205

CICS update

December 2002

In this issue

3	CICS log exit enhancements
13	Browse and edit files under CICS
26	CICS and the CICSPlex SM Dynamic Workload Management model
38	QMF goes to the Web
44	CICS questions and answers
45	CICS news

CICS Update

Published by

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

North American office

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

Subscriptions and back-issues

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

CICS Update on-line

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

Editor

Trevor Eddolls

Contributions

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

CICS log exit enhancements

INTRODUCTION

CICS Transaction Server has been enhanced to provide a consistent means of reading and deleting specific ranges of log records from CICS-managed logstreams. This has been achieved through enhancements to the CICS log exit programs DFHLGCNV and DFHGTCNV.

OVERVIEW OF THE CICS LOG EXIT PROGRAMS

Before the advent of CICS Transaction Server, CICS log records were held on a series of BSAM-managed sequential datasets. The CICS/ESA 4.1.0 Journal Control Program (DFHJCP) was the CICS component that coordinated the writing of this data, for both CICS system logs and user journals. Once the data had been written to disk (or tape, if so defined), customers could run batch utility programs such as the CICS-supplied program DFHJUP to read these sequential datasets and so retrieve the data.

With the introduction of CICS Transaction Server, various components of CICS have been enhanced and restructured, as part of the continuing evolution of the product. This restructuring work included redesigning the recovery component (such as DFHSPP, the syncpoint program) and journalling component (such as DFHJCP). These areas of CICS Transaction Server are now managed by two new object-oriented Domains – the Recovery Manager and the Log Manager Domains. In parallel with this restructuring work, CICS journal data for both system logs and user journals is now held in logstreams, managed by the MVS System Logger address space IXGLOGR. The MVS System Logger subsystem provides a number of callable macros to allow batch and online programs an API for reading, writing, and deleting logstream data (amongst other functions). For example, the IXGWRITE macro is provided to write data to logstreams, IXGBRWSE to read back data, IXGDELET to delete data, etc.

This new MVS Logger API is suitable for use by new applications being developed to manipulate logstream data. However, a considerable number of user-written utilities (and vendor packages) already exist which invoke BSAM calls to read CICS journals. Clearly, there needed to be a way for such existing programs to be able to run unchanged when directed against journal data on CICS Transaction Server-managed logstreams as well as against CICS/ESA journals held on sequential datasets. This facility was provided by the SUBSYS parameter specified on the DD statement for the logstream in the JCL of the batch program. SUBSYS allows the specification of a number of mandatory and optional subparameters that define the batch job's usage of the logstream to the MVS Subsystem Interface (SSI).

By means of this approach at the JCL level, the batch programs themselves can be left unchanged. DFHJUP, for example, knows nothing of logstreams itself, and still issues sequential access method calls to read log data from what it believes to be a sequential journal dataset. The SSI intercepts these requests and invokes an exit routine (as named on the SUBSYS statement); this exit routine then issues the appropriate MVS System Logger macro call to process the logstream. Upon completion of this request, the exit routine returns control via the SSI back to the batch program (DFHJUP in this example). DFHJUP continues processing as if it had just completed invoking a sequential access method function.

Log records written by CICS to the CICS system log logstreams (DFHLOG and DFHSHUNT), or to user journals or forward recovery logs, have a certain format and certain characteristics. CICS therefore provides an exit routine that is designed to be used against such records. The exit routine consists of two load modules – DFHLGCNV and DFHGTCNV. MVS exit requirements need these to reside in the MVS Linklist; as such, they are installed into the CICS SDFHLINK library. The SUBSYS parameter specifies the name of DFHLGCNV as the exit routine to be invoked via the MVS SSI; DFHLGCNV loads and executes code in DFHGTCNV as required.

Since these exit routine load modules are loaded from SDFHLINK in the MVS Linklist, the highest CICS release on any given MVS image should be referenced first in the MVS Linklist concatenation. This lets batch programs always run with the highest available version of the exit routines. A given CICS release's exit routines are backwardly compatible with lower CICS releases' log records.

Below is an example of some (edited) JCL showing a SUBSYS entry for a logstream:

```
//SYSUT1   DD DSN=TESTLOG.CICSTEST.LOGANDY1,
//            DCB=BLKSIZE=3276Ø,
//            SUBSYS=(LOGR,DFHLGCNV,'TO=(2ØØ2/125,Ø8:ØØ)',DELETE)
```

The LOGR parameter defines that requests against this DD statement's dataset (ie logstream) are to be carried out by the IXGLOGR subsystem. DFHLGCNV can be seen as the name of the exit routine to be driven by the MVS SSI when a BSAM request against the dataset is to be intercepted.

THE TRADITIONAL MEANS OF PROCESSING CICS LOGSTREAMS

By using batch utilities such as DFHJUP, and exploiting the SUBSYS options in the job's JCL, users can access and retrieve their log data without the access programs having to be made 'logstream-aware' (DFHJUP still issues BSAM macros, for example). Using DFHJUP, users can print and copy logstream data, and also delete data that is no longer required. In addition, a variety of filtering choices can be specified on the OPTION and CONTROL statements of DFHJUP's SYSIN data, allowing selectivity in what log records are to be handled by DFHJUP.

One requirement for batch processing of a journal is to copy some or all of the log data to another destination (such as an archive dataset) on a daily basis, and then to delete the copied portion of the log data from the underlying logstream. Thus it is important that only successfully copied log data is then deleted. If the copy step were to fail for some reason, or if new data were written to the end of the log whilst the copy step was still running, any uncopied data on the log at the end of the copy step must not then be deleted by the delete step, or else log data will be lost without having being archived. This issue is important when considering that many logstreams can be shared by several MVS subsystems (eg a number of different CICS File Owning Regions sharing a common forward recovery journal, or Application Owning Regions sharing a common user journal for audit purposes). These regions may well be running continuously, in a 24 by 7 environment, which provides no batch window for offline processing of the log record data held on the logstreams.

