CICS update

1 .

Special edition

In this issue

A tc PUBLICATION

CICS Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs $270.00 in the USA and Canada; £175.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) 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.

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

EXEC interface trace tips

CICS application program debugging can be made easier by analysis of the trace entries recorded when CICS is executing commands issued by the program under investigation. In particular, the EXEC CICS trace entries associated with the commands themselves are a good source of information about the program environment being studied.

EXEC CICS events are shown by EIP ENTRY and EIP EXIT trace points. By default, these are standard level 1 trace points. They denote the places where application programs issue EXEC CICS commands to invoke a CICS API or SPI function. The EIP ENTRY trace point is issued on entry to CICS from an application; the EIP EXIT trace point is issued just prior to the flow of control returning to the application from CICS. As such, they delimit the times between CICS and the application being in control.

The EIP ENTRY and EXIT trace points are issued from DFHEIP, the CICS EXEC Interface Program. Their trace point ID is AP 00E1. They contain the basic amount of data needed for analysis of a program's flow of control when invoking CICS commands. If EIP tracing is extended (by setting the EI component to 1-2 under the CETR trace components panel), an additional pair of DFHEIP trace entries are recorded by CICS. These standard level 2 trace point IDs are AP E160 and AP E161 (on entry to and exit from DFHEIP, respectively). These level 2 EI trace entries give far more detail about the command being executed by CICS, such as formatting the parameters, displaying their storage addresses within the CICS address space, showing the EIB, etc. While not expected to be used during production, they are a great aid in application development and debugging, and can be considered the trace equivalent of stepping through the application logic via CEDF.

This article describes some useful debugging tips and techniques that may be applied when working with EIP trace entries.

THE RETURN ADDRESS FIELD

EIP trace entries contain a return address value in the RET field. This is shown only when the trace entries are formatted using the SHORT or FULL trace format option. (Note: the RET value is not displayed if the ABBREVIATED (ABBREV) formatting option is used.)

For example, the following (edited) trace example shows an EXEC CICS ASKTIME command with a RET address of 0C907770:

```
00337 QR    AP 00E1 EIP   ENTRY ASKTIME-ABSTIME
     RET-0C907770 10:16:27.2234375941     =004281=
```

The example also shows that the trace entry was issued by task number 00337, running under the QR TCB at 10:16:27, and was trace entry number 004281 in this run of CICS tracing.

The RET address is the location within memory in the CICS address space where the flow of execution control will return after CICS has processed the EXEC CICS command. In other words, it is the location of the next instruction to be executed within the application program load module. If the entry point of the executing program can be determined, the offset of the EXEC CICS command within the program may be calculated. The entry point can be found in a number of ways. If a CICS transaction dump is available, the entry point will be formatted by the dump formatter. (Similarly, a CICS system dump will show the entry point via the LD VERBX option.) An example from a transaction dump is shown below:

```
TRANSACTION DUMP MODULE INDEX:
LOAD PT.      NAME      ENTRY PT     LENGTH

0C902730    TESTPGM1    0C902750    00001F80

0C907310    ANDYPGM     0C907330    00001D10

0C909020    TESTPGM2    0C909040    00002C98
```

More straightforwardly, preceding trace entries may contain the LD 0002 trace entry showing the successful call to acquire the program in memory, and this will reveal the program’s load and entry points. This last technique is viable, provided the loading of the program was recent enough for its CICS trace entries to remain within the internal trace table, and so not have been lost because of new entries wrapping around the table. (Such wrapping is less likely when dealing with the auxiliary trace datasets.)

As an example, the program ANDYPGM, having issued the EXEC CICS ASKTIME command, generated the following CICS Loader exit trace data:

```
LD ØØØ2 LDLD EXIT - FUNCTION(ACQUIRE_PROGRAM) RESPONSE(OK)

                        ENTRY_POINT(ØC9Ø733Ø) LOAD_POINT(ØC9Ø731Ø)
```

Subtraction of the RET address from the program's entry point shows that the EXEC CICS ASKTIME command was located at offset X'440' in the program ANDYPGM.

Note that a program's load point normally addresses the CICS command level stub. This is link-edited to the program and used as a route for the flow of execution from the program into CICS, when executing EXEC CICS commands within the program. Conversely, the entry point generally points to the start of the application itself. This can be confirmed from the link-edit mapping generated when the program load module was created.

Use of the RET address can be very helpful when tying together the dynamic environment of EXEC CICS commands in a trace and the static contents of a CICS application program listing. For example, suppose an EXEC CICS RETURN command in a COBOL application was being investigated. Using the above technique, assume that in this example an offset of X'51E' was calculated for the command within the program. By locating this offset in the program listing, it can be seen that the offset X'51E' lies immediately after a BALR 14,15 command, and that it is part of the assembly statements generated from a CALL verb (statement number 87):

```
ØØØØ87  CALL
    ØØØ4FA  D21Ø DØAØ AØF3      MVC  16Ø(17,13),243(1Ø)      TS2=8
    ØØØ5ØØ  413Ø DØAØ           LA   3,16Ø(Ø,13)             TS2=8
    ØØØ5Ø4  5Ø3Ø DØ98           ST   3,152(Ø,13)             TS2=Ø
    ØØØ5Ø8  968Ø DØ98           OI   152(13),X'8Ø'           TS2=Ø
    ØØØ5ØC  411Ø DØ98           LA   1,152(Ø,13)             TS2=Ø
    ØØØ51Ø  58FØ AØØØ           L    15,Ø(Ø,1Ø)              V(DFHEI1  )
    ØØØ514  41ØØ 9138           LA   Ø,312(Ø,9)              CLLE@=1
    ØØØ518  58CØ 9Ø8Ø           L    12,128(Ø,9)             TGTFIXD+128
    ØØØ51C  Ø5EF                BALR 14,15

    ØØØ51E  58CØ 9ØE8           L    12,232(Ø,9)             TGTFIXD+232
```

```
ØØØ522  4ØFØ 2ØØØ          STH  15,Ø(Ø,2)               RETURN-CODE
```

The CALL statement is the COBOL instruction generated as part of the CICS translation process that converted the EXEC CICS RETURN command into its COBOL natural language equivalent. The BALR instruction is part of the assembled output generated by the COBOL compilation step; this is how the flow of execution control leaves the application program and passes to CICS, in order to process the EXEC CICS command. The flow is via the CICS command-level stub, link-edited to the application program.

If the COBOL program listing is analysed further, it can be seen that the EXEC CICS RETURN command (statement 86) was commented out as part of the CICS translation process, and replaced by the COBOL CALL verb (statement 87):

```
ØØØØ86     *EXEC CICS RETURN END-EXEC.
ØØØØ87          Call 'DFHEI1' using by content
x'Øe08ØØØØØ7ØØØØ1ØØØfØfØfØf1f5  ØØØØ88     -    '4Ø4Ø4Ø' end-call.
FIELD A AND FIELD B DATA VALUES
```

Limited information is provided by EXEC CICS standard level 1 trace data. However, the contents of FIELD A and FIELD B can be of considerable use when debugging an application. In the edited trace entry example shown below, a program running as CICS task number 01310 has issued an EXEC CICS READ command against a VSAM file:

```
Ø131Ø QR    AP ØØE1 EIP   ENTRY READ    ØØ25Ø6B4 , Ø9ØØØ6Ø2
=592452=
```

The content of FIELD A is 002506B4. This is the address of the application's save area (ie for PL/I it is the program's DSA, for Assembler the EISTG, for COBOL the working storage, etc). It is useful because it can provide a very quick means of looking through a trace to identify switches in application control and determine the scope of potential loops within application processing. If further analysis of this particular task's EXEC CICS commands were required, the following trace formatting command could be used to return only the EIP level 1 trace entries:

```
ABBREV,TASKID=(Ø131Ø),TYPETR=(APØØE1)
```

Any switches between programs would be revealed by changing values in the FIELD A data, since different programs would have different save area addresses in memory. Similarly, repeated patterns of EXEC CICS commands in the trace, with matching repeated FIELD A values, may well reveal the scope of a yielding loop within the program or programs. A yielding loop is one where control returns to the CICS dispatcher within the bounds of the loop, so that (as far as CICS is concerned) a looping application is behaving validly. Loop analysis can be further enhanced by studying the return addresses in the RET fields for the EXEC CICS commands contained within the bounds of the loop and applying the technique described earlier for identifying their offsets within the looping programs.

In the example, the EXEC CICS READ command had a FIELD B value of 09000602. The first byte of the data shows that the application was executing in storage key 9 at the time it issued the EXEC CICS command. CICS runs in storage key 8, and DFHEIP will switch the storage key environment between keys 8 and 9 as appropriate when executing EXEC CICS commands. Storage protection, by means of key switching, is controlled by the STGPROT CICS system initialization parameter.

The last halfword of FIELD B is the EIBFN value for the command being executed. In the case of an EXEC CICS READ command, the EIBFN is 0602. The 06 denotes that it is a CICS file control command, the 02 that it is a READ. In the earlier EXEC CICS RETURN example, the EIBFN is 0E08, and can be seen as the start of the parameter argument data constructed by the CICS translator, in statement 87 of the COBOL program. EIBFN values are documented in the CICS *Application Programming Reference* manual.

CONCLUSION

I hope this article has helped shed some light on the use of EIP trace entries within CICS, and the role they can play in assisting with the debugging of CICS application programs.

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

‘Killing’ CICS tasks with CICS Performance Monitor for z/OS

A question often asked is, “What capabilities exist to ‘kill’ a task from a given CICS system via CICS Performance Manager?”. (Here we are using the term ‘kill’ to refer to the action provided by many performance management monitors.)

CICS Performance Monitor bases its facilities on those of CICSPlex System Manager. CICSPlex SM itself is an EXEC CICS application, and therefore provides the ability to PURGE and FORCEPURGE tasks. Those facilities are exploited by CICS PM.

Killing a task in a CICS region can have catastrophic consequences; for instance, no data integrity is guaranteed. As an example, consider the consequences of purging a task in a DB2WAIT condition. Having killed the task, the DB2 subsystem could subsequently post an ECB in the CICS system where the storage is no longer the requestor’s ECB. This could have disastrous results.

Nevertheless, the customer may deem this risk acceptable. Whilst this is a concern to CICS system programmers or operators, this function is used in the scenario where a region would have to be brought down anyway to resolve the problem, and killing the task will allow the region to continue processing critical work for some time. An example would be resolving a problem in a trading organization just before the market closes.

The problem is actually subtler than this, and can be broken down into the following items:

- Which system is the problem in?
- Which task do I want to kill?
- How do I kill it?

So, we need to know which task to kill before we can kill it.

In order to understand this topic, some appreciation is required of the underlying architecture of CICSPlex SM (upon which CICS PM is built), along with a classification of types of work running within the CICS system, and the CICS system's current state.

CICSPLEX SM AGENT ARCHITECTURE

CICSPlex SM (CPSM) provides single system image management through a network of CICS Managing Address Spaces (CMASs). These CMASs communicate with CPSM agents running in the target CICS regions. For example, these agents are responsible for INQUIREing and SETting the attributes of CICS resources.

Communication from the CMAS to the CPSM agent is achieved through a communications agent, also running in the CICS region, which mediates requests from the CMAS. Communication with this agent is via MVS cross-memory services (not CICS communication services).

