210

CICS update

May 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

Disclaimer

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

Contributions

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

Printed in England.

CICS Resource Management using CICSPlex/SM

This article describes the implementation of a CICS resource management system using the Business Application Services (BAS) component of CICSPlex/SM at a large Swiss company. The requirement arose from an infrastructure design change for our major new CICS project – a shift from running all user applications under one CICS transaction ID to running each application component (Business Service) under its own transid. So from managing at most a few dozen transids and their associated DB2TRAN entries, we had to migrate to several thousand transids in the space of a few months.

We made two decisions. The first was to use BAS to manage the resources. This was a logical step since we were then in the process of implementing CICSPlex/SM and Workload Management. Resources that we intended to workload manage using CICSPlex/SM facilities should themselves be managed by CICSPlex/SM. BAS gave us several big advantages over the existing CSD system. The most important of these was the ability to quickly and accurately disseminate new resources or changes in resources over an entire CICSPlex. We have two CICSPlexes – X-Plex, which encompasses all our development and test environments, and Prod-Plex, for production. The second advantage was having a single point of control for all of our dozens of test and development environments. The third advantage was the availability of a comprehensive API for BAS, which is, in our view, much superior to and less error-prone than the equivalent facilities available for the CSD. Moreover, this API was accessible from REXX and, after a few teething problems and some careful reading of the manuals, we found it fairly easy to use.

Our second decision was to develop a TSO/REXX/ISPF-based system – a CICS Resource Management System (CRMS) – to manage the resources. Using REXX – and the availability of the BAS API – allowed us to develop the application quickly and flexibly. A very important design decision was to use the DB2 V7

REXX/DB2 interface and hold all persistent data in DB2 tables. We use ISPF tables only as work tables. This greatly simplified the coding and also improved our back-up, error handling, and data management. The use of DB2 tables also allowed us to interface more easily with other systems. We had thought to rewrite some of the more static parts of the system in COBOL in order to improve performance, but so far performance remains acceptable.

These decisions were proved correct when we were able to develop a functional prototype CRMS, capable of productive work, in less than three months. The first step in the implementation was a one-time migration of all CSD-managed user transactions to CRMS – IBM and third-party product transids were not migrated and remain in the CSD. This went smoothly with only a few very minor problems. With the exception of some administrative and implementation work to be done for Prod-Plex, CRMS is now in full 'production'.

The CRMS workflow is as follows. There are two functional roles – coordinator and approver. The CRMS coordinator receives an application for a new Business Service from the developer or project leader and enters it into the 'to-be-approved' facility of CRMS. The request is checked for consistency, conformity with standards, etc. The CRMS approver processes the 'to-be-approved' list regularly or on request. At approval time, CRMS does the following:

- Assigns a CICS transid to the Business Service.
- Defines a DB2TRAN entry for this transid.
- Defines and propagates these definitions throughout X-Plex. They become active at the next CICS restart (daily).

Developers and other interested parties can use CRMS in read-only mode to track the status of their request and to find out the name of 'their' CICS transid. Operators and support staff can use CRMS to find out the name of the Business Service and Business Service owner (eg after a CICS abend). All requests, rejections, and approvals are logged in the CRMS audit log.

CRMS is the first system to our knowledge which uses the CICSPlex/SM BAS API to manage CICS resources.

With grateful acknowledgements to Peter Davis, our CICSPlex/SM guru, and Ian MacPhee, who did most of the coding and implementation.

David Roth
Senior CICS Product Engineer (Switzerland)

CICS Transaction Gateway Version 5 and CICS Transaction Server Version 2.2: part 2 – Architecture for connecting an OS/390 or z/OS CICS Transaction Gateway to CICS

INTRODUCTION

In this second of a series of articles on the CICS Transaction Gateway, I am going to describe how to arrange a CICS Transaction Gateway running on MVS (ie OS/390 or z/OS) to contact a CICS region. I am assuming usage of the CICS Transaction Gateway Version 5 and CICS Transaction Server Version 2.2.

I use the term CTG to refer to CICS Transaction Gateway Version 5 and CICS to mean CICS Transaction Server Version 2.2. I use MVS to mean either OS/390 or z/OS.

A WORKSTATION CTG CONFIGURATION

The previous article (see last month's *CICS Update – CICS Transaction Gateway Version 5 and CICS Transaction Server Version 2.2: part 1 – Architecture for connecting a workstation CICS Transaction Gateway to CICS*) explained how to arrange a CICS Transaction Gateway running on a workstation (or

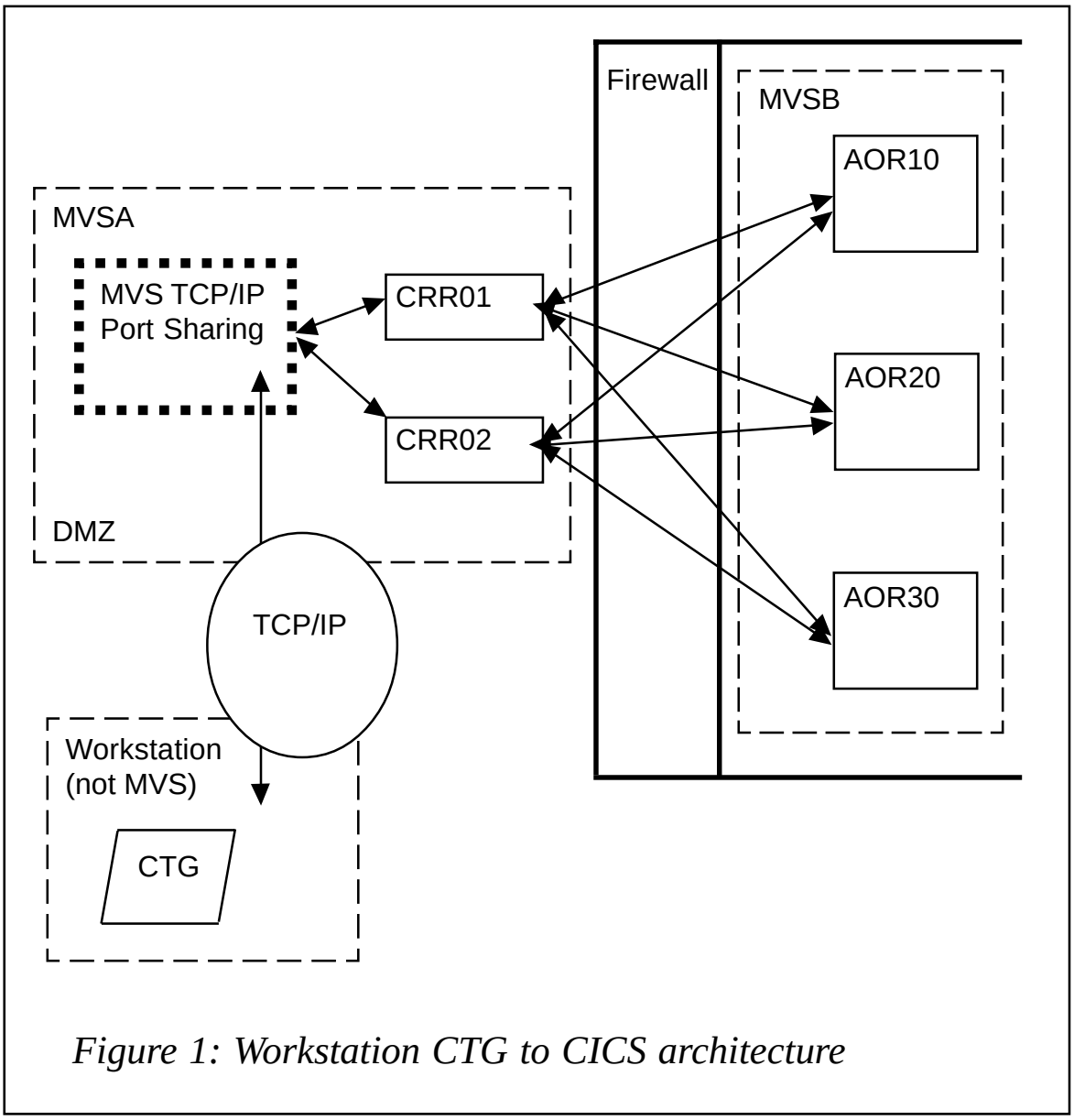

Figure 1: Workstation CTG to CICS architecture

general non-MVS environment) to contact CICS. Figure 1 shows my preferred arrangement.

This provides several things:

- The workstation application does not have to know anything about CTG configuration because the CTG only ever talks to a single CICS region (which I called a CICS Routing Region – CRR).
- The workstation application does not communicate with the

CICS region that contains the CICS program to be executed, and so does not care about the location.

- MVS TCP/IP port sharing functions provide load balancing and failover for the CRRs, so guaranteeing availability.
- The CRR uses standard CICS load balancing and routing functions to select an AOR to execute the request, so providing load balancing and failover for the execution of the requested CICS program.
- A DMZ or firewall can be used to supplement secure CICS communication.

The arrangement in Figure 1 provides a secure execution environment with minimal CTG configuration outside the CICS environment. All the usual CICS reliability and performance characteristics apply.

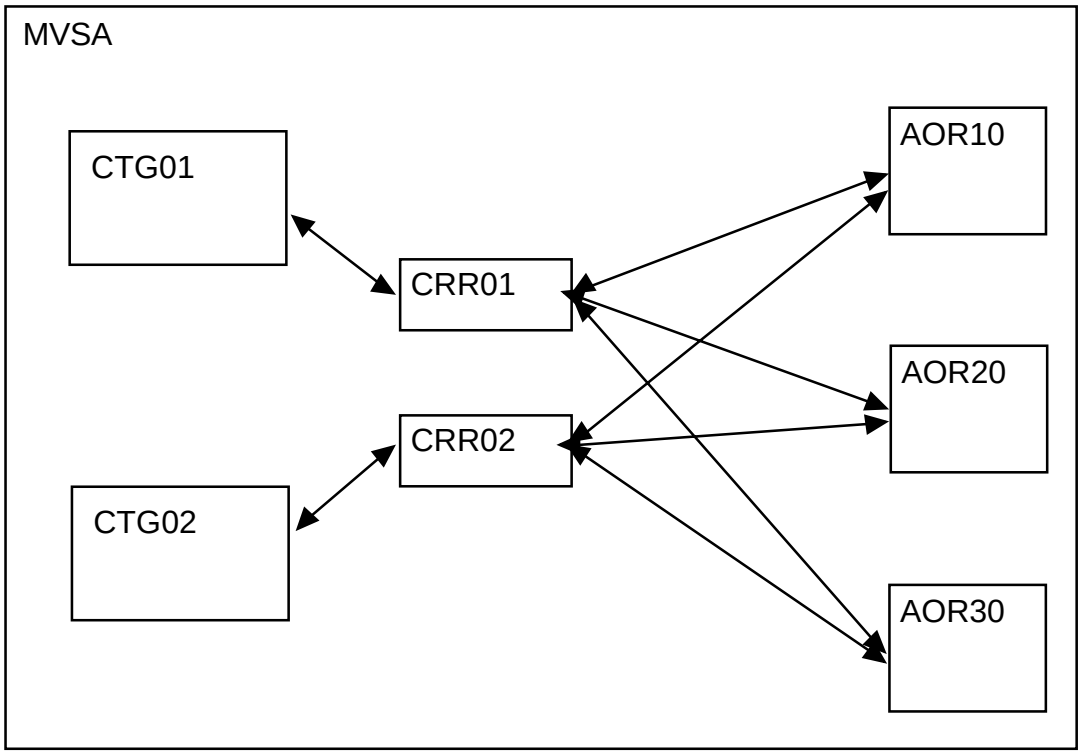

Figure 2: MVS CTGs using EXCI

THE CTG ON MVS

The main thing to understand about a CICS Transaction Gateway running in the MVS environment (OS/390 or z/OS) is that it is a completely different thing from one running in a non-MVS (Windows or a Unix Server) environment.