Traditionally, the recommendation to achieve a consistent copy and delete has been to implement a two-jobstep solution. The first step performs the copy operation and the second performs the delete processing. The delete step uses conditional JCL and is dependent on the copy step completing successfully. The first step could invoke DFHJUP with its COPY option; the second step could simply invoke IEFBR14, and the SUBSYS DELETE option be honoured at jobstep termination. Also, the SUBSYS parameters on both steps specify the same TO= time value, and neither specifies a Julian date. Lack of a date on the TO= option causes the MVS Logger to default to the current date, meaning that the JCL does not require daily modification. Use of the same TO= time sets the same stopping point along the logstream for both the copy and the delete operations. Since logstreams allow concurrent accesses from different jobs, it is quite possible that CICS can be writing data to the logstream whilst a batch program is reading data from the same logstream. If TO= were omitted altogether, or the time value differed for the two jobsteps, it would be possible for uncopied log data to be deleted by the second job step. This is because the limiting point for the delete operation is determined at the time the second jobstep's program connects to the logstream. (Another consideration was when there was the need to perform a number of copies of the same logstream data. If several concurrent jobs were to be run, consistency of data would have required the use of the same TO= on each job's SUBSYS data, or, alternatively, an initial copy to have been taken and each job to then be run against this, rather than the original logstream.)

The recommendation was to submit the archive/delete job quite soon after the time specified on the TO= option. If company policy was to perform a daily archive and deletion of all log data on a given logstream that was written during the previous day, then submitting a two-step job shortly after (say) 8am each day would copy and delete log data written to the associated journal between 8am of the previous day and 8am of the current day. The range of log record dates to be deleted is from the oldest existing record on the log stream up until a record is reached that exceeds the TO= date/time setting.

One problem with the approach outlined above is that a TO= time value close to midnight has the potential for the copy step to span the midnight boundary. Therefore, the following delete step would run after midnight, on the following day, and so would utilize a TO= Julian date that was higher than the date used for the copy step. This could potentially delete uncopied log record data. The same result could occur if (for some reason) submission of the job was delayed so that the copy step only ran just before midnight. The TO= value could be 'low' (say 8am) but, if the copy step overran the midnight boundary, the subsequent delete step would then apply to all records written up to 8am on the following day, and once again there would be the potential for the deletion of uncopied data.

One alternative approach was to use the LASTRUN option on the SUBSYS data for the job, so that jobs could read logstreams starting from the point read to by a previous job which had specified LASTRUN. However, use of the LASTRUN keyword is mutually exclusive with SUBSYS option 1 values, which include the TO= keyword. Hence you could not set a common read and delete point and ensure that no data written to the logstream after a copy step (using LASTRUN) was not then deleted by a follow-on delete step. This is because the delete step could not specify a meaningful TO= value, consistent with where the LASTRUN step ended. Also, LASTRUN maintained a cursor position, which was updated when a job finished and the CICS log exit disconnected from the logstream. Special consideration therefore applied if a copy job that specified LASTRUN were to fail (for instance, because of lack of space on the copy step's archive dataset). The LASTRUN cursor position would have been updated to address the last block read by the failed copy step. To rerun such a copy job necessitated removing the LASTRUN keyword, and determining a FROM= point based upon what log data had been previously copied successfully.

NEW SETBRCUR, REPBRCUR, AND DELBRCUR SUBSYS OPTIONS

Given that such potential problems could occur with the traditional approach to logstream access from batch utilities, as outlined above, CICS development worked to improve the situation, and in doing so provided a consistent and integral approach to logstream access. The requirement was to allow batch jobs to process a logstream whilst it was being appended to by other subsystems, and to allow a repeat job to process the exact same logstream data, irrespective of any additional log records written to the logstream by CICS while the first job was running. The requirement also was to help facilitate automation of such off-line logstream processing, by avoiding the need to modify the SUBSYS data in the JCL before each submission, and to avoid problems with the jobs spanning midnight boundaries. Other jobs that ran against the same logstream and read the log data at the same time had to be shielded from affecting the consistency of the copy and delete job. Finally, the deletion operation had to ensure that only records that had been processed by the prior copy step (or steps) were to be deleted, and not other records that had yet to be copied. In other words, any exposure to potential data loss and inadvertent processing of new log data had to be eliminated.

These requirements have been addressed by the introduction of three new keywords on the SUBSYS interface. The keywords are SETBRCUR, REPBRCUR, and DELBRCUR. They are defined as SUBSYS option 2 values, meaning that they are provided for controlling specific log exit routine functions. In the case of CICS, this means specific to the use of the DFHLGCNV exit. Other such option 2 values include the log data formatting options COMPAT41 and COMPAT41V.

The SETBRCUR option indicates that the starting point to be read from on the logstream is where a delete cursor was set (by a prior job that used the DELBRCUR option). The delete cursor represents the log record where a job last deleted up to on the log. When SETBRCUR is used, the CICS log exit will return records from this delete cursor position through to either the youngest block on the logstream (ie the most recently written records) or a limiting position such as is specified by the TO= keyword. Once the required records have been read from the logstream, SETBRCUR causes a browse cursor to be set to reflect this position on the log. Note that the browse cursor is held at the logstream level – as such, concurrent jobs that specify SETBRCUR should be avoided. Once a browse cursor has been established, it should not be changed by another SETBRCUR job until all the required copying and deletion steps have been carried out. However, if a browse cursor has been set by a SETBRCUR job, it is possible to run another such SETBRCUR job (and hence potentially reset the browse cursor) before performing any further copying and deleting of log data. Note that the starting point for this job would be from the same delete cursor as before (as set by a prior job to use DELBRCUR), and so it would reread the same data as the previous SETBRCUR job.

