable CRR. If one of the CRRs becomes unavailable, the connection from the CTG fails, so the next flow will drive selection (which will be to an active CRR).

FURTHER THOUGHTS

Should a CTG directly talk to an AOR?

If there is one thing you should have got out of this article, it is that I think a CTG talking directly to an AOR is a bad idea for the production environment!

Does a CTG talk to more than one CRR?

Although the CTG may be physically in communication with more than one CRR, MVS TCP/IP Port Sharing provides this functionality. As far as the CTG is concerned, it is talking to one CRR.

Should a CTG talk to more than one CRR?

It is preferable that a CTG does not knowingly communicate with more than one CRR. I say knowingly, because use of the MVS TCP/IP Port Sharing function may establish physical connections to multiple CRRs – but, as far as the CTG is concerned, it's talking to only one CRR.

All requests from a particular CTG instance should flow, as quickly as possible, into a CRR. The CRR (being on a mainframe) can then efficiently dispatch the request to the ‘best place’.

Knowledge, therefore, of the best place is in one location on the mainframe – where it can be maintained using traditional CICS change control procedures. All network and configuration changes are managed by CICS, so CTGs (and workstation applications) do not have to be manipulated for a configuration change.

If a CTG supports multiple destinations, information about them would need to be applied to all workstation applications using the CTG. Consequently, details of configuration have to be known by workstation applications and manipulated accordingly. This is not a good programming style as it leads to problems when the configuration changes (as it always will).

What about many CTGs using the same CRR?

This means lots of workstation CTGs all using the same CRR (the CTGs use an identical communication definition). This is perfectly acceptable, because MVS TCP/IP Port Sharing will be distributing the requests amongst the CRRs. Indeed, this is a highly-desirable arrangement.

You do not have a different TCPIPSERVICE RDO definition for each CTG in the CRRs. All the CTGs use the same TCP/IP port – the TCP/IP Port Sharing function distributes the flows around the CRRs.

There should not be a TCP/IP capacity problem in the CRR and neither will there be a CRR performance bottleneck (as little processing is done to merely receive a request and route it to an AOR).

Should a CTG alter the communication destination?

The CTG supports an exit whereby parameters of the workstation application request can be altered. One of these is the destination (the name of a CICS region).

When using the Figure 5 set-up, you can write workstation applications that either use a fixed destination (such as setting it always to 'CICS') or omit it entirely. The CTG exit can then be used to substitute the real name for the CRR as defined in its configuration file.

This approach makes workstation applications using the CTG completely independent of the network configuration and the CTG Communication Definition is hidden.

If you are wondering what the workstation CTG exit looks like, an example is provided along with the CTG in <CTG>\samples\c\exits\ecix1.c. You just fix up the ECI_PARMS.eci_system_name field within routine CICS_EciExternalCallExit1.

Should an application using the CTG select a CRR?

Use a set-up as shown in Figure 5 and you will never need to write selection code in workstation applications using CICS programs via a CTG. You just name the CICS program to run and a security identity to use, and supply a COMMAREA.

There is:

- No need to interrogate the CTG as to which destinations are available.
- No need to incorporate code about selecting the best destination.
- No need to worry about retrying the request somewhere else if it initially failed to connect.

All of this is done automatically in the Figure 5 set-up – the destination is fixed, the physical AOR selection is managed by CICS, and connection failure should never happen.

CONCLUSION

My preferred arrangement for a workstation application using a CICS program via a CICS Transaction Gateway:

- Does not expose CTG configuration information to a workstation application.

- Enables simple workstation application code.
- Can use a DMZ or firewall to supplement secure CICS communication.
- Does not route requests from the CTG directly to the CICS region containing the CICS program
- Uses multiple CICS regions as listeners (CRRs) to provide guaranteed availability using MVS TCP/IP Port Sharing.
- Routes requests from the CRRs to AORs using standard CICS load balancing and availability functions to provide secure and speedy execution of the requested CICS program.

My implementation of this arrangement is shown in Figure 5.

COMING NEXT

The second in this series of articles covers an equivalent configuration where the CICS Transaction Gateway is being accessed on MVS (perhaps via WebSphere).

Robert Harris
CICS Technical Strategist
IBM Hursley (UK)

Have you come across any undocumented features in CICS Transaction Server? Why not share your discovery with others? Send your findings to us at any of the addresses shown on page 2.

A simple interface to CICS load module Scanner Utility

This tool, CICS Scanner Utility (SCU), uses the CICS Transaction Affinities utility.

The Transaction Affinities Utility is designed to detect potential causes of inter-transaction affinity and transaction-system affinity for those users planning to use the CICS dynamic routing facility, or in order to plan asynchronous processing by CICS function shipping, or the transaction isolation facility.

This tool is only an aid in the search for all affinities in the customers' applications. It can be used in order to find which CICS commands (EXEC CICS) are present in the programs and which can cause the transaction affinity. The CICS commands that can be found with use of this tool are specified below.

Inter-transaction affinity commands are:

- ENQ
- DEQ
- READQ TS
- WRITEQ TS
- DELETEQ TS
- ADDRESS CWA
- LOAD
- RELEASE
- GETMAIN SHARED
- FREEMAIN
- RETRIEVE WAIT
- DELAY

- POST
- START
- CANCEL
- COLLECT STATISTICS.