The communications agent, query, and set agents, along with other CPSM infrastructure services, are initialized at CICS initialization via PLT processing. This establishes long-running tasks that, essentially, process requests for the lifetime of the CICS region. These resources run on the QR TCB. This does expose them to the possibility of being blocked because looping tasks dominate the QR TCB, or delayed through resource shortage (eg storage).

CICS WORKLOAD TYPES

CICS workloads can be classified into various types for the purposes of this discussion as follows:

- Normally executing tasks – the vast majority of CICS tasks. These are susceptible to a CEMT PURGE TASK command.
- Looping tasks – these are tasks that can be looping within themselves (and therefore susceptible to CICS runaway task detection process, ICVR process). They are looping through the CICS exec layer issuing EXEC CICS commands.
- Tasks in a purgable wait – these are susceptible to a CEMT FORCEPURGE TASK command.

- Tasks in a non-purgable wait (eg a DB2WAIT) – these are not susceptible to a CEMT FORCEPURGE command, but are susceptible to a kill command. It should be noted that the number of situations in which tasks are placed into a non-purgable wait has reduced with each CICS release. For example, dispatcher and DB2 changes in CICS TS 2.2 have resulted in the ability to purge tasks in a DB2WAIT state without the need for kill.

CICS STATE

The state of the CICS system can also play a role in being able to kill a task. If the system is at maxtask or is short on storage, then a new task (such as CEMT) cannot be dispatched. This is an instance where CPSM's long-running task architecture will be able to issue the appropriate command, whilst base CEMT would not.

The scenarios are summarized in Figure 1.

Class	CEMT	CPSM, CICS PM	Third-party tools
A	Yes	Yes	Yes – normal stuff
B	No	Yes	Yes – eg region at maxtask
C	No	No	Yes – unable to dispatch on QR TCB
D	No	No	No – the rest

Figure 1: CICS states

Class A is the majority of situations; Class B can still be performed by CPSM; Class C is a very small (and reducing) class because of CICS internal changes; Class D is minute and no-one addresses it.

CICS PURGE PROCESSING

The current CICS purge processing is as follows.

Upon receipt of a purge request, the CICS dispatcher saves the purge request information. If the purge is not successful, a purge is retried each time the task is suspended, and a deferred abend request issued. The purge may be rejected because the task is not suspended, or the task is purge/forcepurge protected.

OPPORTUNITIES TO EXTEND CICS TS IN THIS AREA

CICS PM supports CICS TS 1.3 and CICS TS 2.2. Open Transaction Environment TCBs were introduced in CICS TS 1.3. These TCBs execute independently of the CICS QR TCB. It would be possible to provide communications endpoints and INQuire/SET services from an OTE TCB. In this way, management services would still be available when the QR TCB is hung up.

Extensions to the ICVR and PURGE mechanisms could establish the need to kill a task; if not immediately attainable, this would be performed on subsequent traversal of the CICS exec layer. Other obvious places include RMI calls and DL/I command interfaces. This could exclude some of the current checks that are made to ensure data integrity. Purging of tasks could also be prioritized by age of the tasks.

SUMMARY

Some third-party performance products provide the ability to kill tasks in a CICS system. Extensions to CICS Transaction server have, in each release, reduced the need for such a function. The long-running agent architecture of CICSplex SM also reduces the number of cases in which such a function is necessary. There is, however, a very small subset of scenarios where such a function is required. We are looking at ways in which to provide such function via CICS Transaction Server, which will be utilized by a future release of CICS Performance Monitor.

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

A new CEMT program in COBOL to start and stop CICS

CICS is more than 30 years old, but it now supports applications written in C++ and Java, and it allows a single application image to be spread over several computer systems. To achieve such youthful longevity, CICS has undergone many transformations. IBM mainframe products get three levels of version number within a distinct named product. CICS is on its third distinct product name, and its fifth high-level version number. It has been maintained at two different IBM facilities since first being marketed.

An OLTP manages transactions that exhibit the following four ACID properties, as explained in Gray & Reuter's book (1993): *Atomicity, Consistency, Isolation, and Durability*. Although CICS did not have functions like journalling to support all of these properties at first, it did have them very early in its life. Like any OLTP, CICS has to interact with telecommunications networks, database managers, different programming languages, and the features and constraints of operating systems. Another way to look at CICS is that it allows a large number of users to share a relatively small number of resources with data integrity. The first versions of CICS were developed in a former IBM facility near Chicago. In 1973, IBM moved CICS development to Hursley, a village near Winchester, UK, where it has remained ever since.

Today, no on-line system could survive without some way to access the World Wide Web. IBM introduced the CICS Web Interface product in 1996. Since then, there have been various solutions and enhancements to CICS to allow applications to interact with Web browsers. The latest version, CICS Transaction Server for OS/390 Version 2.2, allows legacy application programs to interact with Web browsers as if they were 3270 terminals, and it allows new programs to present a modern interface on Web browsers. To help customers give legacy applications a new look on the World Wide Web, the CICS team created the Front End Programming Interface (FEPI) in CICS/ESA Version 3.3 in 1992. FEPI allows a CICS transaction to emulate a 3270 terminal. The idea is to write a new FEPI transaction to sit between a legacy CICS application and a Web server. That way, the legacy application, which was written for 3270 terminals, can continue to run unchanged.

I wrote a simple COBOL program to start and to stop CICS.

HOW TO INSTALL XCEM CEMT

Submit the following job to compile the source program:

```
//SRSTXCEM JOB SRS10044,SYSTEMES,CLASS=G,MSGCLASS=T,NOTIFY=&SYSUID
//         EXEC  DFHC3LCL,
//*        INDTG=ESS,
//*        DEBUG=OUI,
//         SP=',SP',
//         OUTC=T
//LIBR.SYSIN DD *
-OPT LIST
-DLM SRSTXCEM
-ADD SRSTXCEM,SEQ=/81,6,1,1/
       IDENTIFICATION DIVISION.
       PROGRAM-ID.   SRSTXCEM.
      ********************** F U N C T I O N *************************
      *                                                              *
      *                          stop CICS                           *
      *                                                              *
      ****************************************************************
      ******************************     *****************************
      *      *cics stop command
      ****************************************************************
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  W-DEBUT              PIC X(8) VALUE 'SRSTXCEM'.
       77  W-RETOUR-LIB         PIC X(9) VALUE 'EIBRESP->'.
       77  W-RETOUR             PIC S9(8) COMP VALUE +0.
       77  W-EIBFN-LIB          PIC X(9) VALUE ' EIBFN ->'.
       77  W-EIBFN              PIC X(2) VALUE SPACES.
       01  W-ANO-CN.
           05 W-APPLID          PIC X(8).
           05 FILLER            PIC X.
           05  FILLER           PIC X(50) VALUE 'UTILIZATION OF XCEM FOR
      -        ' NON AUTHORIZED USERS ('.
           05  W-USERID         PIC X(8).
           05  FILLER           PIC X(7) VALUE ') TERM('.
           05  W-TERM           PIC X(4).
           05  FILLER           PIC X VALUE ')'.
```



```
       Ø1  W-PARAM.
           Ø5 W-TRANS          PIC X(4).
           Ø5 FILLER           PIC X(76) VALUE SPACES.
      *
       PROCEDURE DIVISION.

      *
      *1- ANALYSE  COMMAND .......
      *
       1ØØ-INIT.
           IF EIBTRMID(1:2) NOT = 'CN'
              MOVE EIBTRMID TO W-TERM
              EXEC CICS ASSIGN APPLID(W-APPLID)
              END-EXEC
              EXEC CICS ASSIGN USERID(W-USERID)
              END-EXEC
              EXEC CICS WRITE OPERATOR
                   TEXT(W-ANO-CN)
                   CRITICAL
              END-EXEC
              EXEC CICS RETURN
              END-EXEC
           END-IF
           EXEC CICS RECEIVE INTO(W-PARAM)
                RESP(W-RETOUR)
           END-EXEC
           MOVE 'CEMT' TO W-TRANS
           EXEC CICS XCTL PROGRAM('DFHEMTP')
                INPUTMSG(W-PARAM)
                INPUTMSGLEN(LENGTH OF W-PARAM)
           END-EXEC.

-END
/*
```

XCEM CEMT will run on any CICS from Version 4.1 upwards.

Copy the member XCEM to a loadlib in the DFHRPL list of your desired CICS region.

Enter the following commands, into either CEDA or the batch program DFHCSDUP:

```
DEFINE TRANSACTION(XCEM) GROUP(XCEM) DESCRIPTION(XCEM CEMT)
    PROGRAM(XCEM) TASKDATALOC(ANY) PRIORITY(255)
DEFINE PROGRAM(XCEM) GROUP(XCEM) DESCRIPTION(XCEM CEMT)
    LANGUAGE(ASSEMBLER) DATALOCATION(ANY)
```

The transaction name can be changed, but should start with ‘C’ to allow it to run at Maxtasks.

Use CEDA to install the group XCEM.

Enter the transaction XCEM and enjoy!

Claude Dunand
Systems Programmer (France)

Displaying task information

The following program displays the current tasks running under CICS, with its main characteristics. The screen produced is illustrated below:

```
   CICS51TA                                              Ø3/Ø4/Ø1   Ø8:39:34
 Tasknum Tran Userid    Term Type Pri Status   Susptype Suspval   Susptime Sc Tcl
 -------------------------------------------------------------------------------
 ØØØØØ23 JNL2 STCCICS        Task 255 Suspend                     ØØØØØØØ4 S   Ø1
 ØØØØØ45 OMEG STCCICS        Task 255 Suspend USERWAIT SRVWORK   ØØØ23156 S   Ø1
 ØØØØØ46 OMEG STCCICS        Task 255 Suspend USERWAIT SR2WORK   ØØØ23156 S   Ø1
 ØØØ3364 P8EE ECCSTCV   IX76 Term ØØ1 Running                     ØØØØØØØ2 TP  Ø1
 ØØØ3365 F5RE ECCSTAW   IZ11 Term ØØ1 Running                     ØØØØØØØ2 TP  Ø1
 ØØØ3366 P8AC ECCSDEQ   IAØ1 Term ØØ1 Running                     ØØØØØØØ1 TP  Ø1
 ØØØ3367 F1CA ECCSIUG   ID29 Term ØØ1 Running                     ØØØØØØØ1 TP  Ø1
 ØØØ3368 VTAS ECICLP1   IX23 Term ØØ1 Running                     ØØØØØØØ1 TP  Ø1
```

Each time you press *Enter*, the display is refreshed. The screen is limited to 19 lines. If, in a given instant, there are more than 19 active transactions, only those fitting the screen will be dispayed.

The fields shown are task number, transaction name, userid, facility (the terminal id, if there is one), facilitytype, transaction priority, its status, suspend type and suspend value (if the task status is suspend), suspend time, startcode, and tclass. This application consists of a COBOL program and a BMS map. The transaction associated with the program (VTAS) is declared in the program’s last 77 variable, and you can change it to some other name of your choice.