Having run a job to read a specific range of log records, another step can be executed to reread the same range of records by making use of the REPBRCUR keyword. For example, you may wish to process a given day's range of log data twice, copying it to two different archive datasets; perhaps one copy is to be in new-style CICS Transaction Server log record format, and the other in CICS/ESA 4.1.0 format, by making use of the COMPAT41 formatting option on the SUBSYS JCL. By using REPBRCUR, records are read from the point specified by the delete cursor position up to the position specified by the browse cursor that was set by the last SETBRCUR operation. Use of REPBRCUR ensures that the job reads back exactly the same log records that were processed by preceding jobs – back to the one which specified SETBRCUR. (Note: this approach assumes that the delete cursor is not changed during this time.) Neither the browse nor the delete cursor is modified by use of the REPBRCUR keyword, so there is no limit to the number of jobsteps that can be run to read back the same range of log records by means of the REPBRCUR keyword.

Finally, after the last copy step has been completed, the DELBRCUR keyword can be specified on a jobstep. This tells the CICS log exit to delete log records from the logstream up to the point as specified by the browse cursor set by the previous SETBRCUR operation. Once this deletion has been performed, the delete cursor is set to this browse cursor position. This then acts as the new starting point for a subsequent SETBRCUR operation when the same set of jobsteps are rerun for the next archive and delete operation. Note that the REPBRCUR and DELBRCUR keywords do not require a TO= value to delimit their logstream activity. The end point for these jobs is taken from the browse cursor set by the previous SETBRCUR jobstep.

It is possible to rerun failing jobs that specify the new keywords. For example, if a jobstep that specified SETBRCUR or REPBRCUR were to fail midway through the run (say as the result of lack of space on a copy operation's output dataset), the step may be repeated once the problem has been resolved. The new keywords are intended to provide consistency of log stream data access.

The SETBRCUR and REPBRCUR options should not be used in conjunction with the existing LASTRUN or DELETE SUBSYS options. Similarly, DELBRCUR should not be used with these or other existing SUBSYS options. This is to ensure that the established cursor positions on the logstream are not invalidated. For the same reason, you should not run concurrent jobs that specify LASTRUN or DELETE whilst a logstream is being processed by jobs that use the new log exit options. The intention is that use of these new options provides the consistency and reusability requirements for integral logstream access, and, as such, the existing options such as LASTRUN and DELETE should no longer need to be used.

Below is an edited example of some sample JCL showing a three-step job that utilizes the new log exit SUBSYS options SETBRCUR, REPBRCUR, and DELBRCUR:

```
//JNLCOPY1 EXEC PGM=DFHJUP
//STEPLIB  DD DSN=PTFB.CICS62ØT.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=TEST.ANDY.DFHJØ3,
//          DCB=BLKSIZE=3276Ø,
//          SUBSYS=(LOGR,DFHLGCNV,,SETBRCUR)
//SYSUT4    DD DSNAME=ANDY.TEST.COPYA,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=USRPAK,
//             SPACE=(TRK,(3,1))
//SYSIN    DD *
OPTION   COPY
END
//CHKSET   IF (JNLCOPY1.RC = Ø) THEN
//JNLCOPY2 EXEC PGM=DFHJUP
//STEPLIB  DD DSN=PTFB.CICS62ØT.CICS.SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=TEST.ANDY.DFHJØ3,
//          DCB=BLKSIZE=3276Ø,
//          SUBSYS=(LOGR,DFHLGCNV,,'REPBRCUR,COMPAT41')
//SYSUT4    DD DSNAME=ANDY.TEST.COPYB,DISP=(NEW,CATLG),
//             UNIT=SYSDA,VOL=SER=USRPAK,

//             SPACE=(TRK,(3,1)),
//             DCB=(RECFM=VB,BLKSIZE=32ØØ4,LRECL=32ØØØ)
//SYSIN    DD *
OPTION   COPY  NEWDCB
END
//CHKSET  ENDIF
//CHKREP  IF (JNLCOPY2.RC = Ø) THEN
//IEFBR14   EXEC PGM=IEFBR14
//LOGSTRM DD DSNAME=TEST.ANDY.DFHJØ3,
//           SUBSYS=(LOGR,DFHLGCNV,,DELBRCUR)
//CHKREP  ENDIF
/*
```

Step 1 (JNLCOPY1) copies log data from logstream TEST.ANDY.DFHJ03 to a new dataset (ANDY.TEST.COPYA). Upon reaching the end of this logstream, the job completes, having remembered this end position by the use of SETBRCUR, which set a browse cursor. Step 2 (JNLCOPY2) uses this browse cursor when performing the REPBRCUR function in order to repeat the reading of the same logstream up to this point. Therefore, had CICS been writing to TEST.ANDY.DFHJ03 whilst step 1 was executing, any such new records would not be accessible to step 2. Step 2 copies the same range of log records to a different archive dataset (ANDY.TEST.COPYB), this time in CICS/ESA 4.1.0 format by means of the COMPAT41 option. Note that NEWDCB was also specified to allow the blocking of these output records in RECFM=VB format, to make better use of DASD space on the archive dataset. Finally, step 3 (IEFBR14) specifies DELBRCUR to cause the logstream data to be deleted up to the browse cursor position, whereupon the delete cursor is then set to the same point on the log. This prepares the environment for the next execution of the job, when this delete cursor will be used by the SETBRCUR step as the new starting point for the next copy operation.

SUMMARY

Support for the new SUBSYS keywords SETBRCUR, REPBRCUR, and DELBRCUR is present at the base level of the CICS Transaction Server 2.2 product. For CICS Transaction Server 1.3 users, the function and associated documentation changes have been provided via CICS PTF UQ65336.

Details on the various options that can be specified for the SUBSYS parameter are documented in the *CICS Operations and Utilities Guide*, in the chapter headed *Using batch jobs to read log streams*.