There are many similarities, but also some significant differences. It is these differences that engender the configuration for CTG usage in MVS.

Use of EXCI

One of the most significant differences between a Workstation (non-MVS) CTG and an MVS CTG is that a CICS protocol is used for communication. This is the EXCI mechanism described in SC34-6031, *CICS External Interfaces Guide*. It is important that you understand the EXCI protocols as my recommended configuration relies on this mechanism.

This means that communication from a CTG to the CRR uses EXCI, which implies that the CTG and the CRR are in the same MVS image. The AORs can be in the same MVS image or in different ones. This is shown in Figure 2, where all items are in the same MVS image.

In a simple case, an MVS CTG communicates only with one CRR (unlike Figure 2). This is not suitable for a production arrangement.

The EXCI exit

In my previous article dealing with a non-MVS CTG, I went into great detail about using MVS TCP/IP port sharing to ensure that the workstation CTG would always be able to communicate with one-of-*n* CRRs for availability and performance reasons. So why does this not apply to an MVS CTG? The CTG is being accessed from an MVS job and not from a remote client, so TCP/IP port sharing is not involved. Consequently, another approach is required.

Early versions of the MVS CTG (those prior to Version 3.1.2) had a performance restriction associated with EXCI Pipe initialization. The EXCI Pipe was allocated at the start of a flow and deallocated at the end. The EXCI Exit (DFHXCURM) is driven each time an EXCI Pipe is allocated, so it could select a suitable CRR upon each flow. In effect the MVS CTG could do its own routing.

Performance has improved for recent versions of the MVS CTG. Once an EXCI Pipe has been allocated (from a CTG to a CRR), it is never freed. This avoids significant MVS overhead because a subsequent flow can reuse the Pipe. It also means that the EXCI exit is driven only on those occasions when a new Pipe is obtained and not for each flow. A side effect is that the EXCI exit (DFHXCURM) cannot be used to select a CRR on each flow, so removing the ability of the CTG to do its own routing.

…and this means?

The practical upshot is that an MVS CTG should communicate only with a single CRR in normal circumstances. This CRR then selects a suitable AOR to process the request. A second CRR is used only for back-up purposes.

Some justification

Obviously, other configurations are possible.

You can code up the DFHXCURM EXCI exit to select communication pathways, so an MVS CTG can dynamically communicate to any number of CRRs. The exit would implement code to decide which pathway to use. However, there is no standard mechanism available to decide which is the best, and once a pathway has been created there is no way of discarding it or pointing it to another destination (the aforementioned performance improvement).

There is also a practical problem in that the CTG has a physical limit (like any other MVS job) of 100 EXCI connections.

If you are running a batch job using EXCI to access CICS, you would probably either go to a fixed CRR or directly to the relevant

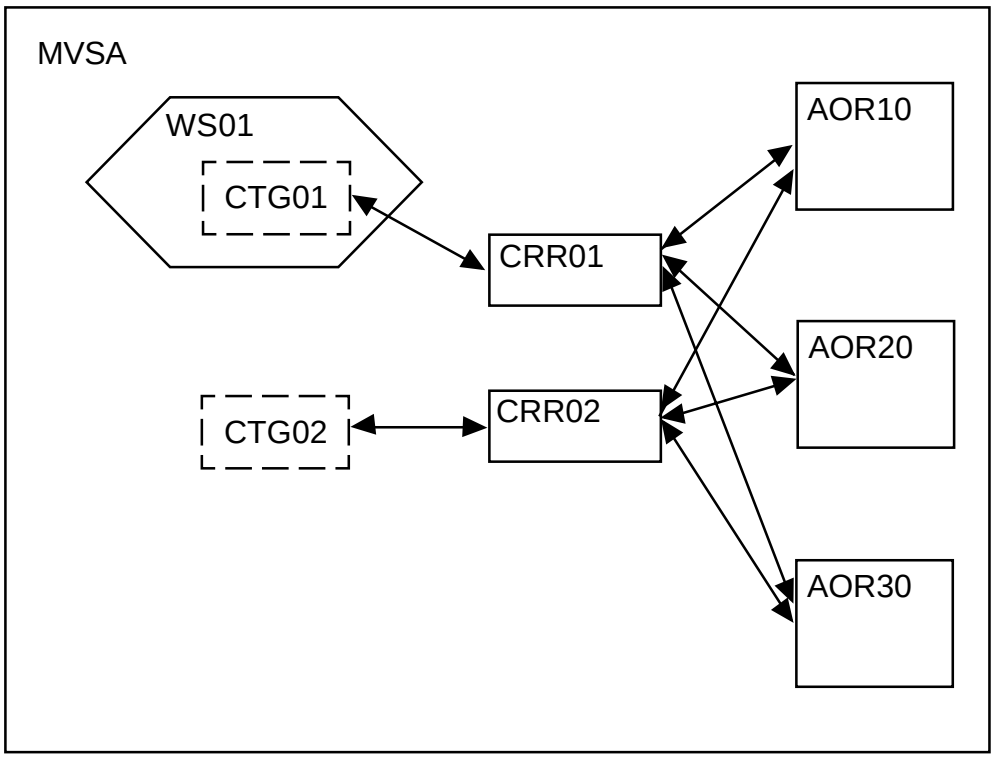

Figure 3: MVS CTGs and WebSphere (simplified)

AOR. (You may have seen my SupportPac enabling EXCI access to CICS from REXX/MVS or an MVS Pipeline.) This is acceptable because the batch job is short-lived and so the problem of EXCI pathways does not arise.

This problem most certainly arises if lots of traffic is being processed from a long-lived source – for example if WebSphere is sending traffic to CICS.

ACCESS FROM WEBSPHERE

A major use of an MVS CTG is to provide access to CICS programs from a server such as WebSphere (running on MVS).

This means that the configuration has to be able to cope with long-running access, which is made on behalf of multiple users.

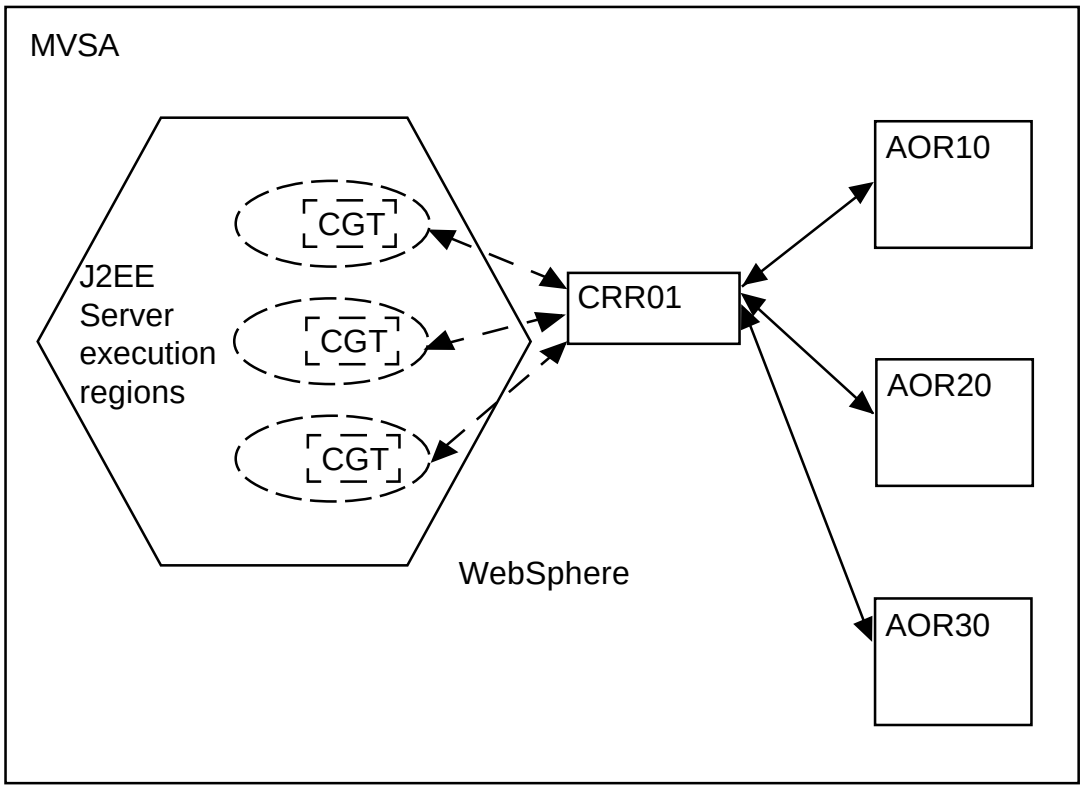

Figure 4: WebSphere J2EE Servers and CTGs

Adding WebSphere

In essence, the framework used for CTG access from WebSphere associates a copy of the CTG with each WebSphere. Figure 3 shows this arrangement (which is a gross simplification!).

At this point, I need to discuss, in a conceptual fashion, what I mean by WebSphere. WebSphere Version 4 consists of a number of regions that act as servers – Daemon Servers, Naming Servers, etc. I am really talking about the WebSphere J2EE Server, which receives HTTP requests and runs servlets or EJBs. Each WebSphere server consists of a control region and what I am going to call execution regions. These execution regions are started by the control region as required to process activity. The discussion in this article proceeds on the crude basis that a WebSphere instance is a set of WebSphere J2EE Server execution regions and throughput is directly related to the

number of these J2EE Server execution regions.