VITASKP SOURCE CODE

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. VITASKP.
      *
       ENVIRONMENT DIVISION.
       DATA DIVISION.
      ****************************************************************
       WORKING-STORAGE SECTION.
      ****************************************************************
       77  X                  PIC S9(4)  COMP VALUE +Ø.
       77  Z                  PIC S9(4)  COMP VALUE +Ø.
       77  W-RESP             PIC S9(8)  COMP VALUE +Ø.
       77  W-RESP2            PIC S9(8)  COMP VALUE +Ø.
       77  ABSTIME            PIC S9(15) COMP-3.
       77  NUM-AUX            PIC 9(8)   VALUE Ø.
       77  LISTSIZE1          PIC S9(8)  COMP VALUE +Ø.
       77  LISTPTR            USAGE IS POINTER.
       77  END-MESSAGE        PIC X(3)   VALUE 'END'.
       77  TRANS-NAME         PIC X(4)   VALUE 'VTAS'.
       COPY DFHAID.
      ****************************************************************
       Ø1 COMMAREA.
      ************************************************ 1625 bytes ***
           Ø2        W-TASKNUMBER    PIC S9(7)  COMP-3.
           Ø2        T-TRANSACTION   PIC  X(4).
           Ø2        T-USERID        PIC  X(8).
           Ø2        T-FACILITY      PIC  X(4).
           Ø2        W-FACILITYTYPE  PIC S9(8)  COMP.
           Ø2        W-PRIORITY      PIC S9(8)  COMP.
           Ø2        W-RUNSTATUS     PIC S9(8)  COMP.
           Ø2        T-SUSPENDTYPE   PIC  X(8).
           Ø2        T-SUSPENDVALUE  PIC  X(8).
           Ø2        W-SUSPENDTIME   PIC S9(8)  COMP.
           Ø2        T-STARTCODE     PIC  X(2).
           Ø2        W-TCLASS        PIC S9(8)  COMP.
      *
           Ø2  VITASKSI.
              Ø4     FILLER          PIC  X(12).
              Ø4     CICSNL    COMP  PIC  S9(4).
              Ø4     CICSNF          PIC  X.
              Ø4     CICSNI          PIC  X(8).
              Ø4     DDATEL    COMP  PIC  S9(4).
              Ø4     DDATEF          PIC  X.
              Ø4     DDATEI          PIC  X(1Ø).
              Ø4     DTIMEL    COMP  PIC  S9(4).
              Ø4     DTIMEF          PIC  X.
              Ø4     DTIMEI          PIC  X(8).
              Ø4     SCREEN-LINES    PIC  X(152Ø).
              Ø4     LINEI REDEFINES SCREEN-LINES OCCURS 19.
```

```
           Ø6  LINEL     COMP  PIC  S9(4).
           Ø6  LINEA           PIC  X.
           Ø6  TASKNUMBER      PIC  X(7).
           Ø6  FILLER          PIC  X(1).
           Ø6  TRANSACTION     PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  USERID          PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  FACILITY        PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  FACILITYTYPE    PIC  X(4).
           Ø6  FILLER          PIC  X(1).
           Ø6  PRIORITY        PIC  X(3).
           Ø6  FILLER          PIC  X(1).
           Ø6  RUNSTATUS       PIC  X(7).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDTYPE     PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDVALUE    PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  SUSPENDTIME     PIC  X(8).
           Ø6  FILLER          PIC  X(1).
           Ø6  STARTCODE       PIC  X(2).
           Ø6  FILLER          PIC  X(2).
           Ø6  TCLASS          PIC  X(2).
*
     Ø2  VITASKSO REDEFINES VITASKSI PIC X(1567).
     Ø2  FILLER                PIC X(1ØØ).
*
***************************************************************
 LINKAGE SECTION.
***************************************************************
 Ø1  DFHCOMMAREA.
     Ø2  FILLER           PIC X(2ØØØ).
 Ø1  TASKLIST.
     Ø4  TASKL OCCURS 3Ø PIC S9(7) COMP-3.
***************************************************************
 PROCEDURE DIVISION.
***************************************************************
*
 FIRST-TIME-ONLY.
*===============*
     IF EIBCALEN = Ø
        MOVE LOW-VALUES TO COMMAREA
        MOVE 1625 TO EIBCALEN
        PERFORM INITIATE-SCREEN
        PERFORM INQUIRE-CICS
        PERFORM SEND-SCREEN-ERASE
        GO TO RETURN-TRANSID
     END-IF.
```



```
*
 OTHER-TIMES.
*===========*
     MOVE DFHCOMMAREA TO COMMAREA
     PERFORM RECEIVE-SCREEN
     PERFORM CLEAN-SCREEN
     PERFORM INQUIRE-CICS
     PERFORM SEND-SCREEN
     GO TO RETURN-TRANSID.
****************************************************************
*    Subroutines                                               *
****************************************************************
*
 INQUIRE-CICS.
*============*
     MOVE Ø TO X.
     EXEC CICS INQUIRE TASK LIST
          SET     (LISTPTR)
          LISTSIZE(LISTSIZE1)
     END-EXEC
     SET ADDRESS OF TASKLIST TO LISTPTR
     PERFORM INQUIRE-CICS-LOOP THRU
             INQUIRE-CICS-LOOP-EXIT UNTIL X > 19.
*
 INQUIRE-CICS-LOOP.
*=================*
     ADD 1 TO X.
     IF X > LISTSIZE1
        MOVE 99 TO X
        GO TO INQUIRE-CICS-LOOP-EXIT
     END-IF

     MOVE TASKL(X) TO W-TASKNUMBER
     EXEC CICS INQUIRE TASK        (W-TASKNUMBER)
                  TRANSACTION (T-TRANSACTION)
                  USERID      (T-USERID)
                  FACILITY    (T-FACILITY)
                  FACILITYTYPE(W-FACILITYTYPE)
                  PRIORITY    (W-PRIORITY)
                  RUNSTATUS   (W-RUNSTATUS)
                  SUSPENDTYPE (T-SUSPENDTYPE)
                  SUSPENDVALUE(T-SUSPENDVALUE)
                  SUSPENDTIME (W-SUSPENDTIME)
                  STARTCODE   (T-STARTCODE)
                  TCLASS      (W-TCLASS)
                  RESP        (W-RESP)
                  RESP2       (W-RESP2)
     END-EXEC
     IF W-RESP2 > Ø
        MOVE 99 TO X
```

```
           GO TO INQUIRE-CICS-LOOP-EXIT
        END-IF

        MOVE   W-TASKNUMBER    TO   NUM-AUX
        MOVE   NUM-AUX(2:7)    TO   TASKNUMBER(X)
        MOVE   T-TRANSACTION   TO   TRANSACTION(X)
        MOVE   T-USERID        TO   USERID(X)
        MOVE   T-FACILITY      TO   FACILITY(X)
        MOVE   W-PRIORITY      TO   NUM-AUX
        MOVE   NUM-AUX(6:3)    TO   PRIORITY(X)
        MOVE   T-SUSPENDTYPE   TO   SUSPENDTYPE(X)
        MOVE   T-SUSPENDVALUE  TO   SUSPENDVALUE(X)
        MOVE   W-SUSPENDTIME   TO   NUM-AUX
        MOVE   NUM-AUX         TO   SUSPENDTIME(X)
        MOVE   T-STARTCODE     TO   STARTCODE(X)
        MOVE   W-TCLASS        TO   NUM-AUX
        MOVE   NUM-AUX(7:2)    TO   TCLASS(X)

        IF W-RUNSTATUS    = DFHVALUE(SUSPENDED)
           MOVE 'Suspend' TO RUNSTATUS(X)
        END-IF
        IF W-RUNSTATUS    = DFHVALUE(RUNNING)
           MOVE 'Running' TO RUNSTATUS(X)
        END-IF
        IF W-RUNSTATUS    = DFHVALUE(DISPATCHABLE)
           MOVE 'Dispatc' TO RUNSTATUS(X)
        END-IF.
        IF W-FACILITYTYPE = DFHVALUE(TASK)
           MOVE 'Task'    TO FACILITYTYPE(X)
        END-IF
        IF W-FACILITYTYPE = DFHVALUE(TERM)
           MOVE 'Term'    TO FACILITYTYPE(X)
        END-IF
        IF W-FACILITYTYPE = DFHVALUE(DEST)
           MOVE 'Dest'    TO FACILITYTYPE(X)
        END-IF.
*
 INQUIRE-CICS-LOOP-EXIT.
*======================*
        EXIT.
*
 CLEAN-SCREEN.
*============*
        PERFORM CLEAN-SCREEN-LINES
                VARYING Z FROM 1 BY 1 UNTIL Z > 19.
*
 CLEAN-SCREEN-LINES.
*==================*
        MOVE SPACES TO LINEI(Z).
*
```

```
 INITIATE-SCREEN.
*===============*
     EXEC CICS ASSIGN APPLID (CICSNI)
     END-EXEC
     EXEC CICS ASKTIME ABSTIME (ABSTIME)
     END-EXEC
     EXEC CICS FORMATTIME
               ABSTIME (ABSTIME)
               DATE    (DDATEI)
               DATESEP ('/')
               TIME    (DTIMEI)
               TIMESEP (':')
     END-EXEC.
*
 RECEIVE-SCREEN.
*==============*
     EXEC CICS HANDLE CONDITION MAPFAIL(RETURN-EXIT)
     END-EXEC
     EXEC CICS RECEIVE MAP('VITASKS')
     END-EXEC.
     IF EIBAID = DFHPF3  OR EIBAID = DFHPF15
        GO TO RETURN-EXIT
     END-IF.
*
 SEND-SCREEN.
*===========*
     EXEC CICS SEND MAP('VITASKS')
                    DATAONLY
     END-EXEC.
*
 SEND-SCREEN-ERASE.
*=================*
     EXEC CICS SEND MAP('VITASKS')
                    ERASE
     END-EXEC.
*
 RETURN-TRANSID.
*==============*
     EXEC CICS RETURN
               TRANSID  (TRANS-NAME)
               COMMAREA (COMMAREA)
               LENGTH   (EIBCALEN)
     END-EXEC.
*
 RETURN-EXIT.
*===========*
     EXEC CICS SEND
               FROM   (END-MESSAGE)
               LENGTH (3)
               ERASE
```

```
           END-EXEC
           EXEC CICS RETURN
           END-EXEC
           GOBACK.
```

BMS MAP

```
MAPSET   DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB),                *
               LANG=COBOL,TIOAPFX=YES,EXTATT=MAPONLY
*
VITASKS  DFHMDI SIZE=(24,80)
*
CICSN    DFHMDF POS=(01,04),LENGTH=08,ATTRB=(ASKIP,PROT,FSET),         *
               COLOR=PINK
DDATE    DFHMDF POS=(01,57),LENGTH=10,ATTRB=(ASKIP,PROT),              *
               COLOR=PINK
DTIME    DFHMDF POS=(01,68),LENGTH=08,ATTRB=(ASKIP,PROT),              *
               COLOR=PINK
         DFHMDF POS=(02,01),LENGTH=07,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Tasknum'
         DFHMDF POS=(02,09),LENGTH=04,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Tran'
         DFHMDF POS=(02,14),LENGTH=06,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Userid'
         DFHMDF POS=(02,23),LENGTH=04,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Term'
         DFHMDF POS=(02,28),LENGTH=04,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Type'
         DFHMDF POS=(02,33),LENGTH=03,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Pri'
         DFHMDF POS=(02,37),LENGTH=06,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Status'
         DFHMDF POS=(02,45),LENGTH=08,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Susptype'
         DFHMDF POS=(02,54),LENGTH=07,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Suspval'
         DFHMDF POS=(02,63),LENGTH=08,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Susptime'
         DFHMDF POS=(02,72),LENGTH=02,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Sc'
         DFHMDF POS=(02,75),LENGTH=03,ATTRB=(ASKIP,PROT),              *
               COLOR=YELLOW,INITIAL='Tcl'
         DFHMDF POS=(03,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=RED,                                              *
               INITIAL='-----------------------------------------------*
               ------------------------------'
*
LINE-01  DFHMDF POS=(04,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
```