These enhancements to the CICS log exit processing options outlined above are to help provide a consistent approach to the reading and deleting of CICS logstream data whilst new records are potentially being added to the logstream by active CICS systems. I hope that this article has helped explain the way in which CICS log exit processing has been enhanced and now operates in CICS Transaction Server 1.3 and above.

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

Browse and edit files under CICS

Traditionally, the tool used to browse through a file or edit one of its records under CICS is the CECI transaction. Although it works well, it requires people who need to use it to also have access to all the other facilities available with that transaction – to virtually all CICS commands. Furthermore, CECI browsing or editing is not very user-friendly, since you have to define variables, know beforehand the key of the record you wish to see, and issue the necessary CICS commands one by one.

So I decided to create a more practical tool that would allow users to browse through the keys of a KSDS file and to select a record to view its contents and, optionally, modify it.

```
File: TESTE4     SDCTR21.VSAM.TESTE4                     Lrecl: Ø234
Attr: Ope Ena Rea Upd Add Bro Del                        Keylen: Ø17
Start browse key (char): ØØØØØØØØ15234                   Offset: ØØØ
                 (hexa): FØFØFØFØFØFØFØFØF1F5F2F3F4
--------------------------------------------------------------------
* ØØØØØØØØ152342431          FØFØFØFØFØFØFØFØF1F5F2F3F4F2F4F3F1
  ØØØØØØØØ155555511          FØFØFØFØFØFØFØFØF1F5F5F5F5F5F5F1F1
  ØØØØØØØØ156786411          FØFØFØFØFØFØFØFØF1F5F6F7F8F6F4F1F1
  ØØØØØØØØ167867811          FØFØFØFØFØFØFØFØF1F6F7F8F6F7F8F1F1
* ØØØØØØØØ213222211          FØFØFØFØFØFØFØFØF2F1F3F2F2F2F2F1F1
  ØØØØØØ45645644651          FØFØFØFØFØFØF4F5F6F4F5F6F4F4F6F5F1
  ØØØØØØ55555555551          FØFØFØFØFØFØF5F5F5F5F5F5F5F5F5F5F1
  ØØØØØØ55557695311          FØFØFØFØFØFØF5F5F5F5F7F6F9F5F3F1F1
  ØØØØØ2ØØØ21231311          FØFØFØFØFØF2FØFØFØF2F1F2F3F1F3F1F1
  ØØØØØ2ØØØ25434511          FØFØFØFØFØF2FØFØFØF2F5F4F3F4F5F1F1
  ØØØØØ3ØØØ33453411          FØFØFØFØFØF3FØFØFØF3F3F4F5F3F4F1F1
  ØØØØØ3ØØØ35689411          FØFØFØFØFØF3FØFØFØF3F5F6F8F9F4F1F1
  ØØØØØ4ØØØ41313411          FØFØFØFØFØF4FØFØFØF4F1F3F1F3F4F1F1
  ØØØØØ5ØØØ52342411          FØFØFØFØFØF5FØFØFØF5F2F3F4F2F4F1F1
  ØØØØØ6ØØØ62424411          FØFØFØFØFØF6FØFØFØF6F2F4F2F4F4F1F1
  ØØØØØ7ØØØ72342341          FØFØFØFØFØF7FØFØFØF7F2F3F4F2F3F4F1
  ØØØØØ8ØØØ89Ø89Ø11          FØFØFØFØFØF8FØFØFØF8F9FØF8F9FØF1F1
--------------------------------------------------------------------
  ENTER Display record        PF8/2ØNext page       PF3/15 Exit
```

Figure 1: Initial screen

Both the keys and the record contents are displayed in character and in hexadecimal, and modifications can also be done in either mode.

This tool consists of a COBOL program and two BMS maps, the first to browse through the keys of a file, and the second to view/edit the contents of one record.

The program is associated with transaction VEDI, but you can choose any other name you prefer. Just change the name in the first variable of the program.

When you launch the transaction, you will see the screen shown in Figure 1.

Type in the CICS DDname in the *File* field and, if you already know the key of the record you wish to see, type the key in either

```
File.....: TESTE4   SDCTR21.VSAM.TESTE4                 Lrecl: Ø234
Rec Key..: ØØØØØ4ØØØØ41313411                          Keylen: Ø17
           FØFØFØFØFØF4FØFØFØF4F1F3F1F3F4F1F1          Offset: ØØØ
-------------------------------------------------------------------
Dec     Ø....+....1....+    Ø . . . . + . . . . 1 . . . . +    Hex
ØØØØØ   ØØØØØ4ØØØØ4131341   FØFØFØFØFØF4FØFØFØF4F1F3F1F3F4F1   ØØØØ
ØØØ16   112ØØ2Ø7256666 A    F1F1F2FØFØF2FØF7F2F5F6F6F6F64ØC1   ØØ1Ø
ØØØ32   ntonio Soares Si    95A3969589964ØE296819985A24ØE289   ØØ2Ø
ØØØ48   lva          Ø23 1  93A5814Ø4Ø4Ø4Ø4Ø4Ø4ØFØF2F34ØF1   ØØ3Ø
ØØØ64   716866   88123      F7F1F6F8F6F64Ø4Ø4ØF8F8F1F2F34Ø4Ø   ØØ4Ø
ØØØ8Ø     è { Î } ä{ êí     ØØ235412CØØØ7633DØØ343CØØ152551D   ØØ5Ø
ØØØ96     ã  È h RlØán%     132546262C74ØØ8822D9938Ø45956C4Ø   ØØ6Ø
ØØ112                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØ7Ø
ØØ128                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØ8Ø
ØØ144                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØ9Ø
ØØ16Ø                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØAØ
ØØ176                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØBØ
ØØ192                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØCØ
ØØ2Ø8                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø   ØØDØ
ØØ224                       4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø               ØØEØ
-------------------------------------------------------------------
F3-Save/return  CLEAR-Return w/o save  F8-Next F7-Prev  ENTER-check
```

Figure 2: Viewing a record

hexadecimal or character in the *Start browse* field; otherwise just leave it blank.