Transaction-system affinity commands are:

- ENABLE PROGRAM
- DISABLE PROGRAM
- EXTRACT EXIT
- INQUIRE
- SET
- PERFORM
- RESYNC
- DISCARD
- CREATE
- WAIT EXTERNAL
- WAIT EVENT
- WAITCICS
- CBTS STARTBROWSE
- CBTS GETNEXT
- CBTS ENDBROWSE.

The Scanner Utility also detects MVS POST SVC calls and MVS POST LINKAGE SYSTEM non-SVC calls, because of their connection with the various EXEC CICS WAIT commands.

The Transaction Affinities Utility does not search for transient data and file control EXEC CICS commands.

The Scanner operation is independent of the language in which the scanned program was written and the release of CICS from which the program was translated.

The Scanner does not detect CICS macro-level commands.

The Scanner does not search for commands issued by any program named DFH*xxxxx* or CAU*xxxxx*.

The essential scope of the tool is to supply a simple interface for the Scanner load component of the transaction affinity utility.

This tool runs the load module scanner utility, which scans a load module library to detect those programs in the library that issue EXEC CICS commands that may cause transaction affinities.

The report produced by the Scanner indicates only that potential affinity problems may exist because it identifies only the programs that issue the commands. It cannot obtain dynamic information about the transactions using the programs, or the names of the resources.

You can run the tool to produce either a summary report or a detailed report of modules that contain possible affinity-causing EXEC CICS commands or MVS POST calls.

The CSU tool locates all commands and provides a summary or detail report based on the choice of the user.

The tool is composed of two parts:

- Generation and execution of the CICS utility job.
- Display of the report produced by the generation function.

The summary report will supply the following information:

- Module name
- Module length
- Module language
- Number of affinity statements for a program
- Number of MVS POSTs for a program

- Comment.

The detail report will supply the following information:

- Module name
- Load module length
- Offset
- Possible command of affinity
- Affinity type
- Total possible affinity commands
- Total possible MVS POSTs.

Both the reports then supply total statistics for the scan, and in particular:

- Total modules in library
- Total modules scanned
- Total CICS modules/tables not scanned
- Total modules in error not scanned
- Total modules containing possible MVS POSTs
- Total modules containing possible affinity commands
- Total Assembler modules
- Total C/370 modules
- Total COBOL modules
- Total COBOL II modules
- Total PL/I modules.

This tool was developed using the REXX language ISPF functions, and was tested in the following environment:

- OS/390 2.6 and OS/390 2.8
- CICS Transaction Server 1.3.