Inside each of these J2EE Server execution regions there is a CTG – each CTG is identically configured. Thus, in Figure 4, the hexagon is a set of WebSphere J2EE Servers and CTGs, which are more properly shown in Figure 5. As all CTGs are identical, they communicate with the same CRR.

In my first article, I discussed using the CTG exit to fix up parameters. Similar facilities are available in the MVS environment (using the EXCI User Replaceable Module DFHXCURM). Consequently, applications running within WebSphere can be shielded from knowing the overall configuration. In particular, they do not need to know the identity of the CRR being used by the CTG running inside the WebSphere J2EE Server execution regions. This means that the WebSphere Application can be deployed into other WebSpheres without any change.

The arrangement of Figure 4 shows that the standard CICS-based mechanisms for failover and load balancing apply between a CRR and the AORs (and so between a WebSphere J2EE Server execution region and the AORs). But what about availability from the view of a user of the WebSphere application? What happens if a CTG-to-CRR connection fails?

MY RECOMMENDED MVS CONFIGURATION

A full answer to this question relies on WebSphere function, which is outside the scope of this article. Let's assume that WebSphere is alive and well, the J2EE Server execution regions are still active, but access to the CRR from the associated CTG has failed.

In this case, the EXCI exit in the CTG will be driven because an attempt to create or use a connection fails. It will have been driven before when the connection to the CRR was initially established (as I'm assuming that the CTG exit is being used to fix up the destination, so avoiding configuration knowledge being imposed on the application).

My recommendation is that the EXCI exit implements code that

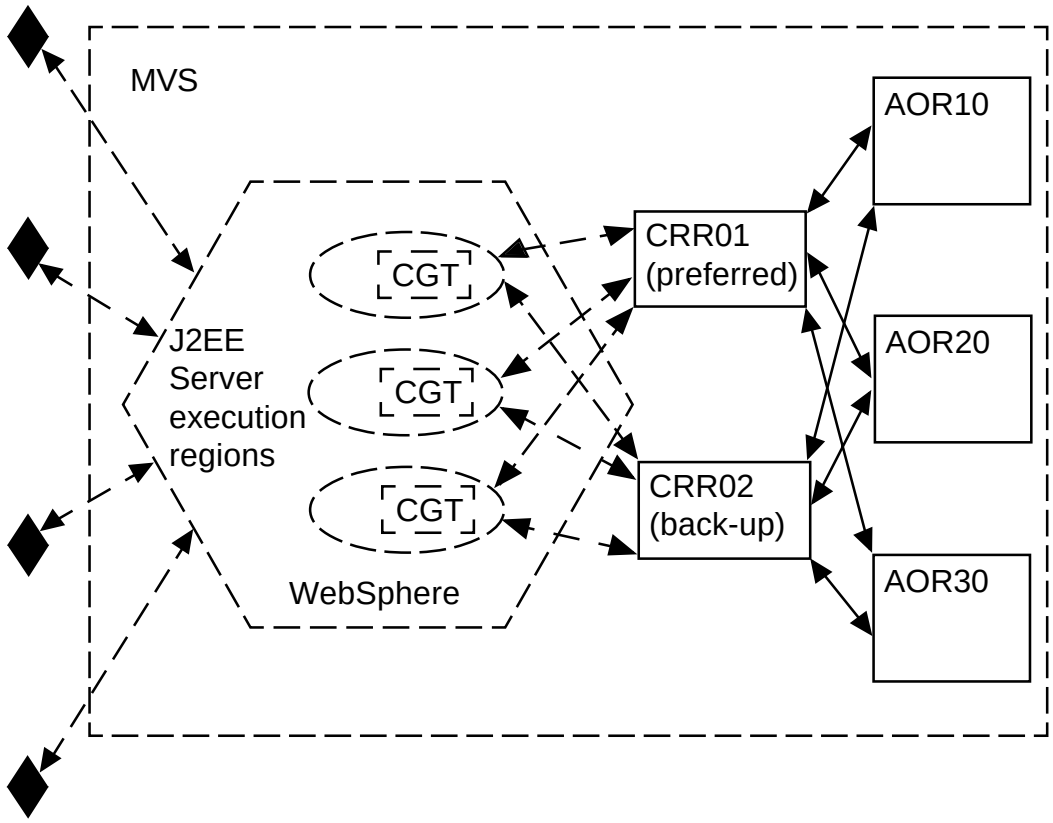

Figure 5: My recommended arrangement for an MVS CTG with WebSphere

has knowledge of the preferred CRR and its back-ups. In the unlikely case of the preferred CRR failing, the EXCI exit should direct traffic to the back-up CRR until the failed CRR restarts. The WebSphere application and WebSphere itself do not have any knowledge of this CRR arrangement nor of a failure having occurred. Although the EXCI exit may have to change when the CRR configuration alters, this is the only thing that has to be amended – WebSphere and its applications using the CTG are unaltered.

The final arrangement

Figure 5 shows my recommended configuration for an MVS CTG associated with WebSphere (there may be a firewall in front of MVSA).

A crucial aspect is that the WebSphere J2EE Server execution region runs its own CTG instance. There is a known trail via the associated CTG to two CRRs, where normal CICS load balancing and failover techniques apply. So, this path starts in each WebSphere J2EE server execution region and ends in a CRR. Only one of these CTG to CRR paths is normally used, the other is for back-up purposes.

SO WHAT HAPPENS IF A CTG OR A CRR DROPS DEAD?

The first effect will be inside WebSphere. The WebSphere J2EE Server will start to slow down, and its control region will start to act. Mechanisms outside the scope of this article will come into play to ensure availability.

As each J2EE Server execution region has its own CTG, a failure in one execution region will not affect any of the others. Consequently, availability is not restricted. A failure in one J2EE Server execution region does not stop traffic in another execution region. This functionality provides an inherently available arrangement.

In the unlikely event of the primary CRR failing, the EXCI exit in the CTGs will notice the path failure and route traffic to the back-up CRR – functionality is still available.

FURTHER THOUGHTS

Can I share CRRs between WebSpheres/CTGs?

It depends on what you mean by sharing. You do not share CTGs in my WebSphere arrangement – instead there are lots of clones around (one per J2EE Server execution region). It is in the nature

of my arrangement that CRRs are shared between these cloned CTGs.

What you really mean is that if one uses other parts of WebSphere not mentioned in this article, is it a good idea to share CRRs? I don't think so. If you are using these facilities, you do not want to introduce an additional point of failure.

Can WebSphere use different CTGs?

No. In my recommended arrangement, the CTG configuration (including the EXCI exit code) is identical. In more complex arrangements, again outside the scope of this article, you can have differently configured CTGs and I would then recommend having additional CRRs.

Can I share CRRs between an MVS CTG and a workstation CTG?

Yes, there is nothing physically preventing a CRR having EXCI communication from an MVS CTG and TCP/IP from a workstation CTG. However, I would not recommend it. In addition to the aforementioned failure problem, you also have to consider the location of the firewall and DMZ. I think it better not to share the CRRs in this fashion.

What about unit of work considerations?

I am not going to rehearse the arguments about wanting a secure and reliable set of updates in this article – as a CICS person you will certainly be familiar with them. Needless to say, you want all activity initiated from WebSphere to obey the usual transactional rules (all-or-nothing).

It is a characteristic of the configuration in Figure 5 that the unit of work (in CICS terminology) can encompass both activity within WebSphere and CICS. This means that recoverable resources accessed in both places are committed and rolled back in one lump as expected (and desired). There is no recoverable resource activity in the CRR, but the unit of work still encompasses this region.

What about security identities?

Security identities are available all the way through the pathway to CICS. WebSphere activity can be authorized under the userid and password supplied by the end user. Access to the CTG within the WebSphere J2EE Server can be secured, and this security identity can be used for access via the CTG. The security identity is checked within the CRR and again within the AOR used to execute the required CICS program.

CONCLUSION

When an MVS CICS Transaction Gateway is being used to provide access to CICS functions from a WebSphere Application, availability is provided using the CICS Transaction Gateway EXCI exit to avoid failures.

Employment of CRRs engenders the usual CICS load balancing and availability functions to provide maximum availability and best performance of requests.

COMING NEXT

After all this theoretical configuration stuff, the last in this series of articles will provide a practical example of workstation CICS Transaction Gateway usage by providing some Visual Basic code to link an Excel spreadsheet with CICS.

Robert Harris
CICS Technical Strategist
IBM Hursley (UK)

Displaying task activity in a CICS region under stress to support CICS TS 1.3 regions

The program MAPTCA, described in *Displaying task activity in a CICS region under stress*, published in *CICS Update*, Issue 182, January 2001and appearing in a recent *Mainframe Week* issue (http://www.mainframeweek.com/journals/show.php/0054/7), works only on a CICS V4.1 region. The essential logic of the program remains mostly unchanged for CICS TS 1.3 regions, but several offset changes are necessitated by minor control block modifications.

Another issue that has been brought to my attention by users of MAPTCA is that it relies on a rather less than satisfactory method of following the CICS control block chains by starting from the Kernel Domain Gate Table entry for the dispatcher as a hard coded value of X'6518'. This is based on the assumption that the Kernel Anchor Block is always at offset X'6000', which, indeed, is often correct, but is by no means guaranteed, even in a CICS V4.1 region. After checking several sites running CICS TS 1.3 under z/OS I have yet to find a region where this value has not changed to X '7000'.

Rather than just change to a different hard-coded starting point, I decided to implement a technique of picking up the Kernel Anchor Block address from the CICS region being analysed. This is easily accomplished by following the region's TCB chain until a pointer to the Authorized Function Control Block is found.

The changes required to achieve both of these goals are self-contained and can easily be inserted into the existing MAPTCA code to give users the ability to display task activity in CICS TS 1.3 regions.