Press *Enter*. The program retrieves some of the file's characteristics and displays them in the first lines. If the file is closed to CICS, you get a warning message. Otherwise, the keys of the first 17 records are displayed, one per line. Only the first 24 characters of the key are displayed, which should be enough in most cases. However, the program keeps internally the full key of each record. You can check the key characteristics and record length of the file in the upper-right corner of the screen.

If you press PF20, you will browse forward through the keys; optionally, you can type in a new *Start browse* name to jump to another departure point.

When you decide which record you want to see, simply put the cursor in the appropriate line and press *Enter* to go to the second screen, shown in Figure 2.

On the left side, you have the decimal offset of each line, followed by 16 bytes of data in character mode, followed by the same 16*2 bytes in hexadecimal mode, followed by the hexadecimal offset. You can move backward and forward through the record with PF20 and PF19. If you want to change the record's contents, just type either in the character or the hexadecimal area. In case of conflict over a specific byte, the hexadecimal typing takes precedence. If you press *Enter* after changing something, the changes will be reflected both in the hexadecimal and in the character display. If you type beyond the record's limit, that input is ignored and cleared from the screen the next time you press *Enter*.

The terminal is set to lowercase, so you can type lowercase letters in the character area. In the hexadecimal area, you can type the hexadecimal symbols A to F in either lower or uppercase. If you type them in lowercase, they will be translated to uppercase when you press *Enter* as part of the verification of valid hexadecimal input.

To leave the current record and go back to the previous screen, use PF15. If you have made any changes to the record, you will be prompted (in the last line of the screen) for whether you want to save the changes to the file or not. If you answer YES, the record will be overwritten.

You must not modify the key contents. If you do and you want to save the record, you get a warning message saying that the key was changed. In that case, you can either abandon the changes made and return to the previous screen with the clear key, or you can retype the key, which is easy to do since the key is always displayed on top of the screen.

In order to keep the COMMAREA length within a four-digit value, the program has the following restrictions: the record length of the file cannot be greater than 4096; and the keylength cannot be greater than 50 bytes. Also, only KSDS files are supported, although the program could easily be adapted to support RRDS files also.

VEDIP00 SOURCE CODE

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. VEDIPØØ.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*========================*
 77   TRANSID      PIC  X(4) VALUE 'VEDI'.
 77   X            PIC S9(4) COMP VALUE +Ø.
 77   Y            PIC S9(4) COMP VALUE +Ø.
 77   Y2           PIC S9(4) COMP VALUE +Ø.
 77   Z            PIC S9(4) COMP VALUE +Ø.
 77   K            PIC S9(4) COMP VALUE +Ø.
 77   R            PIC S9(4) COMP VALUE +Ø.
 77   RECLENDUMMY  PIC S9(4) COMP VALUE +Ø.
 77   CURSORLINE   PIC S9(4) COMP VALUE +Ø.
 77   MSG-END      PIC X(3Ø) VALUE 'Program terminated'.
 77   PAGOFFSET    PIC  9(5).
 77   LINEOFFSET   PIC  9(5).
*
 COPY DFHAID.
 Ø1  MSG-ABEND.
     Ø2 FILLER     PIC X(36)
         VALUE 'An abend has occurred. Eibresp code: '.
     Ø2 MSG-ABEND-EIBRESP  PIC 999.
*
 Ø1  WFILE-FIELDS.
```