CXSUC000 EXEC

```
/* REXX */
/* CICS SCANNER LOAD MODULE utility
   C-List CXSUCØØØ
   Called by tso user command.
   It executes:
      - the CAULMS CICS utility program;
      - display report of scanner load library;             */
Trace ?o
msgØ1 = ''
/***************************************************************/
/* Init proc and choose function                               */
/***************************************************************/
InitProc:
Do forever
ADDRESS ISPEXEC
'ADDPOP ROW(Ø) COLUMN(Ø)'
'VGET ZKEYS PROFILE'
ZWINTTL=' Cics Scanner Load Utility '
'DISPLAY PANEL(cxsupØØØ)'
if rc = 8 then exit
'REMPOP'
func1 = x
func2 = y
func3 = w
func4 = k
if func1 ¬= '' then Call Exec_Function1
if func2 ¬= '' then Call Exec_Function2
if func3 ¬= '' then Call Exec_Function3
if func4 ¬= '' then Call Exec_Function4
signal InitProc
/***************************************************************/
/* Function generation to summary Scan report                  */
/***************************************************************/
Exec_Function1:
func1 = ''
x = ''
'VPUT X PROFILE'
Do forever
ADDRESS ISPEXEC
'ADDPOP ROW(Ø) COLUMN(Ø)'
'VGET ZKEYS PROFILE'
ZWINTTL='Cics Scanner Load Utility: generate summary report'
msgØ1 = ''
'DISPLAY PANEL(cxsupØØ1)'
if rc = 8 then return
'REMPOP'
/***************************************************************/
/* Alloc work file for utility job (Summary Scan report)      */
/***************************************************************/
```

```
 jobdsn= userid()||'.CXSUJ.WORKJOB'
Trace ?o
ADDRESS TSO
 xx = outtrap(trpØØ.,)
    address tso "delete '"jobdsn"'"
    "alloc da('"jobdsn"') dir(Ø) space(1,1) dsorg(ps)" ,
    "recfm(f,b) lrecl(8Ø) blksize(2792Ø) tracks ",
    "unit(worka) new catalog f(fjob)"
 xx = outtrap(off)
 if rc > Ø then do
    msgØ1 = 'Allocazione di WORKJOB fallita.' ,
            ' Return Code 'rc
    Return
    end
/**************************************************************/
/* Definitions CICS utility job and execution CXSUCØ1Ø EXEC   */
/*  (Summary Scan report)                                     */
/**************************************************************/
 sk.1='//'userid()'# JOB (ESORØØØØ),CLASS=S,MSGCLASS=X,          '
 sk.2='//   MSGLEVEL=(1,1),REGION=8M,NOTIFY=&SYSUID              '
 sk.3='//*------------------------------------------            '
 sk.4='//DELETE EXEC  PGM=IDCAMS                                 '
 sk.5='//SYSPRINT DD  SYSOUT=*                                   '
 sk.6='//SYSIN    DD  *                                          '
 sk.7=' DELETE 'userid()'.CICS.SCANLOAD.SUMOUT NONVSAM           '
 sk.8=' DELETE 'userid()'.CICS.SCANLOAD.SUMMODS NONVSAM          '
 sk.9='/*'
sk.1Ø='/*'
sk.11='//*'
sk.12='//DEFINE   EXEC PGM=IEFBR14                               '
sk.13='//SYSPRINT DD  SYSOUT=*                                   '
sk.14='//SYSOUT   DD  SYSOUT=*                                   '
sk.15='//CSULSUM  DD  DISP=(NEW,CATLG,DELETE),                   '
sk.16='//             DSN='userid()'.CICS.SCANLOAD.SUMMODS,      '
sk.17='//             UNIT=WORKA,                                '
sk.18='//             DCB=(DSORG=PS,BLKSIZE=8ØØØ,RECFM=FB,LRECL=8Ø), '
sk.19='//             SPACE=(CYL,(1,1))                          '
sk.2Ø='//*                                                       '
sk.21="//SCAN     EXEC PGM=CAULMS,PARM='$SUMMARY'                "
sk.22='//STEPLIB  DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR         '
sk.23='//INPUT    DD DSN='csusrin',DISP=SHR                      '
sk.24='//SYSPRINT DD DSN='userid()'.CICS.SCANLOAD.SUMOUT,        '
sk.25='//            DISP=(NEW,CATLG,DELETE),                    '
sk.26='//  UNIT=WORKA,SPACE=(CYL,(1Ø,1Ø),RLSE),                  '
sk.27='//   DCB=(DSORG=PS,BLKSIZE=121ØØ,RECFM=FB,LRECL=121)      '
sk.28='//AFFMOD    DD DSN='userid()'.CICS.SCANLOAD.SUMMODS,DISP=SHR '
sk.29='//DETAIL   DD DUMMY                                       '
sk.3Ø='//* ------------------------------------------------ *   '
sk.31='//LABØ      IF (SCAN.RC EQ Ø) THEN                        '
sk.32='//COPYTMP EXEC PGM=IEBCOPY                                '
sk.33='//SYSPRINT  DD  SYSOUT=*                                  '
sk.34='//SYSUT3    DD  UNIT=VIO,SPACE=(CYL,(5,1))                       '
sk.35='//SYSUT4    DD  UNIT=VIO,SPACE=(CYL,(5,1))                       '
sk.36='//INP1      DD  DISP=SHR,DSN=ISP.SISPTENU                        '
sk.37='//OUT1      DD  DSN=&&TENU,DISP=(,PASS),SPACE=(CYL,(1,1,1Ø)),    '
sk.38='//          UNIT=WORKA                                           '
sk.39='//SYSIN     DD *                                                 '
sk.4Ø=' COPY OUTDD=OUT1,INDD=INP1                                       '
sk.41='//LABØEND  ENDIF                                                 '
sk.42='//* ------------------------------------------------- *        '
sk.43='//LAB1      IF (COPYTMP.RC EQ Ø) THEN                            '
sk.44='//CSUEXEC  EXEC PGM=IKJEFTØ1,DYNAMNBR=15Ø                        '
sk.45='//SYSTSPRT  DD  SYSOUT=*                                         '
sk.46='//SYSPROC   DD  DISP=SHR,DSN=ZZNS611.JCLLIB                      '
sk.47='//ISPLOG    DD  DUMMY                                            '
sk.48='//ISPPROF   DD  DISP=(,DELETE,DELETE),SPACE=(CYL,(1,1,1Ø)),      '
sk.49='//   DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB,DSORG=PO),DSN=&&PROF,   '
sk.5Ø='//          UNIT=WORKA                                           '
sk.51='//*                                                              '
sk.52='//ISPPLIB   DD  DISP=SHR,DSN=ISP.SISPPENU                        '
sk.53='//ISPMLIB   DD  DISP=SHR,DSN=ISP.SISPMENU                        '