```
LINE-02  DFHMDF POS=(05,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-03  DFHMDF POS=(06,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-04  DFHMDF POS=(07,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-05  DFHMDF POS=(08,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-06  DFHMDF POS=(09,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-07  DFHMDF POS=(10,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-08  DFHMDF POS=(11,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-09  DFHMDF POS=(12,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-10  DFHMDF POS=(13,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-11  DFHMDF POS=(14,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-12  DFHMDF POS=(15,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-13  DFHMDF POS=(16,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-14  DFHMDF POS=(17,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-15  DFHMDF POS=(18,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-16  DFHMDF POS=(19,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-17  DFHMDF POS=(20,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-18  DFHMDF POS=(21,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
LINE-19  DFHMDF POS=(22,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE
*
         DFHMDF POS=(23,01),LENGTH=77,ATTRB=(ASKIP,PROT),              *
               COLOR=RED,                                              *
               INITIAL='-----------------------------------------------*
               ------------------------------'
         DFHMDF POS=(24,60),LENGTH=13,ATTRB=(ASKIP,PROT),              *
               COLOR=NEUTRAL,INITIAL='PF3/PF15  End'
*
         DFHMSD TYPE=FINAL
         END
```

CICS TS 2.2 threadsafe usage and performance

In today's tough economy, when a lot of companies are looking for a way to cut operations cost, CICS TS 2.2 may present an opportunity to significantly reduce CPU overhead, when DB2 is used. This is achieved by making transactions 'threadsafe' and reducing TCB switches, from QR TCB and L8 TCB.

IBM documented up to 44% CPU reduction, processing 1 million rows, comparing CICS TS 1.3 with CICS TS 2.2 with threadsafe. Although the benefits of using threadsafe can be substantial, a lot of planning and preparation has to be done in order to take advantage of this feature; also, not every shop will be able to use it because of the restrictions associated with it.

WHAT IS THREADSAFE?

Threadsafe means that two copies of your programs can run at the same time on different TCBs without problems. Traditionally CICS has required only quasi-reentrancy and used only one QR TCB, and CICS programs were stopped only by EXEC CICS commands. Running two copies of the program means they must be fully reentrant. They must also be coded so that they do not update CICS shared memory without taking action to serialize the updates.

In an open transaction environment, programs that access shared resources must be aware that these resources can also be accessed by other user tasks running in an open TCB. Programs that use appropriate serialization techniques when accessing shared resources are described as threadsafe.

If CICS is connected to DB2 V6 or later, the task-related user exit is enabled with OPENAPI automatically for you when you start the CICS/DB2 adaptor. So, if the CICS program is threadsafe, it can run under its own L8 TCB – the same one that executes the DB2 commands.

The transaction starts on the QR TCB, but, when DB2 is invoked, CICS switches control to an L8 TCB dedicated to this transaction.

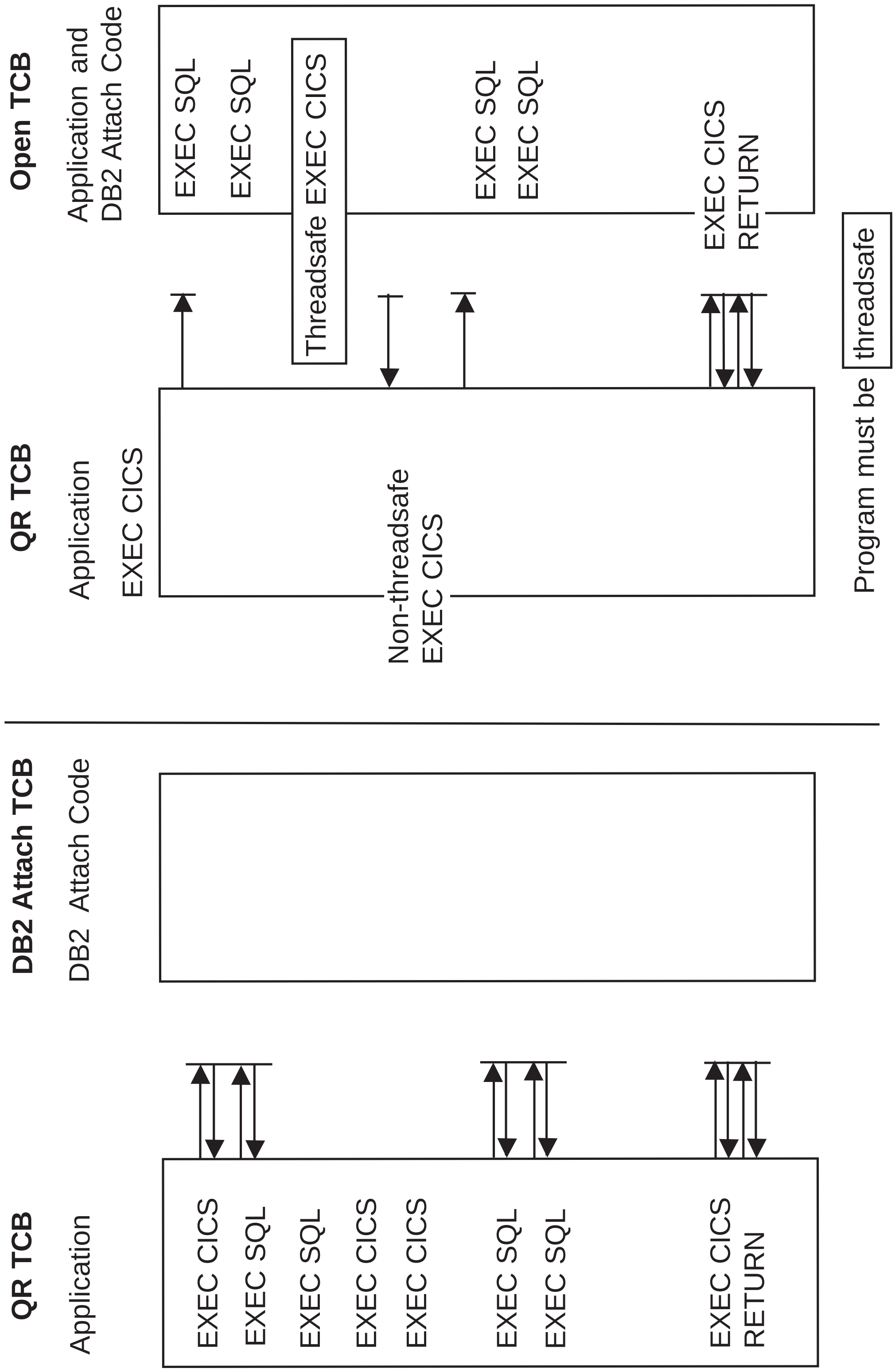

Figure 1: TCB switch processing

On return from DB2, if the application is threadsafe, there is no need to return to QR TCB. It continues to execute in the L8 TCB. Similarly, there is no need to switch TCBs to call CICS commands that are threadsafe. But, it is necessary to switch back to the QR TCB to perform some CICS commands and other functions, such as the syncpoint process. Usually, depending on the number of SQL calls and how they are interleaved with non-threadsafe EXEC CICS commands, the number of TCB switches will be reduced.

Figure 1 illustrates the TCB switch process, with and without threadsafe setting.

For most resources, such as files, DB2 tables, temporary storage queues, and transient data queues, CICS processing automatically ensures access in a threadsafe manner. But for resources that are accessed directly by user programs, like storage, the CICS CWA, global user exit global work areas, and storage acquired by EXEC CICS GETMAIN SHARED commands such as shared storage, it is the responsibility of the user program to ensure threadsafe processing. *CICS Application Programming Guide*, SC34-5702, should be reference for more details.

CONCURRENCY SETTING

CICS TS 2.2 has a new parameter under the RDO called CONCURRENCY, which tells CICS about the multi-use standards applying to the program. In particular, whether or not it can safely execute on multiple TCBs at the same time: this is the threadsafe property.

The CONCURRENCY(QUASIRENT) setting is a default for DB2 usage. Programs running with this setting will run under main CICS QR TCB and there is no performance improvement in using it.

Using the CONCURRENCY(THREADSAFE) setting, after DB2 activity, will let executing programs remain on the L8 TCB used for DB2 access. It will not swap back to use the CICS QR TCB after DB2 activity, so giving a significant performance improvement for more DB2 activity.

Be aware that application programs will swap back to the QR TCB when an EXEC CICS command, which is not threadsafe, is issued. If your application does a lot of non-threadsafe EXEC CICS commands, your response time could actually double!

To take full advantage of this threadsafe performance improvement, it is recommended that you restructure your application programs to group DB2 activity together without any intervening EXEC CICS commands.

USER EXITS AND TCBS

The CONCURRENCY parameter does not only affect programs, but it also applies to TRUEs. When used with a TRUE, it means the TRUE will obey the threadsafe multi-activity rules and so can run on an L8 TCB. You need to be aware that the DB2 TRUE is enabled with CONCURRENCY(THREADSAFE), so that DB2 activity runs under an L8 TCB.

All CICS Global User Exits can now be invoked on any of the TCBs used by CICS. Since two instances of a GLUE program can now be running at the same time on different TCBs, all GLUE programs must be threadsafe. This means that they must be coded to support multiple access to shared resources. This presents a big challenge, since all vendor-supplied GLUE programs must be checked to see whether they are threadsafe.

Some customers have experienced problems in the past using threadsafe with Candle's exit for Omegamon and with XPEDITER. Check with the vendors whether there are any outstanding issues using threadsafe with their products.

There is also the area of GLUE parmlist changes. There are new TCB modes in DFHUEPAR.UEPGIND to show which TCB type is being used. Parmlist changes affect XFAINTU, XRSINDI, and XSNON.

THREADSAFE CODING

Whenever a GLUE program uses the GLUE workarea, access should be 'critical sectioned' to prevent multiple accesses. The most common problem is when one instance is updating a counter whilst another instance gets the previous number. A similar problem may occur when the exit program accesses some shared storage. The Assembler Compare-and-Swap (CS) instruction can be used to update fields or process ECBs in an atomic fashion. Use a DFHNQEDX FUNCTION(ENQUEUE) to delimit the start of the critical section and end with an equivalent DFHNQEDX FUNCTION(DEQUEUE).

The bottom line is that access to the shared resource has to be serialized.

MIGRATION CONSIDERATIONS

Migration considerations are:

1. Check all vendor-supplied GLUEs for threadsafeness.
2. Recode your own GLUEs to obey threadsafe rules.
3. Ensure that access to shared resources is serialized.
4. Restructure your application programs to group DB2 activity together without any intervening EXEC CICS commands.
5. Establish installation standards for GLUE critical sectioning.
6. Use DFHNQEDX to establish critical sections.
7. Use CS instructions to update outside a critical section.
8. Use CS instructions to process ECBs.
9. Establish a good ENQ naming convention.

DETERMINING WHETHER A PROGRAM IS THREADSAFE

In order for the program to be threadsafe it has to be reentrant, which is achieved by using the RENT option (for LE programs) during the compile. If you are running Assembler programs, you can test them for reentrancy by linking with the RENT option and then running in a CICS region with RENTPGM=PROTECT. S0C4 abends will result if non-reentrant programs attempt to modify themselves.

With CICS TS 2.2, IBM supplies a scan utility that can identify potential non-threadsafe programs. DFHEISUP works by scanning application load modules looking for occurrences of commands found in member DFHEIDTH.

DFHEISUP will report, for example, that a program issues an ADDRESS CWA command. Because the CWA is often used to maintain counters or address chains, a program addressing the CWA could be using it in a non-threadsafe manner. On the other hand, the program could also be using the CWA to check for operational flags, file DD names, or other uses that do not raise threadsafe issues. More worrying, DFHEISUP could report no hits on an application program, leading one to believe that the program was threadsafe, when the program was maintaining counters in a shared storage location whose address is passed in the incoming COMMAREA.

While DFHEISUP is helpful in the process of identifying threadsafe applications, the only way to ensure that an application is threadsafe is to have a competent programmer who is familiar with the application review it in its entirety.

CICS.SDFHSAMP(DFHEIDTH):