The block of code INTO ACCESS MODE - R6 FOR CICS ADDRESS SPACE ends with the following statement:

```
LAM   R6,R6,=F'1'
```

After this statement, add the following block of code:

```
*-------------------------------------------------------------------
* FIND CICS KERNEL ANCHOR BLOCK
*-------------------------------------------------------------------
          L     R6,ASCBASXB          ASXB
          L     R6,X'Ø4'(,R6)        ASXBFTCB
PROCESS_TCB DS ØH
          LR    R8,R6                STORE TCB
          L     R6,X'DØ'(,R6)        TCBEXT2
          L     R6,X'14'(,R6)        SUBSYSTEM FACILITY CB (AFCB)
          LTR   R6,R6                AFCB POINTER?
          BNZ   PROCESS_AFCB         YES, USE IT
          LR    R6,R8                RESTORE TCB
          L     R6,X'74'(,R6)        TCBTCB
          B     PROCESS_TCB
PROCESS_AFCB DS ØH
          AH    R6,X'Ø6'(,R6)        VECTOR LIST LENGTH
          LA    R6,X'1Ø'(,R6)        AFCB PREFIX LENGTH
          L     R6,X'Ø4'(,R6)        AFCS
          L     R6,X'Ø8'(,R6)        DFHKEKCB
          L     R6,X'54Ø'(,R6)       DFHDSANCHOR
*-------------------------------------------------------------------
* END OF NEW CODE
*-------------------------------------------------------------------
```

Immediately following this, the lines of code between comments FOLLOW CICS STORAGE CHAINS and OUT OF ACCESS MODE must be deleted in their entirety and replaced with the following code:

```
*-------------------------------------------------------------------
* FOLLOW CICS STORAGE CHAINS
*-------------------------------------------------------------------
          L     R6,X'AC'(,R6)        DTA
PROCESS_DTA DS ØH
          LR    R5,R6                STORE DTA
          MVC   Ø(8,R7),X'ØC'(R6)    RESOURCE NAME
          MVC   8(8,R7),X'14'(R6)    RESOURCE TYPE
          MVC   16(1,R7),X'44'(R6)   TASK STATE
          L     R6,X'8Ø'(,R6)        XMTXN
          LTR   R6,R6                XMTXN PRESENT?
          BZ    NEXT_DTA
          MVC   17(4,R7),X'3C'(R6)   TASK NUMBER
          MVC   21(4,R7),X'48'(R6)   TRANSID
          L     R6,X'88'(,R6)        TCA
          LTR   R6,R6                TCA PRESENT?
          BZ    NEXT_DTA
PROCESS_TCA DS ØH
```

```
          LR    R8,R6                STORE TCA
          L     R6,X'1E4'(,R6)       TIE
PROCESS_TIE DS ØH
          MVC   25(18,R7),X'39'(R6)  LUWID LENGTH(1 BYTE) AND LUWID
          LR    R6,R8                RESTORE TCA
          ST    R6,43(,R7)           TCA ADDRESS
          L     R6,X'DC'(,R6)        CSA
PROCESS_CSA DS ØH
          MVC   CSACDTA,X'4C'(R6)    CURRENTLY DISPATCHED TCA
NEXT_DTA DS    ØH
          LR    R6,R5                RESTORE DTA
          LA    R7,47(,R7)
          L     R6,X'2C'(,R6)        NEXT DTA
          LTR   R6,R6                LAST DTA?
          BNZ   PROCESS_DTA
          ST    R7,TABEND
*-------------------------------------------------------------------
* OUT OF ACCESS MODE
*-------------------------------------------------------------------
```

Optionally, as the hard-coded constant is no longer used, the following two lines may be deleted from the MAPTCA source:

```
KEKCB     DS    ØF
          DC    XL4'ØØØØ6518'
```

As before, MAPTCA must be linked into an APF library with AC=1.

Patrick Mullen
Consultant (Canada)

CICS-XMITIP interface for sending e-mails without activating CICS TCP/IP environment

The objective was to monitor a critical (in my boss's opinion) banking CICS-MQSeries application by sending a warning message to signal anomalies.

Our environment is OS/390 2.10, CICS/TS 1.3 with no TCP/IP interface active, TCP/IP with SMTP working at least for sending e-mail, and XMITIP installed.

XMITIP is a very powerful (and free) SMTP mail generator for OS/390 and z/OS (batch and TSO) written by Lionel B Dyck (for information visit www.lbdsoftware.com). The XMITIP application level used is 4.52 but I think there are no problems with different levels.

The fastest solution is to use OS/390 SMTP and XMITIP to send warning e-mails without activating the CICS TCP/IP interface and without writing an e-mail generator for CICS.

To use XMITIP I had to write an interface between a CICS program and a batch application using the CICS JES interface to submit a batch job to MVS (spool files directed to the JES internal reader are treated as complete jobs and executed). Then I wrote an Assembler CICS program (XFHSMAIL), which submits a job containing a step executing a REXX program (CICSMAIL), which interfaces XMITIP.

A partitioned extended dataset is used to catalog sender addresses, recipient addresses, and e-mail subjects used by XFHSMAIL.

From this dataset, called SYSO.XE00.CICSMAIL in the example, XFHSMAIL reads sender (member XEDGFR01), recipients (member XEDGAD01), and subject (member XEDGSU01), while writing the text message into it (member XEDGMS01). Then CICSMAIL reads and uses these members as parameters in calling XMITIP.

You could submit a job to execute XMITIP directly from CICS but it's better to use a REXX program (CICSMAIL) to interface with it, thus simplifying the job built by XFHSMAIL. Changes to XMITIP parameters and/or library names require modification only to CICSMAIL without the need to recompile XFHSMAIL.

You can create a set of members, depending on the application you want to control, and modify the program to link XFHSMAIL, passing into the COMMAREA the suitable four member names and 80-character text message signalling the problem that occurred. XFHSMAIL creates a job with two steps:

1 IEBUPDTE writes the text message in a member of a partitioned dataset.

2 IKJEFT1B executes CICSMAIL.

CICSMAIL reads those four members, then calls XMITIP, which sends the e-mail using SMTP.

XFHSMAIL uses a TD queue called XSML (to be defined as CICS CSSL, for the RDO definition) to trace its activity. If you don't want to define this queue, XFHSMAIL will write its messages on CICS MSGUSR.

The only system definition required to enable the CICS JES interface is SPOOL=YES within the SIT. No additional DCT or CICS start-up procedure JCL changes are required.

It's important to consider the security. The *CICS RACF Security Guide* says that any CICS user, whether signed on or not, is able to submit jobs that use the SURROGAT userid, provided the CICS userid has authority for SURROGAT. If your installation is using EXEC CICS SPOOLOPEN to submit jobs, you cannot control who can submit jobs (without writing an API global user exit program to screen the commands). CICS spool commands do no CICS resource or command checking.

You can use an EXEC CICS ASSIGN USERID command to find the userid of the user who triggered the application code. Application programmers can then provide code that edits a USER operand on the JOB card destined for the internal reader.

So your CICS region userid (not the DFHSIT CICS default userid) must have no RACF OPERATIONS attribute because a JOB submitted from that CICS with no USER coded on the JOB card will be assigned to CICS region userid so that the job will have full access authorization to all RACF-protected resources in the DATASET, DASDVOL, GDASDVOL, PSFMPL, TAPEVOL, VMBATCH, VMCMD, VMMDISK, VMNODE, and VMRDR classes, with some exceptions (see the *SecureWay Security Server* manuals).

XFHSMAIL uses that method, so before using XFHSMAIL you must:

- Define the SURROGAT resource for SUBMIT:

```
RDEFINE SURROGAT JOB_userid.SUBMIT OWNER(Your_owner_userid) UACC(NONE)
PERMIT JOB_userid.SUBMIT CLASS(SURROGAT) RESET
```

- Authorize your CICS region userid to the resource just created:

```
PERMIT JOB_userid.SUBMIT CLASS(SURROGAT) ID(CICS_region_userid)
ACCESS(READ)
```

- Authorize your job userid to access the datasets needed to execute CICSMAIL (and XMITIP):
  - ACCESS(UPDATE) on the dataset containing the e-mail message (in my example SYSO.XE00.CICSMAIL).
  - ACCESS(READ) on the dataset containing the CICSMAIL, XMITIP datasets and TCPIP datasets.

I recommend authorizing your job userid only on the resources really needed and with minimal authorization to prevent abuses.