sk.54='//ISPTLIB   DD  DISP=(OLD,PASS),DSN=&&TENU                       '
sk.55='//ISPSLIB   DD  DISP=SHR,DSN=ISP.SISPSLIB                        '
sk.56='//SYSTSIN   DD  *                                                '
sk.57='PROF WTPMSG MSGID                                                '
sk.58=' ISPSTART CMD(CXSUCØ1Ø)                                          '
sk.59='//LAB1END  ENDIF                                                 '
sk.6Ø='//*                                                              '
sk.Ø=6Ø
/***************************************************************/
/* Write and submit utility job (Summary Scan report)         */
/***************************************************************/
 "execio * diskw "fjob" (stem sk. finis"
 xx = outtrap(trpØ8.,)
    address tso "submit '"jobdsn"'"
 xx = outtrap(off)
 if rc > Ø then do
    msgØ1 = 'Submit of Generate Summary Report function failed.' ,
            ' Return Code 'rc
    end
           else do
    msgØ1 = 'Verify the execution of Generate job and then ' ,
            ' execute function display.'
 end
"free fi(fjob)"
/***************************************************************/
/* Delete work file for utility job (Summary Scan report)     */
/***************************************************************/
 xx = outtrap(trpØ9.,)
 address tso "delete '"jobdsn"'"
 xx = outtrap(off)
Return
```

```
/***************************************************************/
/* Function display to summary Scan report                     */
/***************************************************************/
Exec_Function2:
tt = Ø
func2 = ''
y = ''
'VPUT Y PROFILE'
/***************************************************************/
/* It verifies that the report is available (Summary Scan report) */
/***************************************************************/
dsnrep = userid()||'.CICS.SCANLOAD.SUMOUT'
ADDRESS TSO
  dd=OUTTRAP(dd.)
  "LISTDS '"dsnrep"' status"
  dd=OUTTRAP('OFF')
  zz = rc
  if zz =4 then do
                  typfunc = 'Noreport'
                  Call CSUL_EventLog
                  Return
                 End
  if zz =8 then do
                  typfunc = 'Allocazione'
                  Call CSUL_EventLog
                  Return
                 End
  if zz ¬= Ø & zz ¬= 4 & zz ¬= 8 then do
                  typfunc = 'Errore'
                  Call CSUL_EventLog
                  Return
                 End
/***************************************************************/
/* Alloc and read summary report (Summary Scan report)         */
/***************************************************************/
  dd=OUTTRAP(dd.)
  "ALLOC DA('"dsnrep"') F(CSULRPT) SHR REUSE"
  dd=OUTTRAP('OFF')
   zz = rc
   if zz¬=Ø then do
                  typfunc = 'Allocazione'
                  Call CSUL_EventLog
                  Return
                 End
            else do
                  ADDRESS TSO
                  dd=OUTTRAP(dd.)
                  "EXECIO * DISKR csulrpt (STEM recr. FINIS"
                  dd=OUTTRAP('OFF')
                  zz = rc
                  if zz¬=Ø then do
```

```
                                typfunc = 'Lettura'
                                Call CSUL_EventLog
                                Return
                               End
                 End
Trace ?o
/**************************************************************/
/* To prepare ISPF table to display summary Scan report      */
/**************************************************************/
tabella = '@@CSUL@@'
ADDRESS ISPEXEC 'TBOPEN 'tabella' NOWRITE'
 if rc=8 then,
   ADDRESS ISPEXEC 'TBCREATE 'tabella ,
   'NAMES(MODULE LANG AFFIN POSTS COMM) NOWRITE'
   msgØ1  = ''
   do i=1 to recr.Ø
      word1  = word(recr.i,1)
      module = substr(recr.i,2,8)
      lang   = substr(recr.i,26,8)
      affin  = substr(recr.i,39,1Ø)
      posts  = substr(recr.i,52,1Ø)
      comm   = substr(recr.i,65,25)
      if module ¬= '========' & word1 ¬= 'Total' Then Do
                               ADDRESS ISPEXEC 'TBADD 'tabella
                               iterate
                              end
      if module = '========' | word1 = 'Total' then do
                               tt = tt + 1
                               rectt.tt = recr.i
                               iterate
                              end
   end
   rectt.Ø = tt
   ADDRESS ISPEXEC 'TBTOP 'tabella
   csrr=Ø
   DO forever
      ADDRESS ISPEXEC
     'ADDPOP ROW(Ø) COLUMN(Ø)'
     'VGET ZKEYS PROFILE'
      ZWINTTL='Cics Scanner Load Utility: display Summary report'
      ADDRESS ISPEXEC 'TBDISPL 'tabella,
      ' PANEL(CXSUPØØ2)',
      ' CSRROW('csrr') AUTOSEL(NO)'
      if RC=8 then leave
      end
ADDRESS ISPEXEC 'TBDELETE 'tabella
ADDRESS ISPEXEC 'TBEND 'tabella
/**************************************************************/
/* To prepare ISPF table to display totals statistics        */
/*  (Summary Scan report)                                     */
/**************************************************************/
```

```
tabella = '@@CSULT@'
ADDRESS ISPEXEC 'TBOPEN 'tabella' NOWRITE'
 if rc=8 then,
   ADDRESS ISPEXEC 'TBCREATE 'tabella ,
   'NAMES(LTOT) NOWRITE'
   msgØ1  = ''
   do g=1 to rectt.Ø
     ltot = rectt.g
     ADDRESS ISPEXEC 'TBADD 'tabella
   end
   ADDRESS ISPEXEC 'TBTOP 'tabella
   csrr=Ø
   DO forever
      ADDRESS ISPEXEC
     'ADDPOP ROW(Ø) COLUMN(Ø)'
     'VGET ZKEYS PROFILE'
      ZWINTTL='Cics Scanner Load Utility: Display Summary report'
      ADDRESS ISPEXEC 'TBDISPL 'tabella,
      ' PANEL(CXSUPØØ3)',
      ' CSRROW('csrr') AUTOSEL(NO)'
      if RC=8 then leave
      end
ADDRESS ISPEXEC 'TBDELETE 'tabella
ADDRESS ISPEXEC 'TBEND 'tabella
ADDRESS TSO
dd=OUTTRAP(dd.)
 "FREE F(CSULRPT)"
dd=OUTTRAP('OFF')
Return