```
########################################################################
# CICS LOAD MODULE SCANNER FILTER TABLE - THREADSAFE INHIBITORS        #
# This table identifies commands which "may" cause the program not to  #
# be threadsafe in that they allow accessibility to shared storage and #
# the application must have the necessary synchronization logic in     #
# place to guard against concurrent update.                            #
########################################################################
########################################################################
# The extract command gives addressibilty to a global work area of a   #
# GLUE or TRUE.                                                        #
########################################################################
EXTRACT EXIT GASET *
########################################################################
# Getmain shared storage can be shared between CICS transactions.      #
########################################################################
GETMAIN SHARED *
########################################################################
# The CWA is shared between all CICS transactions.                     #
########################################################################
ADDRESS CWA *
```

Here is an example of adding the following non-threadsafe commands to this list:

```
ASKTIME
FORMATTIME
WRITE FILE
WRITEQ TD
WRITE JOURNALNAME
SYNCPOINT
SYNCPOINT ROLLBACK
```

Load module scan using supplied DFHEIDTH:

```
//*************************************************************
//* RUN THE LOAD MODULE SCANNER
//* DFHEIDTH CHECKS FOR 'SHARED ' COMMANDS IN THREADSAFE CHECKS
//*************************************************************
//*DFHSCAN EXEC PGM=DFHEISUP,PARM=('SUMMARY, DETAILMODS'),REGION=512M
//DFHSCAN EXEC PGM=DFHEISUP,PARM=('DETAIL,ALL'),REGION=512M
//STEPLIB DD DSN= CICS.TEST.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//DFHFLTR DD DSN=CICS.SDFHSAMP(DFHEIDTH),DISP=SHR <——————————
//*DFHFLTR DD DSN=CICS.TEST.JCL(DFHEIDTH),DISP=SHR
//* MODULE LIST, TO BE CREATED DURING SUMMARY,DETAIL RUNS.
//DFHDTL DD DISP=OLD,DSN=CICS.SCANOUT
//*DFHDTL DD DSN=CICS.SCANOUT,DISP=(NEW,CATLG,DELETE),
//*  DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=8ØØØ),SPACE=(CYL,(1,1))
//* PDS CONTAINING LOADMODULES TO BE SCANNED
//DFHIN DD DISP=SHR,DSN=CICS.TEST.LOAD
```

CICS TS 2.2 ENHANCEMENT IN CICS/DB2 INTERFACE

DB2 users can benefit from the CICS/DB2 interface performance improvements – DB2 group attach and the RMI purge option. The DB2 group attach facility allows a CICS customer to exploit the DB2 facility whereby the name may be specified as a DB2 data sharing group instead of an explicit DB2 subsystem name.

Each group can contain one or more DB2 subsystems. This simplifies Application-Owning Region (AOR) cloning within a sysplex, and provides a greater level of availability. The CICS/DB2 attachment facility is enhanced to exploit CICS Open Transaction Environment (OTE) functions. When CICS TS V2.2 is connected to DB2 V6 or later, the attachment facility will execute using CICS open TCBs and will utilize DB2 functions to move DB2 connections and threads between TCBs. CICS/DB2 applications coded to threadsafe standards, and defined to CICS as threadsafe, may benefit from reduced TCB switching. For threadsafe applications making heavy use of EXEC SQL, the reduced TCB switching provides a significant performance improvement. A new Resource Manager Interface (RMI) purge option is introduced in CICS TS V2.2. This allows the writer of TRUE to specify whether, before calling it, the RMI should defer purge and deactivate runaways.

OTE

The CICS/DB2 attachment facility now exploits the Open Transaction Environment (OTE) to enable the CICS/DB2 task-related user exit to invoke and return from DB2 without switching TCBs. To gain the performance benefits of the OTE, CICS must be connected to DB2 Version 6 or later, and use a threadsafe application program. The CICS OTE was introduced to enable applications to perform, under an open TCB, actions that are not permitted under the CICS quasi-reentrant (QR) TCB.

OTE was initially exploited only by Java applications that run in a Java Virtual Machine (JVM), enabling each JVM to run under its own TCB. With the new enhancements to OTE, the open TCB modes are extended to enable task-related user exits to exploit OTE and avoid the need to manage a private pool of TCBs. The CICS/DB2 adaptor is the first task-related user exit to exploit this OTE enhancement.

CICS TS V2.2 has three separate pools of open TCBs:

- L8 – TCBs allocated for non-Java programs accessing a resource manager through a task-related user exit enabled with OPENAPI option. Used by the CICS/DB2 attachment.

  Note: the OPENAPI option specifies that the task-related user exit program is using non-CICS APIs. If OPENAPI is omitted, CICS will assume that the task-related user exit is either using only the CICS API, or that it performs its own TCB switch to invoke non-CICS services.

- H8 – TCBs allocated by hot-pool HPJ-compiled Java programs.
- J8 – TCBs allocated for the execution of a JVM program (Java programs that require a JVM).

CPU SAVING WITH THREADSAFE

IBM's Redbook *DB2 for z/OS and OS/390 Version 7 Selected Performance Topics* indicates that CPU saving with threadsafe may be as high as 44%, depending on environment.

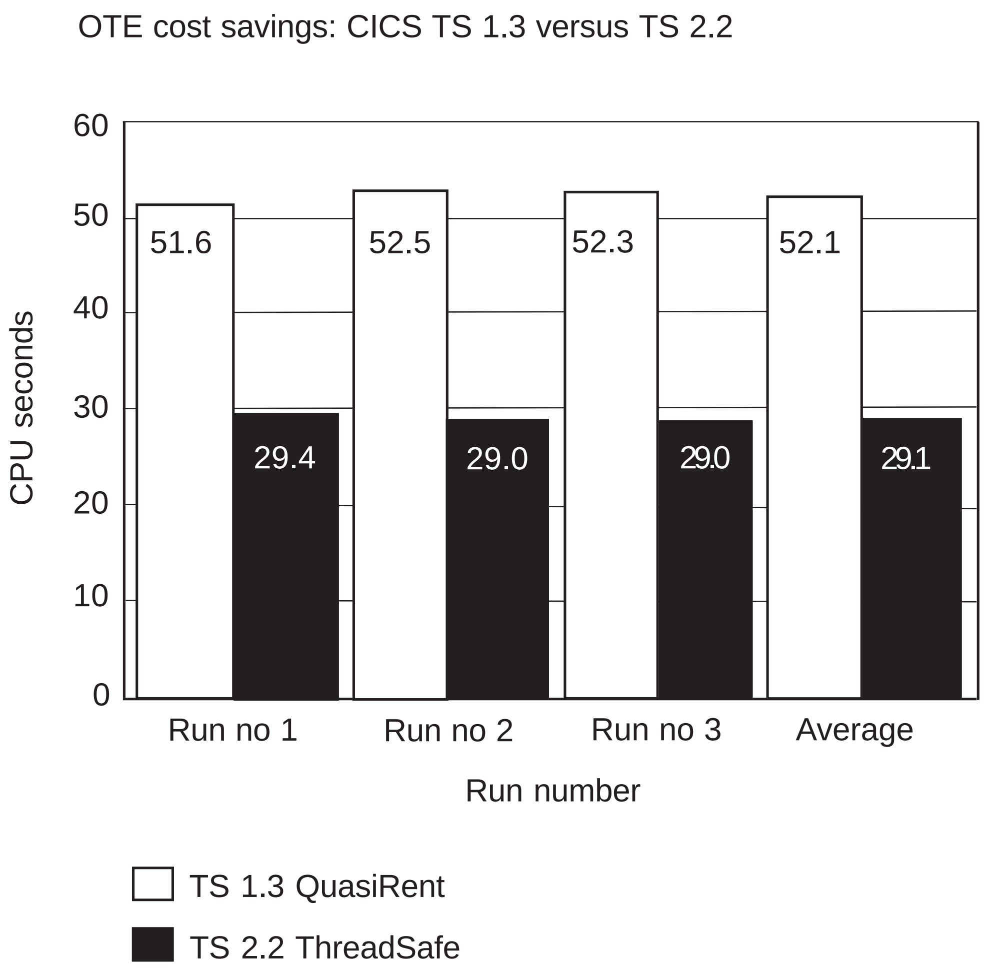

Figure 2: Cost savings with OTE

Figure 2 compares the most efficient environments on TS 1.3, CONCURRENCY(QuasiRent) and TS 2.2 CONCURRENCY(THREADSAFE). These measurements were done with CICS TS 2.2 and CICS TS 1.3 using DB2 7.1.

The delta between CICS TS 1.3 and CICS TS 2.2 of step 2 then gives us the cost savings between CICS TS 1.3 and CICS TS 2.2 running a transaction which fetches 1 million rows.

The CPU cost saving is (1 - 29.1/52.1)*100 = 44%.

IBM states that the number of instructions that were given to switch TCBs to execute an SQL, which would be a round trip, was "Approximately 1.5K instructions per SQL". That is the amount you would save if your program is threadsafe.

CONCLUSION

In the absolute best case, you can save up to 44% of CPU overhead by using threadsafe. More typical CPU saving reported by customers is 10-25%. Ensuring that a program is threadsafe is difficult and may be a time-consuming effort. Threadsafe implementation requires planning and research effort, but at the same time may bring very fruitful results, depending on installation.

Elena Nanos
IBM Certified Solution Expert in CICS Web Enablement
Zurich NA (USA)

Automatic CICS PPT management using CICS statistics and autoinstall function

INTRODUCTION

This article talks about our experience dealing with the boring problem of CICS PPT management. We hope it will help people who have the same problem.

OBJECTIVES

Our objective was to combine the flexibility of the program autoinstall function with the control over PPT definitions given by RDO.

We wanted to save on manual work (my work!), time, and money, by managing the RDO definition process with an automatic procedure. A few programs are excluded from this management (see below).

We hoped to delegate the change management system, where all application programs are defined. The task of maintaining the information needed to manage the PPT in RDO frees us from managing RDO definitions with another product.

Furthermore, by defining in RDO format only those programs that are 'statistically' used in a CICS, we gain the following benefits:

- Better CICS performance (small overhead for autoinstall activity during CICS life).
- Memory savings (no useless PPT entries).
- Faster CICS start (fewer PPT entries to install means a faster start up).

Our environment was:

- z/OS 1.3 (not relevant to our procedure).
- CICS Transaction Server 1.3 (same as above).
- Endevor as our change management system (key for our procedures).

Our project requirements were:

- Endevor.
- The program autoinstall function enabled on CICS regions.
- CICS statistics collection enabled.

DESCRIPTION

In our shop, CICS statistics are downloaded daily and expanded to produce a sequential file, where for each CICS system there is a list of all the defined programs plus a use count during the previous month (the CONT_MP field). Our PPT management procedure starts from that information. An extract from our CICS program statistics sequential file is shown below:

```
----+----1----+----2----+----3---- [...] 9----+----+-- [...]
UTILIZZO PROGRAMS CICS
APPLID    LDRPNAME      COUNT                CONT_MP
          ========      =====                =======
          NOME          N. VOLTE             CONTATORE
          PROGRAM       CHIAMATO             MESE
                        DA TRANS             PRECEDENT
[...]
CXCRP1T4  XFHMSES1      53                   128296
[...]
```

CICS statistics tell us which programs have been used in a CICS system during the previous month. These programs are defined in our Endevor environment with a system, subsystem, and type. This information is all we need to make an RDO definition statement for a program. We use an Endevor program to make a report of its managed programs. This report will be used to load a VSAM file that we call User Master Control File (UMCF), which will be read by MCFREPRT to make searches faster than reading the report sequential file (USERMCF JCL). A record in our Endevor user MCF VSAM file is shown below:

```
+----------- key ------------+
----+----1----+----2----+----3-- -+----4 [...] -8
[...]
XFHMSES1 XC      TP      PTC
[...]
```

We use two-character systems and two-character subsystems to create the RDO group name and the Endevor type to make an RDO definition statement for a program or for a mapset.

The RDO group names are created following the standard, ssbb*PA*ii, where:

- ss is the Endevor system where the program is defined.
- bb is the Endevor subsystem where the program is defined.

- *P* denotes that the group contains a program definition.

In addition to RDO DEFINE statements, the CRTRDOCB program also makes ADD statements for the list you have passed to it. An extract of CRTRDOCB output is shown below:

```
[...]
DEF PROG(MPXCINS ) G(XCTPP)
ADD G(XCTPP) LI(CELISTT4)
[...]
DEF PROG(XFHMSES1) G(XCTPP)
[...]
```

Last month's programs are defined in our Automatic RDO Define Procedure (ARDP) in RDO format into a CSD, and so they are installed when CICS starts. Programs that aren't installed when CICS starts, because they are new or they weren't used last month (there are statistically a few), will be installed dynamically by the autoinstall function. Next month they will be referenced in the CICS statistics and will be defined statically in CSD by our procedure.

Very few definitions are excluded from this system, but they include those referencing our system programs (CICS user EXITs, programs used in PLTPI and PLTSD, etc), those pertaining to products (MERVA, OMEGAMON, etc), and those few belonging to programs with particular attributes like REMOTESYSTEM or EXECKEY(CICS).

This procedure is built from two JCLs. The first, named UMCF, creates the UMCF VSAM file from the Endevor report. The second, named ARDP, uses CICS statistics and the UMCF VSAM file to update the CICS CSD file.

The UMCF JCL is shown below:

```
//*********************************************************************
//UMCF      PROC DSNSYSIN='CR1Ø157.ENDEVOR.RDO',
//          CONLIB='SYS1.SSH.ENDEVOR.BPSYA.SP.LOADLIB.APF',
//          DSNPRFX='DA.CR1Ø157'
//*********************************************************************
//* DELETE OLD DATASETS
//*********************************************************************
//DELETE    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
```

```
//SYSIN      DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.DL)
//*******************************************************************
//* ENDEVOR REPORT
//*******************************************************************
//REPORT     EXEC PGM=NDVRC1,PARM=C1BR1ØØØ,REGION=4Ø96K
//CONLIB     DD DISP=SHR,DSN=&CONLIB
//BSTRPTS    DD DISP=(NEW,CATLG,DELETE),            REPORT OUTPUT
//           DCB=(LRECL=133,RECFM=FBA,BLKSIZE=266ØØ),
//           SPACE=(CYL,(5,15),RLSE),
//           DSN=&DSNPRFX..&ENDENV..REPORTØ3
//BSTINP     DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.RP)  SELECTION CRITERIA
//BSTPDS     DD DUMMY                               FOOTPRINT DATA SET
//BSTIPT     DD DUMMY                               FOOTPRINT CRITERIA
//SMFDATA    DD DUMMY                               SMF DATA SET
//UNLINPT    DD DUMMY                               UNLOAD DATA SET
//BSTPCH     DD DISP=(NEW,PASS),
//           UNIT=SYSDA,SPACE=(CYL,(3ØØ,5Ø)),
//           DCB=(RECFM=FB,LRECL=416,BLKSIZE=416Ø),
//           DSN=&TEMP
//BSTLST     DD SYSOUT=*
//SORTIN     DD UNIT=(SYSDA,2),SPACE=(CYL,(1ØØ,15Ø)),VOL=(,,,2)
//SORTOUT    DD UNIT=(SYSDA,2),SPACE=(CYL,(1ØØ,15Ø)),VOL=(,,,2)
//SORTWKØ1   DD UNIT=SYSDA,SPACE=(CYL,(3ØØ,5Ø))
//SORTWKØ2   DD UNIT=SYSDA,SPACE=(CYL,(3ØØ,5Ø))
//SORTWKØ3   DD UNIT=SYSDA,SPACE=(CYL,(3ØØ,5Ø))
//C1MSGS1    DD SYSOUT=*
//SYSOUT     DD SYSOUT=*
//SYSPRINT   DD SYSOUT=*
//FOOTDD     DD DUMMY
//*******************************************************************
//* EXTRACT INFO FROM ENDEVOR REPORT
//*******************************************************************
//SORTRPRT   EXEC PGM=SORT
//SORTWKØ1   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ2   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ3   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ4   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ5   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ6   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ7   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ8   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ9   DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SYSOUT     DD SYSOUT=*
//SORTIN     DD DISP=SHR,DSN=&DSNPRFX..&ENDENV..REPORTØ3
//SORTOUT    DD DISP=(NEW,PASS),
//           SPACE=(CYL,(1,1),RLSE),
//           DCB=(RECFM=FB,BLKSIZE=2792Ø,LRECL=8Ø),UNIT=SYSDA,
//           DSN=&SRTRPRT
//SYSIN      DD DISP=SHR,DSN=&DSNSYSIN(#UMCFSRT)
```

```
//*****************************************************************
//* MAKE USER MCF VSAM
//*****************************************************************
//DEFUMCF   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENDENV.DF)
//*****************************************************************
//* REPRO ON USER MCF VSAM
//*****************************************************************
//REPUMCF   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//DDIN      DD DISP=SHR,DSN=&SRTRPRT
//DDOUT     DD DISP=OLD,DSN=&DSNPRFX..&ENDENV..USERMCF
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(#UMCFREP)
//*****************************************************************
//          PEND
//*****************************************************************
//MVSRIL    EXEC USERMCF,ENDENV=MVSRIL
//*****************************************************************
```

Firstly it deletes the old Endevor report sequential file and user MCF VSAM file. The &ENDENV.DL (MVSRILDL) member used as sysin for the DELETE JLC step is shown below:

```
 DELETE DA.CR1Ø157.MVSRIL.REPORTØ3
 DELETE DA.CR1Ø157.MVSRIL.USERMCF CL PURGE
 SET MAXCC=Ø
```

Next, the Endevor report utility uses the &ENDENV.RP (MVSRILRP) member as sysin for the REPORT JLC step:

```
     REPORT      Ø3 .
     ENVIRONMENT MVSRIL .
     STAGE       R .
```

The SORTRPRT SORT extracts from the Endevor report just the records we want, ie CICS COBOL programs (type PTC in our Endevor installation), CICS Assembler programs (type PTA), batch COBOL programs (type PBC; it could be a generic routine used in CICS too), batch Assembler programs (type PBA), and CICS maps (type MAP). The #UMCFSRT member used as sysin for the SORTRPRT JLC step:

```
 INCLUDE COND=(43,3,CH,EQ,C'PTA',OR,
               43,3,CH,EQ,C'PTC',OR,
               43,3,CH,EQ,C'PBA',OR,
               43,3,CH,EQ,C'PBC',OR,
```

```
              43,3,CH,EQ,C'MAP')
 SORT FIELDS=(1,49,CH,A)
 OUTREC FIELDS=(2,8,C' ',24,8,34,8,43,8)
```

The DEFUMCF step defines the user MCF. The &ENDENV.DF (MVSRILDF) member is used as sysin for the DEFUMCF JLC step:

```
 DEFINE CLUSTER -
           (NAME(DA.CR1Ø157.MVSRIL.USERMCF) -
           INDEXED -
           SHAREOPTIONS(3,3) -
           NOERASE -
           SPEED -
           NOWRITECHECK -
           NOREUSE) -
        DATA -
           (NAME(DA.CR1Ø157.MVSRIL.USERMCF.DATA) -
           KEYS(33 Ø) -
           CONTROLINTERVALSIZE(8192) -
           RECORDSIZE(8Ø 8Ø) -
           CYLINDER(1Ø 5) -
           NONSPANNED -
           VOLUMES(DAØØ3)) -
        INDEX -
           (NAME(DA.CR1Ø157.MVSRIL.USERMCF.INDEX) -
           CONTROLINTERVALSIZE(2Ø48) -
           TRACK(15 5) -
           NOIMBED -
           UNORDERED -
           NOREPLICATE -
           VOLUMES(DAØØ3))
```

The REPUMCF step loads the user MCF with those records extracted from the Endevor report by the SORTRPRT step. The #UMCFREP member is used as sysin for the REPUMCF JLC step:

```
 REPRO INFILE(DDIN) OUTFILE(DDOUT)
```

The ARDP JCL looks like:

```
//*********************************************************************
//ARDP       PROC DSNSYSIN='DA.CR1Ø157.ENDEVOR.SOURCE',
//           STATEPGM='BTD.XC77.XCWØØP17.STAT.EPGM',
//           LOADLIB='CR1Ø157.LOAD',
//           DSNPRFX='DA.CR1Ø157',
//           SDFHLOAD='PRD.CICSTS.TP.SP.SDFHLOAD',
//           DFHCSD='DA.CR1Ø157.DFHCSD'
//*********************************************************************
```