XFHSMAIL SOURCE

```
*ASM XOPTS(SP)
         TITLE '** XFHSMAIL - SENDING A MAIL FROM CICS VIA XMITIP **'
*
DFHEISTG       DSECT ,
*
*****************************************************************
```

```
* Commarea definition.
*********************************************************************
COMMAREA       DSECT ,
CADDRMEM       DS    CL08              Address list member name
CFROMMEM       DS    CL08              Sender member name
CSUBJMEM       DS    CL08              Subject member name
CMSGMEM        DS    CL08              Message member name
CMESSAGE       DS    CL80              Message text
COMMLEN        EQU   *-COMMAREA
*
*********************************************************************
* Various other variables.
*********************************************************************
RESP           DS    F                 Response code from EXEC CICS
TOKEN          DS    CL08              Spool token
SYSID          DS    CL04              CICS sysid
USERID         DS    CL08              Userid
ABSTIME        DS    PL08              Absolute time
TDQUEUE        DS    CL04              TDQueue
ERROR_TXT      DS    CL46              Error message text
JOB_NAME_TEMP  DS    CL08
PROGRAM        DS    CL08
APPLID         DS    CL08
CICSJOBNAME    DS    CL08
TDQENABLE      DS    F
TDQOPEN        DS    F
*ABCODE         DS    CL04              Abend code
*
*********************************************************************
* Message declarations.
*********************************************************************
*MSG_ACTION     DS    F
*
MSG_BUFFER     DS    0CL256            Message buffer
MSG_BUFFER_R1  DS    0CL128            Message buffer: 1st row
FIRST_MSG_AUTO DS    0CL82             Message buffer: 1st row
MSG_DATE       DS    CL07              Date YYYYDDD
               DS    CL01              Blank
MSG_TIME       DS    CL08              Time HH:MM:SS
               DS    CL01              Blank
               DS    CL11              CICS Applid
               DS    CL01              :
               DS    CL01              Blank
MSG_CAPPLID    DS    CL08              CICS applid
               DS    CL01              Blank
               DS    CL12              CICS JobName
               DS    CL01              :
               DS    CL01              Blank
MSG_CJOBNAME   DS    CL08              CICS jobname
```

```
               DS    CLØ1              Blank
               DS    CLØ1              -
               DS    CLØ1              Blank
               DS    CLØ7              Program
               DS    CLØ1              Blank
MSG_PROGRAM    DS    CLØ8              Program name
               DS    CLØ1              :
               DS    CLØ1              Blank
MSG_TEXT       DS    CL46              Message text
*
               ORG   MSG_TEXT
OK_MSG_AUTO    DS    ØCL36
               DS    CLØ3              Job
               DS    CLØ1              Blank
MSG_JOBNAME    DS    CLØ8              Job name
               DS    CL24
*
               ORG   MSG_TEXT
ERROR_MSG_AUTO DS    ØCL36
               DS    CLØ9
ERROR_TEXT     DS    CL27
*
               DS    CL1Ø
*
MSG_BUFFER_R2  DS    ØCL128            Message buffer: 2dn row
               DS    CLØ7
MSG_MAILMSG    DS    CL8Ø
               DS    CL41
*
*************************************************************************
* Job DSECT.
*************************************************************************
JOB_DSECT      DSECT ,
JOB_JCL_AUTO   DS    ØCL16ØØ
JOB_HEAD       DS    ØCL16Ø
               DS    CLØ2               //
JOB_NAME       DS    ØCLØ8              Job name
               DS    CLØ2
JOB_NAME_SYSID DS    CLØ4
               DS    CLØ2
               DS    CLØ6
               DS    CL13               Account
               DS    CLØ8               ),CLASS=
JOB_CLASS      DS    CLØ1
               DS    CL1Ø               ,MSGCLASS=
JOB_MSGCLASS   DS    CLØ1
               DS    CL31               ,
               DS    CLØ2               //
               DS    CLØ4               4 BLANKS
```

```
                DS    CLØ7               REGION=
JOB_REGION      DS    CL05               Region value
                DS    CL06               ,USER=
JOB_USER        DS    CLØ8               User
                DS    CL48
JOB_IEBUPDTE    DS    ØCL4ØØ
                DS    CL4ØØ
JOB_IEBUPDTE_C  DS    ØCL8Ø
                DS    CL12               ./ ADD NAME=
JOB_IEBUPDTE_M  DS    CLØ8               Message member name
                DS    CL6Ø               ,LIST=ALL
JOB_IEBUPDTE_X  DS    CL8Ø               Message
                DS    CL8Ø               ><
                DS    CL8Ø               //*
JOB_CICSMAIL    DS    ØCL56Ø
                DS    CL56Ø
JOB_CICSMAIL_C  DS    ØCL8Ø
                DS    CL12                 %CICSMAIL
JOB_CICSMAIL_A  DS    CLØ8
                DS    CLØ1
JOB_CICSMAIL_F  DS    CLØ8
                DS    CLØ1
JOB_CICSMAIL_S  DS    CLØ8
                DS    CLØ1
JOB_CICSMAIL_M  DS    CLØ8
                DS    CL33
                DS    CL8Ø
JOB_END         EQU   *
*
         DFHREGS ,
*
*        R2                         Points to COMMAREA
*        R7                         Points to JOB_DSECT
*        R8                         Length of job row
*        R9                         Points to end of JOB_DSECT
*        RA                         Return address
*
*********************************************************************
* XFHSMAIL mainline code.
*********************************************************************
XFHSMAIL CSECT
XFHSMAIL AMODE 31
XFHSMAIL RMODE ANY
*
*********************************************************************
* Addressing commarea.
*********************************************************************
         OC    EIBCALEN,EIBCALEN    Is there a commarea?
         BZ    LBRETURN             No, just return
         L     R2,DFHEICAP          Get the commarea address
```

```
         USING COMMAREA,R2
*
*****************************************************************
* Obtain the program name and CICS applid for messages.
* In addition, obtain the startcode type.
*****************************************************************
LBASSIGN EQU   *
         MVC   MSG_BUFFER_R1,SPACES
         MVC   MSG_BUFFER_R2,SPACES
         MVC   FIRST_MSG_AUTO,FIRST_MSG
         MVC   MSG_BUFFER_R2,SECOND_MSG
         MVC   MSG_MAILMSG,CMESSAGE
         EXEC  CICS ASSIGN                                        *
                PROGRAM(PROGRAM)                                  *
                APPLID(APPLID)                                    *
                SYSID(SYSID)                                      *
                USERID(USERID)                                    *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBE001                  Assign error
*
         MVC   MSG_PROGRAM,PROGRAM
         MVC   MSG_CAPPLID,APPLID
*
*****************************************************************
* Get the CICS Jobname.
*****************************************************************
LBINQSYS EQU   *
         EXEC  CICS INQUIRE SYSTEM                                *
                JOBNAME(CICSJOBNAME)                              *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBE002                  Inquire system error
*
         MVC   MSG_CJOBNAME,CICSJOBNAME
*
*****************************************************************
* Open the spool.
*****************************************************************
         EXEC  CICS SPOOLOPEN                                     *
                OUTPUT                                            *
                USERID('INTRDR')                                  *
                NODE('LOCAL')                                     *
                TOKEN(TOKEN)                                      *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBE003                  Spool open error
*
*****************************************************************
* Addressing JOB_DSECT.
```

```
*********************************************************************
         LA    R7,JOB_JCL
         USING JOB_DSECT,R7
         LA    R8,8Ø                   JCL row length
         LA    R9,JOB_END              End of JCL
         SR    R9,R8
*
         MVC   JOB_NAME_SYSID,SYSID
         MVC   JOB_NAME_TEMP,JOB_NAME
         MVC   JOB_USER,USERID
         MVC   JOB_IEBUPDTE_M,CMSGMEM
         MVC   JOB_IEBUPDTE_X,CMESSAGE
         MVC   JOB_CICSMAIL_A,CADDRMEM
         MVC   JOB_CICSMAIL_F,CFROMMEM
         MVC   JOB_CICSMAIL_S,CSUBJMEM
         MVC   JOB_CICSMAIL_M,CMSGMEM
*
*********************************************************************
* Write on spool.
*********************************************************************
LBSPLWRT EXEC  CICS SPOOLWRITE                                        *
                TOKEN(TOKEN)                                          *
                FROM(Ø(R7))                                           *
                FLENGTH(=F'8Ø')                                       *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBEØØ4                  Spool write error
*
         BXLE  R7,R8,LBSPLWRT
*
*********************************************************************
* Close the spool. (Submit the job...)
*********************************************************************
         EXEC  CICS SPOOLCLOSE                                        *
                TOKEN(TOKEN)                                          *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBEØØ5                  Spool close error
*        B     LBNMSG
*
*********************************************************************
* Normal end of procedure. (Send a message on WTO...)
*********************************************************************
LBNMSG   EQU   *
         MVC   OK_MSG_AUTO,OK_MSG
         MVC   MSG_JOBNAME,JOB_NAME_TEMP
         LA    RA,LBRETURN             Load return address
         B     LBTIME
*
*********************************************************************
```

```
* End the program and return to CICS. (Some error occurs...)
*********************************************************************
LBRETURN EQU   *
         EXEC  CICS RETURN
*
*********************************************************************
* Abnormal end of procedure.
*********************************************************************
LBENDRØ  EQU   *
         LA    RA,LBRETURN             Load return address
*        B     LBTIME
*
*********************************************************************
* Set date and time.
*********************************************************************
LBTIME   EQU   *
         EXEC  CICS ASKTIME ABSTIME(ABSTIME)
         EXEC  CICS FORMATTIME                                       *
                ABSTIME(ABSTIME)                                     *
                YYYYDDD(MSG_DATE)                                    *
                TIME(MSG_TIME) TIMESEP                               *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBEØØ6                  FormatTime error
         MVC   TDQUEUE,XSML
*
*********************************************************************
* Get info about TDQueue.
*********************************************************************
LBINQTDQ EQU   *
         EXEC  CICS INQUIRE                                          *
                TDQUEUE(TDQUEUE)                                     *
                ENABLESTATUS(TDQENABLE)                              *
                OPENSTATUS(TDQOPEN)                                  *
                RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BE    LBITDQOK                Inquire TDQueue ok
         CLC   RESP,DFHRESP(QIDERR)
         BE    LBCHGTDQ                Queue not found
         B     LBEØØ7                  Inquire TDQueue error
*
*********************************************************************
* Verify XSML TDQueue status.
*********************************************************************
LBITDQOK EQU   *
         CLC   TDQENABLE,DFHVALUE(ENABLED)
         BNE   LBCHGTDQ                XSML not enabled
         CLC   TDQOPEN,DFHVALUE(OPEN)
         BNE   LBCHGTDQ                XSML not opened
         B     LBWRTTDQ                Write on XSML
```

```
*
*********************************************************************
* Set TDQueue to CSSL CICS default output queue (MSGUSR).
*********************************************************************
LBCHGTDQ EQU   *
         MVC   TDQUEUE,CSSL
*        B     LBWRTTDQ
*
*********************************************************************
* Procedure to issue a message to the console
*********************************************************************
LBWRTTDQ EQU   *
*        MVC   MSG_ACTION,DFHVALUE(EVENTUAL)
*        EXEC  CICS WRITE OPERATOR                                  *
*               TEXT(MSG_BUFFER) TEXTLENGTH(L'MSG_BUFFER)           *
*               ACTION(MSG_ACTION)                                  *
*               RESP(RESP)
         EXEC  CICS WRITEQ TD                                        *
               QUEUE(TDQUEUE) FROM(MSG_BUFFER_R1)                    *
               LENGTH(L'MSG_BUFFER_R1)                               *
               RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBE00B                  TDQueue write error
         EXEC  CICS WRITEQ TD                                        *
               QUEUE(TDQUEUE) FROM(MSG_BUFFER_R2)                    *
               LENGTH(L'MSG_BUFFER_R2)                               *
               RESP(RESP)
         CLC   RESP,DFHRESP(NORMAL)
         BNE   LBE00B                  TDQueue write error
         BR    RA
*
*********************************************************************
* Labels to branch to when a particular error occurs.
*********************************************************************
LBE001   EQU   *
         MVC   ERROR_TXT,=CL27'Assign.'
         B     LBEMSG
LBE002   EQU   *
         MVC   ERROR_TXT,=CL27'Inquire System.'
         B     LBEMSG
LBE003   EQU   *
         MVC   ERROR_TXT,=CL27'Spool Open.'
         B     LBEMSG
LBE004   EQU   *
         MVC   ERROR_TXT,=CL27'Spool Write.'
         B     LBEMSG
LBE005   EQU   *
         MVC   ERROR_TXT,=CL27'Spool Close.'
         B     LBEMSG
LBE006   EQU   *
```

```
         MVC   ERROR_TXT,=CL27'Format Time.'
         B     LBEMSG
LBEØØ7   EQU   *
         MVC   ERROR_TXT,=CL27'Inquire TDQueue.'
         B     LBEMSG
LBEØØ8   EQU   *
         MVC   ERROR_TXT,=CL27'WriteQ TD.'
         B     LBEMSG
*
*****************************************************************
* Abend.
*****************************************************************
*LBABEND  EQU   *
*         EXEC  CICS ABEND ABCODE(ABCODE)
*         B     LBRETURN
*
*****************************************************************
* Write an error message.
*****************************************************************
LBEMSG   EQU   *
         MVC   ERROR_MSG_AUTO,ERROR_MSG
         MVC   ERROR_TEXT,ERROR_TXT
         B     LBENDRØ
*
*****************************************************************
* Constants.
*****************************************************************
SPACES      DC   CL128' '
XSML        DC   CLØ4'XSML'
CSSL        DC   CLØ4'CSSL'
*
*****************************************************************
* Messages.
*****************************************************************
FIRST_MSG   DC    ØCL82' '
            DC    CL4Ø'YYYYDDD HH:MM:SS CICS Applid: C_APPLID C'
            DC    CL4Ø'ICS JobName: CJOBNAME - Program PROGNAME'
            DC    CLØ2': '
*
ERROR_MSG   DC    ØCL36' '
            DC    CLØ9'Error on '
            DC    CL27' '
*
OK_MSG      DC    ØCL36' '
            DC    CLØ4'Job '
            DC    CLØ8'JOB_NAME'
            DC    CL24' submitted successfully.'
*
SECOND_MSG  DC    ØCL128' '
            DC    CLØ7'  MSG: '
```

```
            DC    CL121' '
*
JOB_JCL     DC CL8Ø'//XEXXXXSM JOB (XEØØØØPØØØØØØ),CLASS=A,MSGCLASS=O,'
            DC CL8Ø'//    REGION=ØØØØK,USER=UUUUUUUU                    '
            DC CL8Ø'//STEPØØ1Ø  EXEC PGM=IEBUPDTE,PARM=NEW'
            DC CL8Ø'//SYSUT1    DD DUMMY'
            DC ØCL8Ø' ,
            DC CL35'//SYSUT2    DD DISP=(OLD,PASS),DSN='
            DC CL45'SYSO.XEØØ.CICSMAIL'
            DC CL8Ø'//SYSPRINT  DD SYSOUT=*'
            DC ØCL8Ø' '
            DC CL24'//SYSIN     DD DATA,DLM='
            DC X'7D'
            DC CLØ2'><'
            DC X'7D'
            DC CL52' '
            DC CL8Ø'./ ADD NAME=MMMMMMMM,LIST=ALL'
            DC CL8Ø' '
            DC CL8Ø'><'
            DC CL8Ø'//*'
            DC CL8Ø'//CICSMAIL  EXEC PGM=IKJEFT1B'
            DC CL8Ø'//SYSEXEC   DD DISP=SHR,DSN=SYSO.CED.ISPCLIB'
            DC CL8Ø'//XEMAILDS  DD DISP=OLD,DSN=SYSO.XEØØ.CICSMAIL'
            DC CL8Ø'//SYSTSPRT  DD SYSOUT=*'
            DC CL8Ø'//SYSTSIN   DD *'
            DC CL8Ø'  PROFILE NOPREFIX'
            DC CL8Ø'  EXECUTIL SEARCHDD(YES)'
            DC CL8Ø'  %CICSMAIL AAAAAAAA FFFFFFFF SSSSSSSS MMMMMMMM'
            DC CL8Ø'//*'
*
         LTORG
*
*********************************************************************
* End of XFHSMAIL
*********************************************************************
         END   XFHSMAIL
```