```
     Ø2   WFILE-OPENSTATUS    PIC S9(8)  COMP.
     Ø2   WFILE-ENABLESTATUS  PIC S9(8)  COMP.
     Ø2   WFILE-READ          PIC S9(8)  COMP.
     Ø2   WFILE-UPDATE        PIC S9(8)  COMP.
     Ø2   WFILE-ADD           PIC S9(8)  COMP.
     Ø2   WFILE-BROWSE        PIC S9(8)  COMP.
     Ø2   WFILE-DELETE        PIC S9(8)  COMP.
     Ø2   WFILE-TYPE          PIC S9(8)  COMP.
     Ø2   WFILE-RECSIZE       PIC S9(8)  COMP.
     Ø2   WFILE-KEYLEN        PIC S9(8)  COMP.
     Ø2   WFILE-KEYPOS        PIC S9(8)  COMP.
     Ø2   WFILE-LEN           PIC S9(4)  COMP.
     Ø2   WFILE-LEN-KEEP      PIC S9(4)  COMP.
     Ø2   WFILE-KLEN          PIC S9(4)  COMP.
     Ø2   WFILE-NUMBER        PIC  9(8).
     Ø2   WFILE-INPUT         PIC X(256).
*
 Ø1  CHAR-TABFILE.
     Ø2  CHARTABF1.
      Ø4 AC PIC X(16) VALUE X'CØC1C2C3C4C5C6C7C8C9CACBCCCDCECF'.
      Ø4 AD PIC X(16) VALUE X'DØD1D2D3D4D5D6D7D8D9DADBDCDDDEDF'.
      Ø4 AE PIC X(16) VALUE X'EØE1E2E3E4E5E6E7E8E9EAEBECEDEEEF'.
      Ø4 AF PIC X(16) VALUE X'FØF1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'.
      Ø4 A4 PIC X(16) VALUE X'4Ø4142434445464748494A4B4C4D4E4F'.
      Ø4 A5 PIC X(16) VALUE X'5Ø5152535455565758595A5B5C5D5E5F'.
      Ø4 A6 PIC X(16) VALUE X'6Ø6162636465666768696A6B6C6D6E6F'.
      Ø4 A7 PIC X(16) VALUE X'7Ø7172737475767778797A7B7C7D7E7F'.
      Ø4 A8 PIC X(16) VALUE X'8Ø8182838485868788898A8B8C8D8E8F'.
      Ø4 A9 PIC X(16) VALUE X'9Ø9192939495969798999A9B9C9D9E9F'.
      Ø4 AA PIC X(16) VALUE X'AØA1A2A3A4A5A6A7A8A9AAABACADAEAF'.
      Ø4 AB PIC X(16) VALUE X'BØB1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
      Ø4 AØ PIC X(16) VALUE X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9ØAØBØCØDØEØF'.
      Ø4 A1 PIC X(16) VALUE X'1Ø1112131415161718191A1B1C1D1E1F'.
      Ø4 A2 PIC X(16) VALUE X'2Ø2122232425262728292A2B2C2D2E2F'.
      Ø4 A3 PIC X(16) VALUE X'3Ø3132333435363738393A3B3C3D3E3F'.
     Ø2  CHARTABF REDEFINES CHARTABF1 PIC X OCCURS 256.
*
 Ø1  CHAR-TABDISP.
     Ø2 CHARTABD1.
      Ø4 AC PIC X(16) VALUE X'CØC1C2C3C4C5C6C7C8C9CACBCCCDCECF'.
      Ø4 AD PIC X(16) VALUE X'DØD1D2D3D4D5D6D7D8D9DADBDCDDDEDF'.
      Ø4 AE PIC X(16) VALUE X'EØE1E2E3E4E5E6E7E8E9EAEBECEDEEEF'.
      Ø4 AF PIC X(16) VALUE X'FØF1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'.
      Ø4 A4 PIC X(16) VALUE X'4Ø4142434445464748494A4B4C4D4E4F'.
      Ø4 A5 PIC X(16) VALUE X'5Ø5152535455565758595A5B5C5D5E5F'.
      Ø4 A6 PIC X(16) VALUE X'6Ø6162636465666768696A6B6C6D6E6F'.
      Ø4 A7 PIC X(16) VALUE X'7Ø7172737475767778797A7B7C7D7E7F'.
      Ø4 A8 PIC X(16) VALUE X'8Ø8182838485868788898A8B8C8D8E8F'.
      Ø4 A9 PIC X(16) VALUE X'9Ø9192939495969798999A9B9C9D9E9F'.
      Ø4 AA PIC X(16) VALUE X'AØA1A2A3A4A5A6A7A8A9AAABACADAEAF'.
      Ø4 AB PIC X(16) VALUE X'BØB1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
      Ø4 AØ PIC X(16) VALUE X'4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø'.
      Ø4 A1 PIC X(16) VALUE X'4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø'.
      Ø4 A2 PIC X(16) VALUE X'4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø'.
      Ø4 A3 PIC X(16) VALUE X'4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø4Ø'.
     Ø2  CHARTABD REDEFINES CHARTABD1 PIC X OCCURS 256.
*
 Ø1  HEX-TAB.
     Ø2  HEXTAB1.
      Ø4 HC PIC X(32) VALUE 'CØC1C2C3C4C5C6C7C8C9CACBCCCDCECF'.
      Ø4 HD PIC X(32) VALUE 'DØD1D2D3D4D5D6D7D8D9DADBDCDDDEDF'.
      Ø4 HE PIC X(32) VALUE 'EØE1E2E3E4E5E6E7E8E9EAEBECEDEEEF'.
      Ø4 HF PIC X(32) VALUE 'FØF1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'.
      Ø4 H4 PIC X(32) VALUE '4Ø4142434445464748494A4B4C4D4E4F'.
      Ø4 H5 PIC X(32) VALUE '5Ø5152535455565758595A5B5C5D5E5F'.
      Ø4 H6 PIC X(32) VALUE '6Ø6162636465666768696A6B6C6D6E6F'.
      Ø4 H7 PIC X(32) VALUE '7Ø7172737475767778797A7B7C7D7E7F'.
      Ø4 H8 PIC X(32) VALUE '8Ø8182838485868788898A8B8C8D8E8F'.
      Ø4 H9 PIC X(32) VALUE '9Ø9192939495969798999A9B9C9D9E9F'.
      Ø4 HA PIC X(32) VALUE 'AØA1A2A3A4A5A6A7A8A9AAABACADAEAF'.
      Ø4 HB PIC X(32) VALUE 'BØB1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
      Ø4 HØ PIC X(32) VALUE 'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9ØAØBØCØDØEØF'.
      Ø4 H1 PIC X(32) VALUE '1Ø1112131415161718191A1B1C1D1E1F'.
      Ø4 H2 PIC X(32) VALUE '2Ø2122232425262728292A2B2C2D2E2F'.
      Ø4 H3 PIC X(32) VALUE '3Ø3132333435363738393A3B3C3D3E3F'.
     Ø2  HEXTAB REDEFINES HEXTAB1 PIC X(2) OCCURS 256.
*
 Ø1  SCREEN-OFFSET.
     Ø2  HEXA-PAG PIC X(16) VALUE 'Ø123456789ABCDEF'.
     Ø2  HP REDEFINES HEXA-PAG PIC X OCCURS 16.
     Ø2  HEXA-OFF PIC X(32)
                    VALUE 'ØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9ØAØBØCØDØEØFØ'.
     Ø2  HO REDEFINES HEXA-OFF PIC XX OCCURS 16.
     Ø2  CHAR-OFF.
       Ø3  FILLER PIC X(24) VALUE 'ØØØØ16Ø32Ø48Ø64Ø8ØØ96112'.
       Ø3  FILLER PIC X(24) VALUE '12814416Ø1761922Ø822424Ø'.
     Ø2  CO REDEFINES CHAR-OFF PIC 9(3) OCCURS 16.
*
 Ø1  COMMAREA.
     Ø2  INDICA     PIC 9.
     Ø2  RECALTERED PIC X.
     Ø2  PAG        PIC 999.
     Ø2  PAG-MAX    PIC 999.
     Ø2  REMBYTES   PIC S9(4) COMP.
     Ø2  REMLINES   PIC S9(4) COMP.
     Ø2  REMCHAR    PIC S9(4) COMP.
     Ø2  REMCHAR2   PIC S9(4) COMP.
     Ø2  RECLEN     PIC S9(4) COMP.
     Ø2  CURRKEY    PIC X(5Ø).
     Ø2  STARTKEY   PIC X(5Ø).
     Ø2  FILEKEY    PIC X(5Ø) OCCURS 17.
     Ø2  FILE-IN.
```