```
//DELETE    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENV.&CICSID.DEL)
//*********************************************************************
//SORTEPGM  EXEC PGM=SORT
//SORTWKØ1  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ2  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ3  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ4  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ5  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ6  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ7  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ8  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ9  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SYSOUT    DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=&STATEPGM
//SORTOUT   DD DISP=(NEW,PASS),
//             SPACE=(CYL,(1,1),RLSE),
//             DCB=(RECFM=FB,BLKSIZE=266Ø,LRECL=133),UNIT=SYSDA,
//             DSN=&SORTEPGM
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(&ENV.&CICSID.SORT)
//*********************************************************************
//MCFREPRT  EXEC PGM=MCFREPRT,PARM='RDO'
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB
//UMCF      DD DISP=SHR,DSN=&DSNPRFX..&ENDENV..USERMCF
//EPGM      DD DISP=SHR,DSN=&SORTEPGM
//PROGLIST  DD DISP=(NEW,PASS),
//             DCB=(LRECL=8Ø,RECFM=FB,BLKSIZE=2792Ø),
//             SPACE=(CYL,(1,1),RLSE),UNIT=SYSDA,
//             DSN=&PROGLIST
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSMDUMP  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSABOUT  DD SYSOUT=*
//*********************************************************************
//SORTPROG  EXEC PGM=SORT
//SORTWKØ1  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ2  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ3  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ4  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ5  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ6  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ7  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ8  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SORTWKØ9  DD UNIT=SYSDA,SPACE=(CYL,(1ØØ,1Ø))
//SYSOUT    DD SYSOUT=*
//SORTIN    DD DISP=SHR,DSN=&PROGLIST
//SORTOUT   DD DISP=(,PASS),
```



```
//          SPACE=(CYL,(1,1),RLSE),
//          DCB=(RECFM=FB,BLKSIZE=2792Ø,LRECL=8Ø),UNIT=SYSDA,
//          DSN=&SORTPROG
//SYSIN     DD DISP=SHR,DSN=&DSNSYSIN(#ARDPSRT)
//*********************************************************************
//CRTRDOCB  EXEC PGM=CRTRDOCB,PARM='&RDOLIST'
//STEPLIB   DD DISP=SHR,DSN=&LOADLIB
//PROGLIST  DD DISP=SHR,DSN=&SORTPROG
//RDODEF    DD DSN=&DSNPRFX..RDO.DEFINE.&ENV.&CICSID,
//          DISP=(,CATLG,DELETE),
//          DCB=(LRECL=8Ø,RECFM=FB,BLKSIZE=Ø),
//          SPACE=(CYL,(1,1),RLSE),UNIT=SYSDA
//SYSOUT    DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSMDUMP  DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//SYSABOUT  DD SYSOUT=*
//*********************************************************************
//CSDUPDT   EXEC PGM=DFHCSDUP,REGION=1Ø24K
//STEPLIB   DD DISP=SHR,DSN=&SDFHLOAD
//DFHCSD    DD DISP=SHR,DSN=&DFHCSD
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DISP=SHR,DSN=&DSNPRFX..RDO.DEFINE.&ENV.&CICSID
//*********************************************************************
//          PEND
//*********************************************************************
//P1T4      EXEC ARDP,ENV=P1,CICSID=T4,ENDENV=MVSRIL,RDOLIST=CELISTT4
//*********************************************************************
```

The first step deletes old output. The &ENV.&CICSID.DEL (P1T4DEL) member is used as sysin for the DELETE JLC step:

```
 DELETE DA.CR1Ø157.RDO.DEFINE.P1T4
 SET MAXCC=Ø
```

From the CICS statistics sequential file we get only those records related to a particular CICS applid, with the CONT_MP field not null and not CICS (DFH...) modules. The &ENV.&CICSID.SORT (P1T4SORT) member is used as sysin for the SORTEPGM JCL step:

```
INCLUDE COND=(1,8,CH,EQ,C'CXCRP1T4',
               AND,95,1,CH,NE,C'Ø',AND,95,1,CH,NE,C'.',
               AND,11,3,CH,NE,C'DFH')
 OPTION COPY
```

The MCFREPRT program reads a list of programs and the Endevor MCF searching for systems and subsystems:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.                MCFREPRT.
 AUTHOR.            GIANLUCA BONZANO.
*****************************************************************
*                                                               *
*---------------------------------------------------------------*
*****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
         DECIMAL-POINT IS COMMA.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  UMCF-FILE       ASSIGN TO UMCF
                             ORGANIZATION IS INDEXED
                             ACCESS MODE IS DYNAMIC
                             RECORD KEY IS UMCF-KEY
                             FILE STATUS IS UMCF-FILE-STATUS.
     SELECT  EPGM-FILE       ASSIGN TO EPGM
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS EPGM-FILE-STATUS.
     SELECT  PROGLIST-FILE   ASSIGN TO PROGLIST
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS PROGLIST-FILE-STATUS.
 DATA DIVISION.
 FILE SECTION.
 FD  UMCF-FILE
     DATA RECORD IS UMCF-REC.
 Ø1  UMCF-REC.
     Ø5  UMCF-KEY.
         Ø7  UMCF-KEY-PROGNAME       PIC X(8).
         Ø7  FILLER                  PIC X(1).
         Ø7  UMCF-KEY-SYSTEM         PIC X(8).
         Ø7  UMCF-KEY-SUBSYSTEM      PIC X(8).
         Ø7  UMCF-KEY-TYPE           PIC X(8).
     Ø5  FILLER                      PIC X(47).
 FD  EPGM-FILE
     RECORDING MODE IS F
     RECORD CONTAINS 133 CHARACTERS
     DATA RECORD EPGM-REC.
 Ø1  EPGM-REC.
     Ø5  EPGM-APPLID                 PIC X(8).
     Ø5  FILLER                      PIC X(2).
     Ø5  EPGM-LDRPNAME               PIC X(8).
     Ø5  FILLER                      PIC X(76).
     Ø5  EPGM-CONT-MP                PIC X(8).
     Ø5  FILLER                      PIC X(31).
 FD  PROGLIST-FILE
     RECORDING MODE IS F
```

```
    RECORD CONTAINS 8Ø CHARACTERS
    DATA RECORD PROGLIST-REC.
Ø1  PROGLIST-REC                       PIC X(8Ø).
WORKING-STORAGE SECTION.
77  FILLER  PIC  X(3Ø)  VALUE  '*** START  WORKING STORAGE ***'.
Ø1  UMCF-FILE-STATUS                   PIC X(2).
Ø1  EPGM-FILE-STATUS                   PIC X(2).
Ø1  PROGLIST-FILE-STATUS               PIC X(2).
Ø1  EOF-INDICATOR                      PIC 9(1) VALUE Ø.
    88  PROG-EOF                                VALUE 1.
    88  PROGLIST-EOF                            VALUE 1.
    88  UMCF-EOF                                VALUE 1.
Ø1  OUT-HEADER1.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-PROGRAM            PIC X(8) VALUE 'PROGRAM '.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-SYSTEM             PIC X(8) VALUE 'SYSTEM'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-SUBSYSTEM          PIC X(9) VALUE 'SUBSYSTEM'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER1-TYPE               PIC X(4) VALUE 'TYPE'.
    Ø5  FILLER                         PIC X(6) VALUE SPACES.
    Ø5  OUT-HEADER1-COUNT              PIC X(1Ø) VALUE 'TIMES USED'.
Ø1  OUT-HEADER2.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-PROGRAM            PIC X(8) VALUE '--------'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-SYSTEM             PIC X(8) VALUE '--------'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-SUBSYSTEM          PIC X(9) VALUE '--------'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-TYPE               PIC X(8) VALUE '--------'.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-HEADER2-COUNT              PIC X(1Ø) VALUE '----------'.
Ø1  OUT-ITEM.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-PROGRAM               PIC X(8) VALUE SPACES.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-SYSTEM                PIC X(8) VALUE SPACES.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-SUBSYSTEM             PIC X(8) VALUE SPACES.
    Ø5  FILLER                         PIC X(3) VALUE SPACES.
    Ø5  OUT-ITEM-TYPE                  PIC X(8) VALUE SPACES.
    Ø5  FILLER                         PIC X(2) VALUE SPACES.
    Ø5  OUT-ITEM-COUNT                 PIC X(8) VALUE SPACES.
Ø1  PROGLIST-ITEM.
    Ø5  PROGLIST-ITEM-PROGRAM          PIC X(8) VALUE SPACES.
    Ø5  FILLER                         PIC X(1) VALUE SPACES.
    Ø5  PROGLIST-ITEM-SYSTEM           PIC X(2) VALUE SPACES.
```

```
       Ø5  PROGLIST-ITEM-SUBSYSTEM      PIC X(2) VALUE SPACES.
       Ø5  FILLER                       PIC X(1) VALUE SPACES.
       Ø5  PROGLIST-ITEM-TYPE           PIC X(8) VALUE SPACES.
   77  FILLER  PIC  X(3Ø)  VALUE  '**** END  WORKING STORAGE ****'.
   LINKAGE SECTION.
   Ø1  PARM-DATA.
       Ø5 FILLER                PIC X(2).
       Ø5 PARM-FUNCTION         PIC X(3).
   PROCEDURE DIVISION USING PARM-DATA.
   ØØØ-MAIN.
       PERFORM ØØ1-OPEN-FILE
       DISPLAY OUT-HEADER1
       DISPLAY OUT-HEADER2
       PERFORM UNTIL PROG-EOF
          MOVE SPACES TO EPGM-REC
          READ EPGM-FILE
             AT END
                SET PROG-EOF TO TRUE
             NOT AT END
                PERFORM ØØ2-BROWSE-UMCF
          END-READ
       END-PERFORM
       PERFORM ØØ3-CLOSE-FILE
       STOP RUN
       .
   ØØ1-OPEN-FILE.
       OPEN INPUT  EPGM-FILE
                   UMCF-FILE
       IF PARM-FUNCTION = 'RDO' THEN
          OPEN OUTPUT PROGLIST-FILE
       END-IF
       .
   ØØ2-BROWSE-UMCF.
       MOVE LOW-VALUES TO UMCF-KEY
       MOVE EPGM-LDRPNAME TO UMCF-KEY-PROGNAME
       START UMCF-FILE KEY IS GREATER THAN UMCF-KEY
          INVALID KEY
             DISPLAY 'UMCF FILESTATUS: ', UMCF-FILE-STATUS
       END-START
       READ UMCF-FILE NEXT
       IF UMCF-KEY-PROGNAME = EPGM-LDRPNAME THEN
          MOVE UMCF-KEY-PROGNAME TO OUT-ITEM-PROGRAM
          MOVE UMCF-KEY-SYSTEM TO OUT-ITEM-SYSTEM
          MOVE UMCF-KEY-SUBSYSTEM TO OUT-ITEM-SUBSYSTEM
          MOVE UMCF-KEY-TYPE TO OUT-ITEM-TYPE
          IF PARM-FUNCTION = 'RDO' THEN
             MOVE UMCF-KEY-PROGNAME TO PROGLIST-ITEM-PROGRAM
             MOVE UMCF-KEY-SYSTEM TO PROGLIST-ITEM-SYSTEM
             MOVE UMCF-KEY-SUBSYSTEM TO PROGLIST-ITEM-SUBSYSTEM
```

```
                MOVE UMCF-KEY-TYPE TO PROGLIST-ITEM-TYPE
                WRITE PROGLIST-REC FROM PROGLIST-ITEM
             END-IF
          ELSE
             MOVE EPGM-LDRPNAME TO OUT-ITEM-PROGRAM
             MOVE '--' TO OUT-ITEM-SYSTEM
             MOVE '--' TO OUT-ITEM-SUBSYSTEM
             MOVE '--' TO OUT-ITEM-TYPE
          END-IF
          MOVE EPGM-CONT-MP TO OUT-ITEM-COUNT
          DISPLAY OUT-ITEM
          .
      ØØ3-CLOSE-FILE.
          CLOSE EPGM-FILE
                UMCF-FILE
          IF PARM-FUNCTION = 'RDO' THEN
             CLOSE PROGLIST-FILE
          END-IF
          .
     *----------------------------------------------------------------*
```