CICSMAIL SOURCE

```
/******************************* REXX ***************************************/
/* Program    : CICSMAIL                                                    */
/* Author     : Gianluca Bonzano                                            */
/* Company    : Cedacri [Ovest] S.p.A.                                      */
/* Description: XMITIP interface for CICS mail                              */
/* Caller     : CICS program XFHSMAIL                                       */
/* Input      :                                                             */
/* Output     :                                                             */
/* JCL        : //CICSMAIL  EXEC PGM=IKJEFT1B                              */
/*              //SYSEXEC   DD DISP=SHR,DSN=YOUR.ISPCLIB                   */
```

```
/*              //XEMAILDS  DD DISP=OLD,DSN=SYSO.XEØØ.CICSMAIL        */
/*              //SYSTSPRT  DD SYSOUT=*                               */
/*              //SYSTSIN   DD *                                      */
/*                PROFILE NOPREFIX                                    */
/*                EXECUTIL SEARCHDD(YES)                              */
/*                %CICSMAIL AddrMember FromMember SubjMember MsgMember*/
/******************************** REXX *********************************/

ADDRESS TSO

ARG AddrMember FromMember SubjMember MsgMember

XEMAILDS = "SYSO.XEØØ.CICSMAIL"
XMITIPDS = «SYSO.XMITIP.EXEC»
TCPIPDS  = "SYSO.TCPO1.PARMS"

SubjectDS = XEMAILDS"("SubjMember")"
"Allocate DA('"SubjectDS"') F(SUBJDD) SHR REU"
"Execio 1 diskr SUBJDD (Finis stem insubj."
"free fi(SUBJDD)"

FromDS = XEMAILDS"("FromMember")"
"Allocate DA('"FromDS"') F(FROMDD) SHR REU"
"Execio * diskr FROMDD (Finis stem infrom."
"free fi(FROMDD)"

MessageDS = XEMAILDS"("MsgMember")"
"Allocate DA('"MessageDS"') F(MSGDD) SHR REU"
"Execio * diskr MSGDD (Finis stem inmsg."
"free fi(MSGDD)"

AddressDS = XEMAILDS"("AddrMember")"
"Allocate DA('"AddressDS"') F(ADDRDD) SHR REU"

TCPData = TCPIPDS"(TCPDATA)"
"Allocate DA('"TCPData"') F(SYSTCPD) SHR REU"

Call ExecXmitip

Exit RC

ExecXmitip:
"altlib activate application(exec) dataset('"XMITIPDS"')"
do i = 1 to inmsg.Ø
  queue inmsg.i
end
"%XMITIP * ADDRESSFILEDD ADDRDD FROM "strip(infrom.1,'t') ,
 "SUBJECT '"strip(insubj.1,'t')"' MSGQ"
Return RC
```

HOW TO LINK XFHSMAIL IN A COBOL PROGRAM

```
       WORKING-STORAGE SECTION.
     ...
       Ø1  XFHSMAIL          PIC X(8) VALUE 'XFHSMAIL'.
       Ø1  WS-MAIL.
           Ø5  WS-MAIL-DEST      PIC X(Ø8) VALUE SPACES.
           Ø5  WS-MAIL-FROM      PIC X(Ø8) VALUE SPACES.
           Ø5  WS-MAIL-SUBJ      PIC X(Ø8) VALUE SPACES.
           Ø5  WS-MAIL-MSG       PIC X(Ø8) VALUE SPACES.
           Ø5  WS-MAIL-DESCR     PIC X(8Ø) VALUE SPACES.
     ...

       PROCEDURE DIVISION.
     ...
           PERFORM AØ3Ø-MAIL    THRU AØ3Ø-EXIT
     ...

       AØ3Ø-MAIL.
           MOVE 'XEDGADØ1'      TO WS-MAIL-DEST
           MOVE 'XEDGFRØ1'      TO WS-MAIL-FROM
           MOVE 'XEDGSUØ1'      TO WS-MAIL-SUBJ
           MOVE 'XEDGMSØ1'      TO WS-MAIL-MSG
           MOVE WS-MSG-CONSOLE  TO WS-MAIL-DESCR
           EXEC CICS LINK PROGRAM(XFHSMAIL)
                          COMMAREA(WS-MAIL)
                          LENGTH(LENGTH OF WS-MAIL)
           END-EXEC.
       AØ3Ø-EXIT.   EXIT.
```

EXAMPLE JOB SUBMITTED BY XFHSMAIL

```
//XECEHBSM JOB (XEØØØØPØØØØØØ),CLASS=A,MSGCLASS=9,
//    REGION=ØØØØK,USER=CICSTD
//STEPØØ1Ø  EXEC PGM=IEBUPDTE,PARM=NEW
//SYSUT1    DD DUMMY
//SYSUT2    DD DISP=(OLD,PASS),DSN=SYSO.XEØØ.CICSMAIL
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DATA,DLM='><'
./ ADD NAME=XEDGMSØ1,LIST=ALL
XM73TEST ERR CONV XML/COBOL: 4 WBØØBØØ3: Nome campo del TAG non presente
><
//*
//CICSMAIL  EXEC PGM=IKJEFT1B
//SYSEXEC   DD DISP=SHR,DSN=SYSO.CED.ISPCLIB
//XEMAILDS  DD DISP=OLD,DSN=SYSO.XEØØ.CICSMAIL
//SYSTSPRT  DD SYSOUT=*
//SYSTSIN   DD *
```

```
  PROFILE NOPREFIX
  EXECUTIL SEARCHDD(YES)
  %CICSMAIL XEDGAD01 XEDGFR01 XEDGSU01 XEDGMS01
//*
```

EXAMPLE OF SENDER, RECIPIENT(S), SUBJECT, AND TEXT FOR AN E-MAIL

```
VIEW       SYSO.XE00.CICSMAIL(XEDGFR01) - 01.00      Columns 00001 00072
Command ===>                                             Scroll ===> CSR
****** ************************** Top of Data ***************************
000001 SCROOGEMCDUCK@DISNEY.COM
****** ************************* Bottom of Data ************************

VIEW       SYSO.XE00.CICSMAIL(XEDGAD01) - 01.03      Columns 00001 00072
Command ===>                                             Scroll ===> CSR
****** ************************* Top of Data ****************************
000001 TO MICKEY.MOUSE@DISNEY.COM
000002 TO DONALD.DUCK@DISNEY.COM
****** ************************** Bottom of Data ************************

VIEW       SYSO.XE00.CICSMAIL(XEDGSU01) - 01.01      Columns 00001 00072
Command ===>                                             Scroll ===> CSR
****** ************************** Top of Data ***************************
000001 DRIVER MQ: NEW MESSAGE ON CONSOLE
****** ************************** Bottom of Data ************************

VIEW       SYSO.XE00.CICSMAIL(XEDGMS01) - 01.00      Columns 00001 00072
Command ===>                                             Scroll ===> CSR
****** ************************* Top of Data ****************************
000001 XMQGPROD WARNING: DRIVER-MONITOR XM88 NOT ACTIVE
****** ************************ Bottom of Data ***************************
```

RDO DEFINITION FOR CICS XSML TD QUEUE FOR XFHSMAIL MESSAGES

```
DEFINE TDQUEUE(XSML) GROUP(DCTHB)
DESCRIPTION(USED FOR XFHSMAIL MESSAGES)
       TYPE(EXTRA) DATABUFFERS(1) DDNAME(XSMLOUT) ERROROPTION(IGNORE)
       OPENTIME(INITIAL) TYPEFILE(OUTPUT) RECORDSIZE(132)
       BLOCKSIZE(136) RECORDFORMAT(VARIABLE) BLOCKFORMAT(UNBLOCKED)
       DISPOSITION(SHR)
```

Gianluca Bonzano
Cedacri Ovest SpA(Italy)

A simple interface to CICS load module Scanner Utility – part 2

This month we conclude the code for the tool that helps to search for all affinities in a customer's applications. It can be used to find which CICS commands (EXEC CICS) are present in the programs and which can cause the transaction affinity.