```
          Ø3  FILEC-IN                                OCCURS 16.
           Ø4  FILEC-IN1                    PIC X(256).
           Ø4  FILEC    REDEFINES FILEC-IN1 PIC X     OCCURS 256.
           Ø4  FILEC16  REDEFINES FILEC-IN1           OCCURS 16.
            Ø5 FILEC161                     PIC X     OCCURS 16.
         Ø2  SCREEN-OUT.
          Ø3   FILEC-OUT                    PIC X(256).
          Ø3   FILECO   REDEFINES FILEC-OUT PIC X     OCCURS 256.
          Ø3   FILECO16 REDEFINES FILEC-OUT           OCCURS 16.
           Ø4  FILECO161                    PIC X     OCCURS 16.
          Ø3   FILEH-OUT                    PIC X(512).
          Ø3   FILEHO   REDEFINES FILEH-OUT PIC X(2)  OCCURS 256.
          Ø3   FILEHO16 REDEFINES FILEH-OUT           OCCURS 16.
           Ø4  FILEHO161                    PIC X(2)  OCCURS 16.
          Ø3   FILEHO32 REDEFINES FILEH-OUT           OCCURS 16.
           Ø4  FILEHO321                    PIC X     OCCURS 32.
    *
         Ø2    VEDISØØI.
           Ø4     FILLER          PIC   X(12).
           Ø4     DDNAMEØL  COMP  PIC   S9(4).
           Ø4     DDNAMEØF        PIC   X(Ø1).
           Ø4     DDNAMEØI        PIC   X(Ø8).
           Ø4     DSNAMEØL  COMP  PIC   S9(4).
           Ø4     DSNAMEØF        PIC   X(Ø1).
           Ø4     DSNAMEØI        PIC   X(44).
           Ø4     RECSIZØL  COMP  PIC   S9(4).
           Ø4     RECSIZØF        PIC   X(Ø1).
           Ø4     RECSIZØI        PIC   9(Ø4).
           Ø4     STATUSL   COMP  PIC   S9(4).
           Ø4     STATUSF         PIC   X(Ø1).
           Ø4     STATUSI         PIC   X(3Ø).
           Ø4     KEYLENØL  COMP  PIC   S9(4).
           Ø4     KEYLENØF        PIC   X(Ø1).
           Ø4     KEYLENØI        PIC   9(Ø3).
           Ø4     STARTKCL  COMP  PIC   S9(4).
           Ø4     STARTKCF        PIC   X(Ø1).
           Ø4     STARTKCI        PIC   X(38).
           Ø4     KEYPOSØL  COMP  PIC   S9(4).
           Ø4     KEYPOSØF        PIC   X(Ø1).
           Ø4     KEYPOSØI        PIC   9(Ø3).
           Ø4     STARTKHL  COMP  PIC   S9(4).
           Ø4     STARTKHF        PIC   X(Ø1).
           Ø4     STARTKHI        PIC   X(5Ø).
           Ø4     STARTKHI1  REDEFINES STARTKHI PIC XX OCCURS 25.
           Ø4     STARTKHI11 REDEFINES STARTKHI PIC X  OCCURS 5Ø.
           Ø4     KEYLINES        PIC   X(1394).
           Ø4     KEYLINI    REDEFINES KEYLINES OCCURS 17.
              Ø6  SELCØL    COMP  PIC   S9(4).
              Ø6  SELCØF          PIC   X(Ø1).
              Ø6  SELCØI          PIC   X(Ø1).
              Ø6  KEYCHAL   COMP  PIC   S9(4).
              Ø6  KEYCHAF         PIC   X(Ø1).
              Ø6  KEYCHAI         PIC   X(24).
              Ø6  KEYHEXL   COMP  PIC   S9(4).
              Ø6  KEYHEXF         PIC   X(Ø1).
              Ø6  KEYHEXI         PIC   X(48).
           Ø4     ERROØL    COMP  PIC   S9(4).
           Ø4     ERROØF          PIC   X(Ø1).
           Ø4     ERROØI          PIC   X(5Ø).
         Ø2   VEDISØØO REDEFINES VEDISØØI PIC X(1663).
    *
         Ø2 VEDISØ1I.
           Ø4     FILLER          PIC   X(12).
           Ø4     DDNAME1L  COMP  PIC   S9(4).
           Ø4     DDNAME1F        PIC   X(Ø1).
           Ø4     DDNAME1I        PIC   X(Ø8).
           Ø4     DSNAME1L  COMP  PIC   S9(4).
           Ø4     DSNAME1F        PIC   X(Ø1).
           Ø4     DSNAME1I        PIC   X(41).
           Ø4     RECSIZ1L  COMP  PIC   S9(4).
           Ø4     RECSIZ1F        PIC   X(Ø1).
           Ø4     RECSIZ1I        PIC   9(Ø4).
           Ø4     RECKEYDL  COMP  PIC   S9(4).
           Ø4     RECKEYDF        PIC   X(Ø1).
           Ø4     RECKEYDI        PIC   X(48).
           Ø4     KEYLEN1L  COMP  PIC   S9(4).
           Ø4     KEYLEN1F        PIC   X(Ø1).
           Ø4     KEYLEN1I        PIC   9(Ø3).
           Ø4     RECKEYHL  COMP  PIC   S9(4).
           Ø4     RECKEYHF        PIC   X(Ø1).
           Ø4     RECKEYHI        PIC   X(48).
           Ø4     KEYPOS1L  COMP  PIC   S9(4).
           Ø4     KEYPOS1F        PIC   X(Ø1).
           Ø4     KEYPOS1I        PIC   9(Ø3).
           Ø4   DATLINI OCCURS 16.
              Ø6  OFSETDL   COMP  PIC   S9(4).
              Ø6  OFSETDF         PIC   X(Ø1).
              Ø6  OFSETDI         PIC   X(Ø5).
              Ø6  DTCHARL   COMP  PIC   S9(4).
              Ø6  DTCHARF         PIC   X(Ø1).
              Ø6  DTCHARI         PIC   X(16).
              Ø6  DTCHARI1  REDEFINES DTCHARI PIC X OCCURS 16.
              Ø6  DTHEXAL   COMP  PIC   S9(4).
              Ø6  DTHEXAF         PIC   X(Ø1).
              Ø6  DTHEXAI         PIC   X(32).
              Ø6  DTHEXAI1  REDEFINES DTHEXAI PIC XX OCCURS 16.
              Ø6  DTHEXAI11 REDEFINES DTHEXAI PIC X  OCCURS 32.
              Ø6  OFSETHL   COMP  PIC   S9(4).
              Ø6  OFSETHF         PIC   X(Ø1).
              Ø6  OFSETHI         PIC   X(Ø4).
           Ø4     ERRO1L    COMP  PIC   S9(4).
           Ø4     ERRO1F          PIC   X(Ø1).
```