/**************************************************************/
/* Function generation to Detail Scan report                  */
/**************************************************************/
Exec_Function3:
func3 = ''
w = ''
'VPUT W PROFILE'
Do forever
ADDRESS ISPEXEC
'ADDPOP ROW(Ø) COLUMN(Ø)'
'VGET ZKEYS PROFILE'
ZWINTTL='Cics Scanner Load Utility: generate detail report'
msgØ1 = ''
'DISPLAY PANEL(cxsupØ31)'
if rc = 8 then return
'REMPOP'
/**************************************************************/
/* Alloc work file for utility job (Detail Scan report)       */
/**************************************************************/
 jobdsn= userid()||'.CXSUJ.WORKJOB'
ADDRESS TSO
 xx = outtrap(trpØØ.,)
```

```
    address tso "delete '"jobdsn"'"
    "alloc da('"jobdsn"') dir(Ø) space(1,1) dsorg(ps)" ,
    "recfm(f,b) lrecl(8Ø) blksize(2792Ø) tracks ",
    "unit(worka) new catalog f(fjob)"
 xx = outtrap(off)
 if rc > Ø then do
    msgØ1 = 'Allocazione di WORKJOB fallita.' ,
            ' Return Code 'rc
    Return
    end
/***************************************************************/
/* Definitions CICS utility job and execution CXSUCØ2Ø EXEC    */
/*     (Detail Scan report)                                    */
/***************************************************************/
 sk.1='//'userid()'# JOB (ESORØØØØ),CLASS=S,MSGCLASS=X,            '
 sk.2='//   MSGLEVEL=(1,1),REGION=8M,NOTIFY=&SYSUID                 '
 sk.3='//*------------------------------------------------         '
 sk.4='//DELETE EXEC  PGM=IDCAMS                                    '
 sk.5='//SYSPRINT DD  SYSOUT=*                                      '
 sk.6='//SYSIN    DD  *                                             '
 sk.7=' DELETE 'userid()'.CICS.SCANLOAD.DETOUT NONVSAM               '
 sk.8='/*'
 sk.9='/*'
sk.1Ø='//*'
sk.11="//SCAN     EXEC PGM=CAULMS,PARM='$DETAIL'                     "
sk.12='//STEPLIB  DD DSN=CICSTS13.CICS.SDFHLOAD,DISP=SHR              '
sk.13='//INPUT    DD DSN='csudrin',DISP=SHR                           '
sk.14='//SYSPRINT DD DSN='userid()'.CICS.SCANLOAD.DETOUT,             '
sk.15='//            DISP=(NEW,CATLG,DELETE),                         '
sk.16='//  UNIT=WORKA,SPACE=(CYL,(1Ø,1Ø),RLSE),                       '
sk.17='//   DCB=(DSORG=PS,BLKSIZE=121ØØ,RECFM=FB,LRECL=121)           '
sk.18='//DETAIL   DD DSN='csudrdt',DISP=SHR                           '
sk.19='//AFFMOD   DD DUMMY                                            '
sk.2Ø='//* ---------------------------------------------------- *    '
sk.21='//LABØ      IF (SCAN.RC EQ Ø) THEN                             '
sk.22='//COPYTMP EXEC PGM=IEBCOPY                                     '
sk.23='//SYSPRINT  DD  SYSOUT=*                                       '
sk.24='//SYSUT3    DD  UNIT=VIO,SPACE=(CYL,(5,1))                     '
sk.25='//SYSUT4    DD  UNIT=VIO,SPACE=(CYL,(5,1))                     '
sk.26='//INP1      DD  DISP=SHR,DSN=ISP.SISPTENU                      '
sk.27='//OUT1      DD  DSN=&&TENU,DISP=(,PASS),SPACE=(CYL,(1,1,1Ø)),  '
sk.28='//          UNIT=WORKA                                         '
sk.29='//SYSIN     DD *                                               '
sk.3Ø=' COPY OUTDD=OUT1,INDD=INP1                                     '
sk.31='//LABØEND  ENDIF                                               '
sk.32='//* ---------------------------------------------------- *    '
sk.33='//LAB1      IF (COPYTMP.RC EQ Ø) THEN                          '
sk.34='//CSUEXEC  EXEC PGM=IKJEFTØ1,DYNAMNBR=15Ø                      '
sk.35='//SYSTSPRT  DD  SYSOUT=*                                       '
sk.36='//SYSPROC   DD  DISP=SHR,DSN=ZZNS611.JCLLIB                    '
sk.37='//ISPLOG    DD  DUMMY                                          '
sk.38='//ISPPROF   DD  DISP=(,DELETE,DELETE),SPACE=(CYL,(1,1,1Ø)),    '
sk.39='//   DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB,DSORG=PO),DSN=&&PROF, '
sk.4Ø='//          UNIT=WORKA                                          '
sk.41='//*                                                             '
sk.42='//ISPPLIB   DD  DISP=SHR,DSN=ISP.SISPPENU                       '
sk.43='//ISPMLIB   DD  DISP=SHR,DSN=ISP.SISPMENU                       '
sk.44='//ISPTLIB   DD  DISP=(OLD,PASS),DSN=&&TENU                      '
sk.45='//ISPSLIB   DD  DISP=SHR,DSN=ISP.SISPSLIB                       '
sk.46='//SYSTSIN   DD  *                                               '
sk.47='PROF WTPMSG MSGID                                               '
sk.48=' ISPSTART CMD(CXSUCØ2Ø)                                         '
sk.49='//LAB1END  ENDIF                                                '
sk.5Ø='//*                                                             '
sk.Ø=5Ø
/***************************************************************/
/* Write and submit utility job  (Detail Scan report)         */
/***************************************************************/
 "execio * diskw "fjob" (stem sk. finis"
 xx = outtrap(trp1Ø.,)
    address tso "submit '"jobdsn"'"
 xx = outtrap(off)
 if rc > Ø then do
    msgØ1 = 'Submit Generate Detail Report function failed.' ,
            ' Return Code 'rc
    end
           else do
    msgØ1 = 'Verify the execution of Generate job and then ' ,
            ' execute function display.'
    end
"free fi(fjob)"
/***************************************************************/
/* Delete work file for utility job   (Detail Scan report)    */
/***************************************************************/
 xx = outtrap(trp11.,)
 address tso "delete '"jobdsn"'"
 xx = outtrap(off)
Return
/***************************************************************/
/* Function display to Detail Scan report                     */
/***************************************************************/