Panel CXSUP00H

```
)ATTR
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(§§) WINDOW(62 22) SMSG(VIDEO)
+$VIDEO
\ ***********************#H E L P\************************** +
+                                                            +
+    This utility can scan load libraries for the CICS       +
+    commands in load modules, and identify which modules    +
+    contain certain API commands.                           +
+                                                            +
+    The available functions are:                            +
+                                                            +
+ -%Generate + of Summary and/or Detail Scan report on the   +
+              specific load library;                        +
+              It executes a CICS utility in order to        +
+              produce an output report.                     +
+                                                            +
+ -%Display  + of the reports generated with generate        +
+              option;                                       +
+                                                            +
+                                                            +
+ PF3=Exit from Help                                         +
+                                                            +
)INIT
)PROC
)END
```

Panel CXSUP001

```
)ATTR
 '   COLOR(turq) TYPE(TEXT)
 ?   COLOR(white) TYPE(TEXT)
```

```
 <   COLOR(blue) TYPE(TEXT) intens(high)
 # TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(yellow)
 \ type(TEXT) intens(HIGH) COLOR(green) hilite(reverse)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
)BODY EXPAND(//) WINDOW(68 18) SMSG(VIDEO)
+$VIDEO
+
<                                    User.....: &ZUSER
'  * * CICS SCANNER UTILITY * *     <Date.....: &ZDAY &ZMONTH &ZYEAR
?    Generate Summary Report        <Time.....: &ZTIME
<                                    Appl.....: Cics SCANLoad
<
<
<
+
<   \ Load Library  + #csusrin                                        +
+
+
+
+
+
+ $msgØ1                                                              +
+ PF1=Help  PF3=Exit  ENTER=Continue
)INIT
 .CURSOR = csusrin
 &ZCMD=' '
  VGET (csusrin) PROFILE
  .HELP = CXSUPØ1H
  &ZHTOP = CXSUPØ1H
  &ZHINDEX = CXSUPØ1H
)PROC
IF (&csusrin = ' ') .MSG = 'CXSUMØØ2'
VPUT (csusrin) PROFILE
)END
```

Panel CXSUP01H

```
)ATTR
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(@@) WINDOW(62 18) SMSG(VIDEO)
+$VIDEO
\ ***********************#H E L P\************************** +
+                                                            +
+  Function of%Generate+Summary/Detail Scanner CICS load     +
+  module report.                                            +
+                                                            +
+                                                            +
```

```
+  Specify the following field:                                  +
+                                                                +
+ -%Load Library + the name of CICS load library to scan.       +
+                                                                +
+ -%Module List  + the dataset name with module list to         +
+                  scan from load library. (detail run only)    +
+                                                                +
+                                                                +
+                                                                +
+ PF3=Exit from Help                                            +
+                                                                +
)INIT
)PROC
)END
```

Panel CXSUP002

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
 [ TYPE(OUTPUT) INTENS(high) color(green) hilite(reverse)
 # TYPE(OUTPUT) INTENS(high) color(red) hilite(blink)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) color(yellow)
 | TYPE(text) INTENS(HIGH) color(green)
)BODY
+
+ %Command ===>_ZCMD                                                +
+
%                 CICS Scanner Load Utility - Summary Report  +      +
+                                                                    +
+    #MSGØ1                                                         +
+ ==========I============I============I==========I==================+
+   Module  I  Module    I  Affinity  I MVS POSTs I Comment          +
+    Name   I Language   I Statements I           I                 +
+ ==========I============I============I==========I==================+
)MODEL
 [module    +@lang       @affin       +@posts     $comm
+
)INIT
  .CURSOR = ZCMD
 &ZCMD=' '
  .HELP = CXSUPØ2H
  &ZHTOP = CXSUPØ2H
  &ZHINDEX = CXSUPØ2H
)PROC
)END
```

Panel CXSUP02H

```
)ATTR
```

```
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(@@) WINDOW(62 16) SMSG(VIDEO)
+$VIDEO
\ **********************#H E L P\************************** +
+                                                          +
+    Function of%display+Summary/Detail Scanner report.    +
+                                                          +
+                                                          +
+    It reads CICS utility output obtained with the Generate +
+    function and produces the main information and the    +
+    statistics totals.                                    +
+                                                          +
+                                                          +
+                                                          +
+                                                          +
+                                                          +
+ PF3=Exit from Help                                       +
+                                                          +
)INIT
)PROC
)END
```

Panel CXSUP003

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
 [ TYPE(OUTPUT) INTENS(high) color(green) hilite(reverse)
 # TYPE(OUTPUT) INTENS(high) color(red) hilite(blink)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) color(yellow)
 | TYPE(text) INTENS(HIGH) color(green)
)BODY
+
+ %Command ===>_ZCMD                                              +
+
%        CICS Scanner Load Utility - Summary Report Totals  +     +
+                                                                 +
+    #MSGØ1                                                       +
+                                                                 +
)MODEL
 $ltot
+
)INIT
  .CURSOR = ZCMD
 &ZCMD=' '
  .HELP = CXSUPØ2H
  &ZHTOP = CXSUPØ2H
  &ZHINDEX = CXSUPØ2H
```

```
)PROC
)END
```

Panel CXSUP031

```
)ATTR
 '   COLOR(turq) TYPE(TEXT)
 ?   COLOR(white) TYPE(TEXT)
 <   COLOR(blue) TYPE(TEXT) intens(high)
 # TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(yellow)
 \ type(TEXT) intens(HIGH) COLOR(green) hilite(reverse)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
)BODY EXPAND(//) WINDOW(68 18) SMSG(VIDEO)
+$VIDEO
+
<                                    User.....: &ZUSER
'  * * CICS SCANNER UTILITY * *     <Date.....: &ZDAY &ZMONTH &ZYEAR
?     Generate Detail Report        <Time.....: &ZTIME
<                                    Appl.....: Cics SCANLoad
<
<
<
<   \ Load Library  + #csudrin                                      +
+
+
<   \ Module List   + #csudrdt                                      +
+
+
+
+ $msgØ1                                                            +
+ PF1=Help  PF3=Exit  ENTER=Continue
)INIT
 .CURSOR = csudrin
 &ZCMD=' '
  VGET (csudrin,csudrdt) PROFILE
  .HELP = CXSUPØ1H
  &ZHTOP = CXSUPØ1H
  &ZHINDEX = CXSUPØ1H
)PROC
IF (&csudrin = ' ' AND &csudrdt = ' ') .MSG = 'CXSUMØØ3'
IF (&csudrin = ' ' AND &csudrdt ¬= ' ') .MSG = 'CXSUMØØ2'
IF (&csudrin ¬= ' ' AND &csudrdt = ' ') .MSG = 'CXSUMØØ4'
VPUT (csudrin,csudrdt) PROFILE
)END
```

Panel CXSUP042

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
```

```
 [ TYPE(text) INTENS(high) color(green) hilite(reverse)
 # TYPE(OUTPUT) INTENS(high) color(red) hilite(blink)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) color(yellow)
 | TYPE(text) INTENS(HIGH) color(green)
)BODY
+
+ %Command ===>_ZCMD                                                  +
+
%             CICS Scanner Load Utility - Detail Report  +            +
+                                                                      +
+    #MSGØ1                                                           +
[ ================================================================   +
)MODEL
 @row
+
)INIT
  .CURSOR = ZCMD
 &ZCMD=' '
  .HELP = CXSUPØ2H
  &ZHTOP = CXSUPØ2H
  &ZHINDEX = CXSUPØ2H
)PROC
)END
```

Panel CXSUP043

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
 [ TYPE(OUTPUT) INTENS(high) color(green) hilite(reverse)
 # TYPE(OUTPUT) INTENS(high) color(red) hilite(blink)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) color(green)
 | TYPE(text) INTENS(HIGH) color(green)
)BODY
+
+ %Command ===>_ZCMD                                                  +
+
%         CICS Scanner Load Utility - Detail Report Totals  +         +
+                                                                      +
+    #MSGØ1                                                           +
+                                                                      +
)MODEL
 $rtot
+
)INIT
  .CURSOR = ZCMD
 &ZCMD=' '
  .HELP = CXSUPØ2H
  &ZHTOP = CXSUPØ2H
  &ZHINDEX = CXSUPØ2H
```

```
)PROC
)END
```

ISPF MESSAGES

CXSUM00

```
CXSUM000 'Select function'                                    .ALARM=YES
' To select one at least function.'
CXSUM001 'Wrong Selection'                                    .ALARM=YES
' To specify one single function for time.'
CXSUM002 'Cics Scan Generate'                                 .ALARM=YES
' To specify INPUT LOAD library.'
CXSUM003 'Cics Scan Generate'                                 .ALARM=YES
' To specify INPUT LOAD library and the list of the modules.'
CXSUM004 'Cics Scan Generate'                                 .ALARM=YES
' To specify the list of the modules for the detail Scan.'
```

SAMPLE DISPLAYS

Figure 1 shows the select function screen. Figure 2 gives an example list of modules and affinities. Figure 3 shows a summary