The program reads the sequential file created by the previous step and performs a search for systems and subsystems of input programs in the user MCF. An extract from the MCFREPRT report is shown below:

```
  PROGRAM   SYSTEM    SUBSYSTEM  TYPE      TIMES USED
  --------  --------  --------  --------  ----------
[...]
  XFHMSES1  XC        TP         PTC       128296
[...]
```

If PARM='RDO' is specified, it also reads the sequential file for program name, system+subsystem and type, like this:

```
[...]
XFHMSES1 XCTP PTC
[...]
```

SORTPROG sorts by system and subsystem the sequential file built by the previous step to identify those programs having the same system and subsystem. This prevents the next step from making duplicate ‘ADD GROUP’ RDO statements. The #ARDPSRT member is used as sysin for the SORTPROG JLC step of ARDP:

```
 SORT FIELDS=(1Ø,4,CH,A),FORMAT=CH
```

The CRTRDOCB program:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.                CRTRDOCB.
 AUTHOR.            GIANLUCA BONZANO.
****************************************************************
*                                                              *
*--------------------------------------------------------------*
****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
         DECIMAL-POINT IS COMMA.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.

     SELECT  PROGLIST-FILE   ASSIGN TO PROGLIST
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS PROGLIST-FILE-STATUS.
     SELECT  RDODEF-FILE     ASSIGN TO RDODEF
                             ORGANIZATION IS SEQUENTIAL
                             FILE STATUS IS RDODEF-FILE-STATUS.
 DATA DIVISION.
 FILE SECTION.
 FD  RDODEF-FILE
     RECORDING MODE IS F
     RECORD CONTAINS 8Ø CHARACTERS
     DATA RECORD RDODEF-REC.
 Ø1  RDODEF-REC                       PIC X(8Ø).
 FD  PROGLIST-FILE
     RECORDING MODE IS F
     RECORD CONTAINS 8Ø CHARACTERS
     DATA RECORD PROGLIST-REC.
 Ø1  PROGLIST-REC.
     Ø5  PROGLIST-PROGRAM             PIC X(8).
     Ø5  FILLER                       PIC X(1).
     Ø5  PROGLIST-SYSTEM              PIC X(2).
     Ø5  PROGLIST-SUBSYSTEM           PIC X(2).
     Ø5  FILLER                       PIC X(1).
     Ø5  PROGLIST-TYPE                PIC X(8).
 WORKING-STORAGE SECTION.
 77  FILLER  PIC  X(3Ø)  VALUE  '*** START  WORKING STORAGE ***'.
 Ø1  RDODEF-FILE-STATUS               PIC X(2).
 Ø1  PROGLIST-FILE-STATUS             PIC X(2).
 Ø1  PREV-SYSTEM                      PIC X(2) VALUE SPACES.
 Ø1  PREV-SUBSYSTEM                   PIC X(2) VALUE SPACES.
 Ø1  EOF-INDICATOR                    PIC 9(1) VALUE Ø.
     88  RDODEF-EOF                            VALUE 1.
     88  PROGLIST-EOF                          VALUE 1.
 Ø1  RDODEFINE-ITEM.
     Ø5  FILLER                       PIC X(4)  VALUE 'DEF '.
     Ø5  RDODEFINE-ITEM-RESTYPE       PIC X(4)  VALUE SPACES.
```



```
    Ø5  FILLER                          PIC X(1)  VALUE '('.
    Ø5  RDODEFINE-ITEM-PROGRAM          PIC X(8)  VALUE SPACES.
    Ø5  FILLER                          PIC X(4)  VALUE ') G('.
    Ø5  RDODEFINE-ITEM-GROUP.
        Ø7  RDODEFINE-ITEM-GROUP-SYSTEM    PIC X(2) VALUE SPACES.
        Ø7  RDODEFINE-ITEM-GROUP-SUBSYSTEM PIC X(2) VALUE SPACES.
        Ø7  FILLER                         PIC X(4) VALUE 'P   '.
    Ø5  FILLER                          PIC X(1)  VALUE ')'.
    Ø5  FILLER                          PIC X(5Ø) VALUE SPACES.
Ø1  RDOADD-ITEM.
    Ø5  FILLER                          PIC X(6)  VALUE 'ADD G('.
    Ø5  RDOADD-ITEM-GROUP               PIC X(8)  VALUE SPACES.
    Ø5  FILLER                          PIC X(5)  VALUE ') LI('.
    Ø5  RDOADD-ITEM-LIST                PIC X(8)  VALUE SPACES.
    Ø5  FILLER                          PIC X(1)  VALUE ')'.
    Ø5  FILLER                          PIC X(52) VALUE SPACES.
77  FILLER  PIC  X(3Ø)  VALUE  '**** END  WORKING STORAGE ****'.
LINKAGE SECTION.
Ø1  PARM-DATA.
    Ø5 FILLER               PIC X(2).
    Ø5 PARM-LIST            PIC X(8).
PROCEDURE DIVISION USING PARM-DATA.
ØØØ-MAIN.
    PERFORM ØØ1-OPEN-FILE
    PERFORM UNTIL PROGLIST-EOF
       MOVE SPACES TO PROGLIST-REC
       READ PROGLIST-FILE
          AT END
             SET PROGLIST-EOF TO TRUE
          NOT AT END
             PERFORM ØØ2-WRITE-RDODEF
       END-READ
    END-PERFORM
    PERFORM ØØ3-CLOSE-FILE
    STOP RUN
    .
ØØ1-OPEN-FILE.
    OPEN INPUT  PROGLIST-FILE
    OPEN OUTPUT RDODEF-FILE
    .
ØØ2-WRITE-RDODEF.
    IF PROGLIST-TYPE = 'MAP' THEN
       MOVE 'MAPS' TO RDODEFINE-ITEM-RESTYPE
    ELSE
       MOVE 'PROG' TO RDODEFINE-ITEM-RESTYPE
    END-IF
    MOVE PROGLIST-PROGRAM TO RDODEFINE-ITEM-PROGRAM
    MOVE PROGLIST-SYSTEM TO RDODEFINE-ITEM-GROUP-SYSTEM
    MOVE PROGLIST-SUBSYSTEM TO RDODEFINE-ITEM-GROUP-SUBSYSTEM
    DISPLAY RDODEFINE-ITEM
```

```
          WRITE RDODEF-REC FROM RDODEFINE-ITEM
          IF PREV-SYSTEM NOT EQUAL PROGLIST-SYSTEM OR
             PREV-SUBSYSTEM NOT EQUAL PROGLIST-SUBSYSTEM THEN
             MOVE RDODEFINE-ITEM-GROUP TO RDOADD-ITEM-GROUP
             MOVE PARM-LIST TO RDOADD-ITEM-LIST
             DISPLAY RDOADD-ITEM
             WRITE RDODEF-REC FROM RDOADD-ITEM
          END-IF
          MOVE PROGLIST-SYSTEM TO PREV-SYSTEM
          MOVE PROGLIST-SUBSYSTEM TO PREV-SUBSYSTEM
          .
      ØØ3-CLOSE-FILE.
          CLOSE PROGLIST-FILE
                RDODEF-FILE
          .
     *---------------------------------------------------------------*
```

The CRTRDOCB program reads a list of records with program, system+subsystem, and type, and then, using the RDOLIST parameters, makes 'DEFINE PROGRAM' and 'ADD GROUP' RDO statements.

In the CSDUPDT step of ARDP, the DFHCSDUP CICS utility updates the CSD file, reading the RDO statements just created.

A partitioned directory containing sysin used by the JCL is shown below:

```
  Menu  Functions  Confirm  Utilities  Help
------------------------------------------------------------------------------
EDIT              CR1Ø157.ENDEVOR.RDO                  Row ØØØØ1 of ØØØ16
Command ===>                                               Scroll ===> CSR
           Name    Prompt Size   Created         Changed          ID
_________ $ARDP              93  2ØØ4/Ø1/13  2ØØ4/Ø1/13 16:39:46  CR1Ø157
_________ $USERMCF           87  2ØØ4/Ø1/12  2ØØ4/Ø1/13 17:14:43  CR1Ø157
_________ #ARDPSRT            1  2ØØ4/Ø1/13  2ØØ4/Ø1/13 16:13:1Ø  CR1Ø157
_________ #UMCFREP            1  2ØØ4/Ø1/13  2ØØ4/Ø1/13 1Ø:23:Ø8  CR1Ø157
_________ #UMCFSRT            5  2ØØ4/Ø1/13  2ØØ4/Ø1/13 1Ø:24:1Ø  CR1Ø157
_________ MVSRILDF           25  2ØØ4/Ø1/13  2ØØ4/Ø1/13 1Ø:21:45  CR1Ø157
_________ MVSRILDL            3  2ØØ4/Ø1/13  2ØØ4/Ø1/13 12:48:47  CR1Ø157
_________ MVSRILRP            3  2ØØ4/Ø1/13  2ØØ4/Ø1/13 1Ø:2Ø:Ø9  CR1Ø157
_________ P1T4DEL             2  2ØØ4/Ø1/13  2ØØ4/Ø1/13 11:4Ø:49  CR1Ø157
_________ P1T4SORT            4  2ØØ4/Ø1/13  2ØØ4/Ø1/13 11:5Ø:42  CR1Ø157
          **End**
```

Gianluca Bonzano
Systems Programmer
Cedacri (Italy)

WANT TO SUBSCRIBE?

Each monthlyly issue of *CICS Update* is packed with ideas for improving *your* CICS installation – and all of the technical details necessary for *you* to put those ideas into practice.

With *CICS Update* ***you*** get:

- A practical toolkit of ready-made enhancements, developed and tested in a working environment by CICS experts throughout the world.
- Tutorial articles on system internals.
- Performance tuning tips and measurements.
- Early user reports on new products and releases.

Plus:

- *CICS Update* will repay the cost of subscribing many times over!

All for a fraction of the cost of a single training course! So what are *you* waiting for?

- Go to http://www.xephonUSA.com/subscribe now to subscribe!
- Subscribe now and receive 25% off of a 12-month subscription*!

***** using promotional code **C1CS21**.

* * *

WANT TO CONTRIBUTE?

CICS Update is written by technical professionals just like you with a desire to share their expert knowledge with the world.

Xephon is always seeking talented individuals to contribute articles to *CICS Update* – **and get paid for it!**

If you have insight into how to make CICS more functional, secure, reliable, user friendly, or to generally improve CICS performance, please visit http://www.xephonUSA.com/contribute for more details on how you can contribute to this definitive industry publication.