Exec_Function4:
tt = Ø
func4 = ''
k = ''
'VPUT K PROFILE'
/***************************************************************/
/* It verify if the report is available  (Detail Scan report) */
/***************************************************************/
dsnrep = userid()||'.CICS.SCANLOAD.DETOUT'
ADDRESS TSO
  dd=OUTTRAP(dd.)
  "LISTDS '"dsnrep"' status"
  dd=OUTTRAP('OFF')
  zz = rc
  if zz =4 then do
                typfunc = 'Noreport'
                Call CSUL_EventLog
                Return
               End
  if zz =8 then do
                typfunc = 'Allocazione'
                Call CSUL_EventLog
                Return
               End
  if zz ¬= Ø & zz ¬= 4 & zz ¬= 8 then do
                typfunc = 'Errore'
                Call CSUL_EventLog
                Return
               End
/***************************************************************/
/* Alloc and read Detail Scan report                           */
/***************************************************************/
  dd=OUTTRAP(dd.)
  "ALLOC DA('"dsnrep"') F(CSULRPT) SHR REUSE"
  dd=OUTTRAP('OFF')
   zz = rc
   if zz¬=Ø then do
                typfunc = 'Allocazione'
                Call CSUL_EventLog
                Return
               End
          else do
                ADDRESS TSO
                dd=OUTTRAP(dd.)
                "EXECIO * DISKR csulrpt (STEM recr. FINIS"
                dd=OUTTRAP('OFF')
                zz = rc
                if zz¬=Ø then do
                               typfunc = 'Lettura'
                               Call CSUL_EventLog
                               Return
                              End
               End
/***************************************************************/
/* To prepare ISPF table to display Detail Scan report         */
/***************************************************************/
Trace ?o
tabella = '@@CSUD@@'
ADDRESS ISPEXEC 'TBOPEN 'tabella' NOWRITE'
 if rc=8 then,
   ADDRESS ISPEXEC 'TBCREATE 'tabella ,
   'NAMES(ROW) NOWRITE'
 msgØ1  = ''
 do i=1 to recr.Ø
    row = copies(' ',78)
    fcntl  = substr(recr.i,2,8)
    if fcntl = 'Module N' then do
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                row = substr(recr.i,2,58)
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                iterate
                               end
    if fcntl = 'Offset  ' | fcntl = '--------' then do
                                field1 = substr(recr.i,2,9)
                                field2 = substr(recr.i,75,47)
                                row = field1||field2
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                iterate
                               end
    if fcntl = 'Total po' then do
                                row = substr(recr.i,2,44)
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                iterate
                               end
if fcntl ¬= 'Module N' & fcntl ¬= 'Offset  ' & fcntl ¬= 'Total po' & ,
   fcntl ¬= '--------' & fcntl ¬= '========' & fcntl ¬= 'Total mo' & ,
   fcntl ¬= 'Total CI' & fcntl ¬= '  Total ' Then do
                                       field = substr(recr.i,75,46)
                                       row = fcntl||' '||field
                                       ADDRESS ISPEXEC 'TBADD 'tabella
                                       iterate
                                      end
   if fcntl = '========' | fcntl = 'Total mo' | fcntl = 'Total CI' | ,
       fcntl = '  Total ' Then do
                              tt = tt + 1
                              rectt.tt = recr.i
                              iterate
                             end
 end
 rectt.Ø = tt
 ADDRESS ISPEXEC 'TBTOP 'tabella
 csrr=Ø
 DO forever
    ADDRESS ISPEXEC
   'ADDPOP ROW(Ø) COLUMN(Ø)'
   'VGET ZKEYS PROFILE'
    ZWINTTL='Cics Scanner Load Utility: display Detail report'
    ADDRESS ISPEXEC 'TBDISPL 'tabella,
    ' PANEL(CXSUPØ42)',
    ' CSRROW('csrr') AUTOSEL(NO)'
    if RC=8 then leave
    end
```

```
ADDRESS ISPEXEC 'TBDELETE 'tabella
ADDRESS ISPEXEC 'TBEND 'tabella
/***************************************************************/
/* To prepare ISPF table to display totals statistics         */
/*       (Detail Scan report)                                  */
/***************************************************************/
tabella = '@@CSUDT@'
ADDRESS ISPEXEC 'TBOPEN 'tabella' NOWRITE'
 if rc=8 then,
   ADDRESS ISPEXEC 'TBCREATE 'tabella ,
   'NAMES(RTOT) NOWRITE'
   msgØ1  = ''
   do g=1 to rectt.Ø
     rtot = rectt.g
     ADDRESS ISPEXEC 'TBADD 'tabella
   end
   ADDRESS ISPEXEC 'TBTOP 'tabella
   csrr=Ø
   DO forever
      ADDRESS ISPEXEC
     'ADDPOP ROW(Ø) COLUMN(Ø)'
     'VGET ZKEYS PROFILE'
      ZWINTTL='Cics Scanner Load Utility: Display Detail report'
      ADDRESS ISPEXEC 'TBDISPL 'tabella,
      ' PANEL(CXSUPØ43)',
      ' CSRROW('csrr') AUTOSEL(NO)'
      if RC=8 then leave
      end
ADDRESS ISPEXEC 'TBDELETE 'tabella
ADDRESS ISPEXEC 'TBEND 'tabella
ADDRESS TSO
dd=OUTTRAP(dd.)
 "FREE F(CSULRPT)"
dd=OUTTRAP('OFF')
Return
/***************************************************************/
/* Management errors routine                                   */
/***************************************************************/
CSUL_EventLog:
Select
when typfunc = 'Allocazione' then
  msgØ1 = 'The allocation report is failed. Notify CICS system support.'
when typfunc = 'Lettura' then
  msgØ1 = 'The read report is failed. Notify CICS system support.'
when typfunc = 'Noreport' then
  msgØ1 = '          Report is not available.'
when typfunc = 'Errore' then
  msgØ1 = '          Report in error. Notify CICS system support.'
Otherwise nop
End
ADDRESS TSO
```

```
dd=OUTTRAP(dd.)
 "FREE F(CSULRPT)"
dd=OUTTRAP('OFF')
Return
```