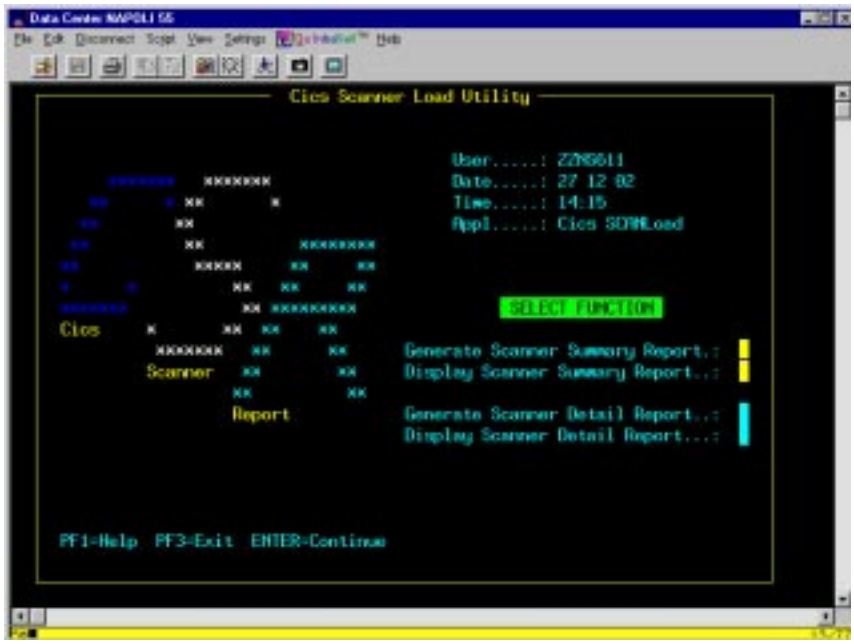

Figure 1: Select function screen

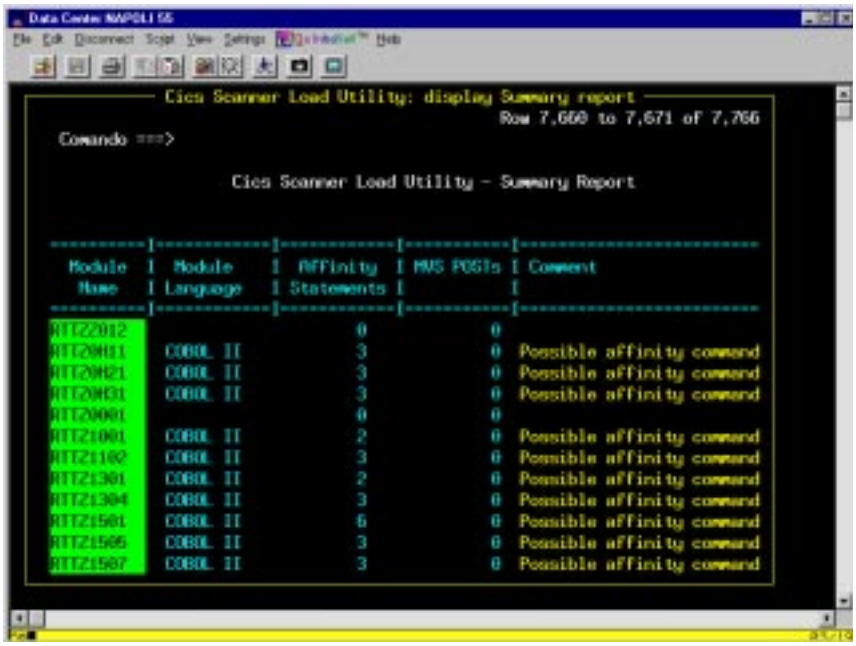

Figure 2: Example list of modules and affinities

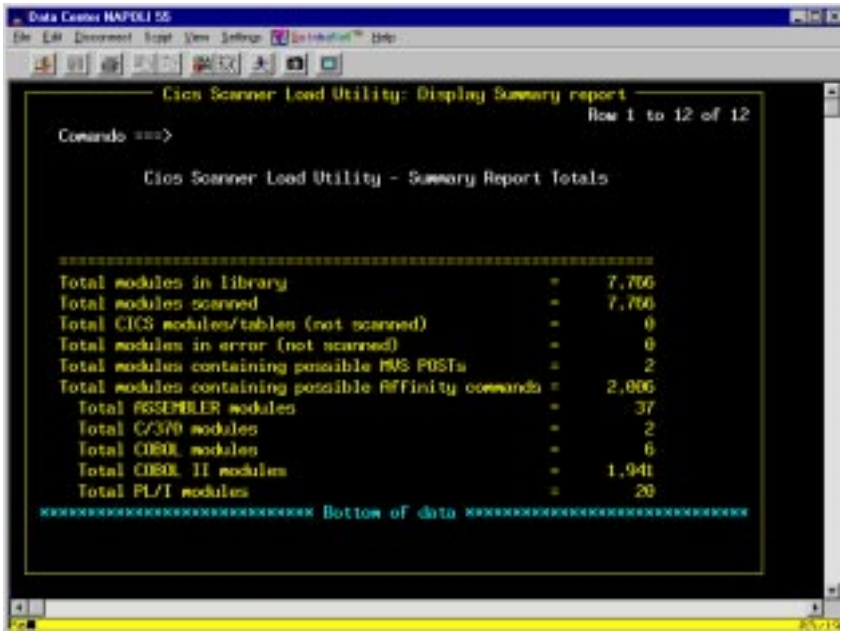

Figure 3: A summary report

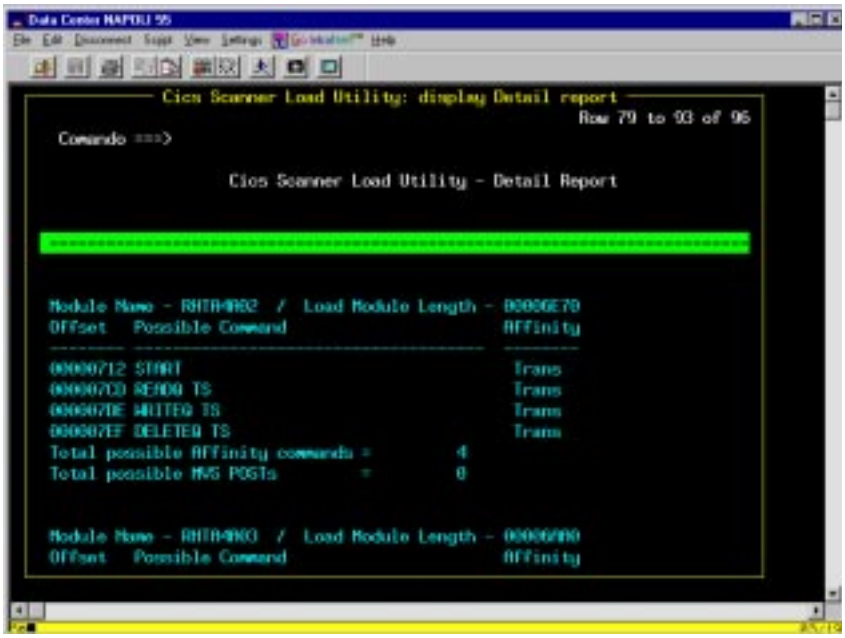

Figure 4: Details of modules and affinities

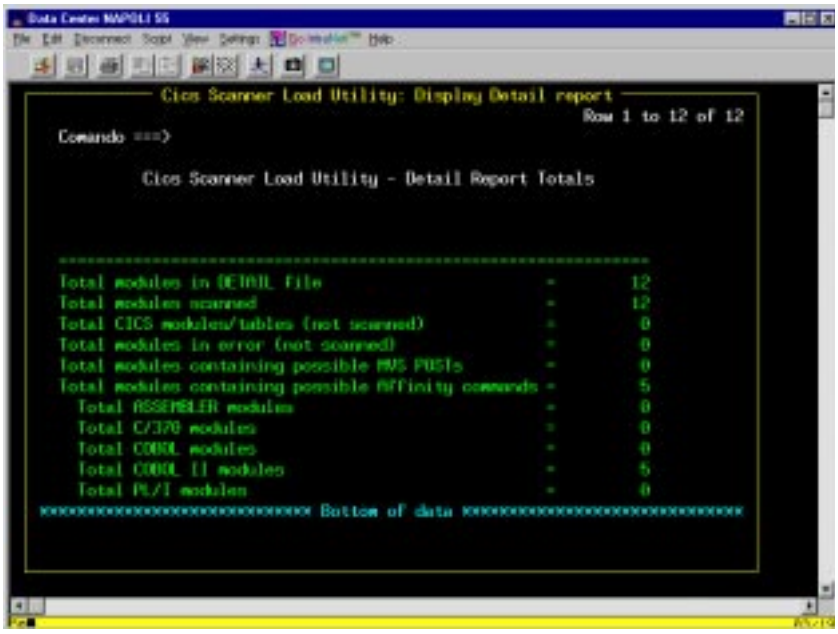

Figure 5: Details of a totals report.

report. Figure 4 shows details of modules and affinities. Figure 5 shows details of a totals report.

Espedito Morvillo
Systems Programmer (Italy)

CICS questions and answers

Q During the execution of a COBOL program running in a CICS environment I have to know how the SIT parameter, DBCTLCON, of the environment is set (or was set in the start-up job). Is this possible with an EXEC CICS command? What other possibilities do I have to find it out?

A The following will give you what you want:

```
EXEC CICS INQUIRE
  EXITPROGRAM("DFHDBAT")
  ENTRYNAME("DBCTL")
  CONNECTST(cvda)
```

The cvda values are: CONNECTED – DBCTL connection is active and DL/I calls can be issued; NOTCONNECTED – DBCTL connection is not currently active and DL/I calls cannot be issued; NOTAPPLIC – DBCTL connection is not set up in this CICS region and DL/I calls cannot be issued.

Q We are about to lose a product that shows scheduled events (ICEs). Is there a CICS way to get the same information?

A There is no equivalent display with CEMT in CICS TS 1.3, but there is a new SPI command – EXEC CICS INQUIRE REQID – that could be used in a program to build a similar display of ICEs. If you don't want a program then CECI can be used to browse through the list – using START, NEXT, END on a CECI INQ REQID command.

If you have any CICS-related questions, please send them in and we will do our best to find answers. Alternatively, e-mail them directly to cicsq@xephon.net.

Compuware has announced availability of Version 2.2 of its OptimalJ application development environment, which supports J2EE standards and implements the OMG's Model Driven Architecture.

Enhancements target simplifying the integration of a wide range of technologies such as CICS COBOL applications, CORBA, Java, and Web Services.

Web services can be created by elevating a generated Session Bean to a Web service. The product will generate a Web Service Definition Language (WSDL) file that other applications can use to invoke the new Web service.

For existing Web services, it generates the Session Bean code needed to invoke the foreign Web service, based on the imported WSDL file.

It also supports Borland's JBuilder integrated development environment in addition to the NetBeans IDE, plus all leading application servers, databases, and modelling tools.

Version 2.2 provides a pluggable pattern architecture, which apparently means that upgrading an application from EJB 1.1 to EJB 2.0 requires only the installation of the EJB 2.0 pattern. This version is also designed to let architects implement their own standards and guidelines.

For further information contact:
Compuware, 31440 Northwestern Highway, Farmington Hills, MI 48334-2564, USA.
Tel: (248) 737 7300.
URL: http://www.compuware.com/products/optimalj/.

* * *

Information Builders division iWay has announced its brokerless integration software as an alternative to broker-based EAI techniques and tools. It uses XML, Web services, and intelligent adapters, which provide a universal translator between applications that don't require the infrastructure usually included in integration broker suites.

This approach, says the vendor, allows any IT resource to be accessed through XML documents, which means its XML Transformation Engine can simply map from one XML document to another instead of writing custom transformation code on non-XML-based APIs.

It also means legacy databases and ERP applications interact with standardized XML documents that can be incorporated into any application or business process.

The intelligent adapters provide immediate technical connectivity to virtually any information system, including packaged applications such as SAP and Siebel, legacy data such as IMS and VSAM, and transaction systems such as CICS and IMS/TM.

The adapters also validate and transform B2B documents such as EDI and XML, including specific industry formats such as HIPAA, HL7, SWIFT, and FIX, into XML documents.

For further information contact:
iWay, Two Penn Plaza, New York, NY 10121-2898, USA.
Tel: (212) 330 1700.
URL: http://www.iwaysoftware.com.

* * *