CXSUC010 EXEC

```
/* REXX */
/* CICS SCANNER LOAD MODULE utility
   C-List CXSUCØ1Ø
   Called by utility job (by generation Summary function).
   It executes:
      - ISPF macro to create Summary Scan report;           */
Trace ?o
dsnrep = userid()||'.CICS.SCANLOAD.SUMOUT'
address ispexec "edit dataset('"dsnrep"') macro(CXSUMØØØ)"
  zz = rc
  if zz ¬= Ø then do
        mess1 = 'ISPF Macro error. Notify CICS system support.'
        say time() mess1
                 End
Exit
```

CXSUC020 EXEC

```
/* REXX */
/* CICS SCANNER LOAD MODULE utility
   C-List CXSUCØ2Ø
   Called by utility job (by generation Detail function).
   It execute:
      - ISPF macro to create Detail Scan report;            */
Trace ?o
dsnrep = userid()||'.CICS.SCANLOAD.DETOUT'
address ispexec "edit dataset('"dsnrep"') macro(CXSUMØ1Ø)"
  zz = rc
  if zz ¬= Ø then do
        mess1 = 'ISPF Macro error. Notify system support.'
        say time() mess1
                 End
Exit
```

CXSUM000 ISPF MACRO

```
/* REXX */
/* CICS SCANNER LOAD MODULE utility
   Ispf macro CXSUMØØØ
   Called by CXSUCØ1Ø exec.
   It execute:
      - definition of the Summary Scan report;              */
trace ?o
   address isredit macro
     "isredit change 'Ø' ' ' 1 all"
     "isredit change '1' ' ' 1 all"
     "isredit change '-' ' ' 1 all"
      isredit x all
     "isredit find 'CICS TRANSACTION AFFINITIES' all"
     "isredit find 'LOAD MODULE SCANNER -' all"
     "isredit find 'Module       Module' all"
     "isredit find 'Name         Length' all"
     "isredit find '--------    --------' all"
     "isredit find 'LOAD LIBRARY STATISTICS' all"
     "isredit find '                      ' 1 all"
      isredit del nx all
      isredit save
   isredit end
   return
```

CXSUM010 ISPF MACRO

```
/* REXX */
/* CICS SCANNER LOAD MODULE utility
   Ispf macro CXSUMØ1Ø
   Called by CXSUCØ2Ø exec.
   It execute:
      - definition of the Detail Scan report;               */
trace ?o
   address isredit macro
     "isredit change 'Ø' ' ' 1 all"
     "isredit change '1' ' ' 1 all"
     "isredit change '-' ' ' 1 all"
      isredit x all
     "isredit find 'CICS TRANSACTION AFFINITIES' all"
     "isredit find 'LOAD MODULE SCANNER -' all"
     "isredit find 'LOAD LIBRARY STATISTICS' all"
     "isredit find '                      ' 1 all"
      isredit del nx all
      isredit save
   isredit end
   return
```

ISPF PANELS

Panel CXSUP000

```
)ATTR
 %   TYPE(TEXT) INTENS(HIGH) SKIP(ON)
 }   TYPE(TEXT) HILITE(BLINK) COLOR(GREEN)
 !   TYPE(INPUT) CAPS(ON) JUST(LEFT) HILITE(REVERSE)
 {   TYPE(INPUT) CAPS(ON) JUST(LEFT) PAD(_)
 '   COLOR(turq) TYPE(TEXT)
 ?   COLOR(white) TYPE(TEXT)
 <   COLOR(blue) TYPE(TEXT) intens(high)
 >   COLOR(yellow) TYPE(TEXT) intens(high)
 @ TYPE(INPUT) INTENS(HIGH) PAD('_') CAPS(ON) hilite(reverse)
 # TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(yellow)
 £ TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(turq)
 \ type(TEXT) intens(HIGH) COLOR(green) hilite(reverse)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
)BODY EXPAND(//) WINDOW(75 22) SMSG(VIDEO)
+$VIDEO
+
+                                        User.....: &ZUSER
<      *******+ ?*******+                Date.....: &ZDAY &ZMONTH &ZYEAR
<    **      *?**       *+               Time.....: &ZTIME
<   **+      ?**'                        Appl.....: Cics SCANLoad
<  **+      ?  **'         ********+
< **+      ?    *****'    **     **+
< *      *?         **'  **     **+
< *******?           **'*********+             \ SELECT FUNCTION +
> Cics+ ?  *       **' **    **+
+       ?   *******'  **      **+  Generate Scanner Summary Report.: #X+
+         >Scanner +'**        **+ Display Scanner Summary Report..: #Y+
+                  '**          **+
+                  >Report+        Generate Scanner Detail Report..: £W+
+                                  Display Scanner Detail Report...: £K+
+
+ $msgØ1                                                               +
+
+
+ PF1=Help  PF3=Exit  ENTER=Continue
)INIT
 .CURSOR = X
 &ZCMD=' '
  VGET (X,Y,W,K) PROFILE
  .HELP = CXSUPØØH
  &ZHTOP = CXSUPØØH
  &ZHINDEX = CXSUPØØH
)PROC
IF (&X = ' ' AND &Y = ' ' AND &W = ' ' AND &K = ' ') .MSG = 'CXSUMØØØ'
IF (&X ¬= ' ' AND &Y ¬= ' ' AND &W ¬= ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1'
IF (&X ¬= ' ' AND &Y ¬= ' ' AND &W ¬= ' ' AND &K = ' ') .MSG =
'CXSUMØØ1'
IF (&X ¬= ' ' AND &Y ¬= ' ' AND &W = ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1'
```

```
IF (&X ¬= ' ' AND &Y = ' ' AND &W ¬= ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1'
IF (&X = ' ' AND &Y ¬= ' ' AND &W ¬= ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1'
IF (&X = ' ' AND &Y = ' ' AND &W ¬= ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1 '
IF (&X = ' ' AND &Y ¬= ' ' AND &W = ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1 '
IF (&X ¬= ' ' AND &Y ¬= ' ' AND &W = ' ' AND &K = ' ') .MSG =
'CXSUMØØ1 '
IF (&X ¬= ' ' AND &Y = ' ' AND &W ¬= ' ' AND &K = ' ') .MSG =
'CXSUMØØ1 '
IF (&X = ' ' AND &Y ¬= ' ' AND &W ¬= ' ' AND &K = ' ') .MSG =
'CXSUMØØ1 '
IF (&X ¬= ' ' AND &Y = ' ' AND &W = ' ' AND &K ¬= ' ') .MSG =
'CXSUMØØ1 '
VPUT (X,Y,W,K) PROFILE
)END
```

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

Espedito Morvillo
Systems Programmer (Italy)

CICS news

IBM has announced Version 3.2 of its CICSVR, allowing control of the recovery process at the group level, providing finer control of recovery and improved use of storage. It incorporates usability improvements and maintenance identified from the use of CICSVR V3.1.

Included in V3.2 is a new Group function for selecting VSAM data sets for recovery and building a recovery job. It works with DFSMS ISMF to group data sets for recovery according to each site's criteria.

A Data Set List feature provides an alternative way to group VSAM spheres. Users can enter the name of a dataset that contains a list of VSAM spheres to be recovered. There's also the ability to specify one set of recovery parameters and generate a recovery job for multiple VSAM spheres using those recovery parameters.

Use of VSAM storage has been improved and simplified, so the software can now request and use appropriate VSAM local shared resources during recovery.

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

* * *

Jacada has partnered with Computer Associates to provide enhanced Web-to-host integration to complement CA's data integration and management tools, including Advantage EDBC, Advantage CA-IDMS, and Advantage CA-Datacom.

The partnership also positions Jacada, it says, as an option for sites using other CA tools that require mainframe application access such as CleverPath Portal.

Jacada software extends legacy applications into modern computing environments through automated integration, Web-to-host, modernization, extension, and terminal emulation products. Its software enables use of System/390, zSeries, CICS, etc with Web functionality, including support for Java, XHTML, XML, EJB, J2EE, .NET, and Web services.

The company has provided modernization and Web-enabling tools for a number of CA applications on an OEM basis for the past six years.

For further information contact:
Jacada, 400 Perimeter Center Terrace, Suite 100, Atlanta, GA 30346, USA.
Tel: (770) 352 1300.
URL: http://www.jacada.com/News/PR163.htm.

* * *

Dignus has begun shipping its first 64-bit compiler for z/Architecture, notably the Systems/C and Systems/C++ Version 1.60. They maintain support for earlier architectures, and exploit z/Architecture allowing programs to execute in 24, 31, or 64-bit, as well as allowing all dynamic storage to be allocated above the 31-bit bar.

In addition to major processor-related enhancements, this software also includes a CICS C pre-processor and support for a number of ANSI 99 language features.

Dignus, 8354 Six Forks Road, Suite 201, Raleigh, NC 27615, USA.
Tel: (919) 676 0847.
URL: http://www.dignus.com/dcc.

* * *