```
          Ø4     ERRO1I          PIC   X(35).
          Ø4     ANSWERL   COMP  PIC   S9(4).
          Ø4     ANSWERF         PIC   X(Ø1).
          Ø4     ANSWERI         PIC   X(Ø3).
        Ø2   VEDISØ1O REDEFINES VEDISØ1I PIC X(1336).
        Ø2   FILLER PIC X(5ØØ).
*
 LINKAGE SECTION.
*================*
 Ø1  DFHCOMMAREA.
        Ø2   FILLER PIC X(95ØØ).
*==========================================================*
 PROCEDURE DIVISION.
*==========================================================*
 FIRST-TIME-ONLY.
*================*
        IF EIBCALEN = Ø
           MOVE LOW-VALUES TO COMMAREA
           MOVE 89ØØ  TO EIBCALEN
           MOVE Ø     TO INDICA PAGOFFSET
           MOVE 1     TO PAG
           MOVE 4Ø96  TO RECLEN
           MOVE -1    TO DDNAMEØL
           PERFORM SEND-MAPØ
           GO TO RETURN-TRANSID
        END-IF.
*
 OTHER-TIMES.
*============*
        EXEC CICS HANDLE ABEND LABEL (ABEND-PROGRAM)
        END-EXEC
        MOVE DFHCOMMAREA TO COMMAREA
*
        IF INDICA = Ø
           PERFORM RECEIVE-MAPØ
           IF DDNAMEØL = Ø
              MOVE -1 TO DDNAMEØL
              GO TO ERRO-INPUTØ
           END-IF
           PERFORM DATASET-INQUIRE
           PERFORM DATASET-START-BROWSE THRU DATASET-END-BROWSE
           PERFORM SET-LOWERCASE
           PERFORM SEND-MAPØ
           MOVE 1 TO INDICA
           GO TO RETURN-TRANSID
        END-IF
*
        IF INDICA = 1
           PERFORM RECEIVE-MAPØ
           IF STARTKCL NOT = Ø OR STARTKHL NOT = Ø
           OR EIBAID = DFHPF8 OR EIBAID = DFHPF2Ø
```

```
            PERFORM DATASET-START-BROWSE THRU DATASET-END-BROWSE
            PERFORM SEND-MAPØ
            GO TO RETURN-TRANSID
         END-IF
         IF EIBAID = DFHENTER
            COMPUTE CURSORLINE = EIBCPOSN / 8Ø - 4
            IF CURSORLINE < 1 OR > 17
               OR KEYHEXI(CURSORLINE) = LOW-VALUES
               MOVE -1 TO SELCØL(1)
               PERFORM SEND-MAPØ
            ELSE
               MOVE DDNAMEØI            TO DDNAME1I
               MOVE DSNAMEØI            TO DSNAME1I
               MOVE KEYLENØI            TO KEYLEN1I
               MOVE KEYPOSØI            TO KEYPOS1I
               MOVE RECSIZØI            TO RECSIZ1I
               MOVE KEYCHAI(CURSORLINE) TO RECKEYDI
               MOVE KEYHEXI(CURSORLINE) TO RECKEYHI
               MOVE FILEKEY(CURSORLINE) TO CURRKEY
               MOVE '*' TO SELCØI(CURSORLINE)
               PERFORM DATASET-READ
               MOVE LOW-VALUES TO FILEC-OUT FILEH-OUT
               PERFORM FILEIN-TO-SCREENOUT
                       VARYING X FROM 1 BY 1 UNTIL X > 256
                       AFTER   Y FROM 1 BY 1 UNTIL Y > 256
               PERFORM LOAD-MAP1
                       VARYING K FROM 1 BY 1 UNTIL K > 16
               MOVE 2 TO INDICA
               PERFORM SEND-MAP1
               GO TO RETURN-TRANSID
            END-IF
         END-IF
         PERFORM SEND-MAPØ
         GO TO RETURN-TRANSID
      END-IF
*
      IF INDICA = 2
         PERFORM RECEIVE-MAP1
         PERFORM CHECK-LINES
                 VARYING K FROM 1 BY 1 UNTIL K > 16
         IF EIBAID = DFHPF3 OR = DFHPF15
            IF RECALTERED = 1
               MOVE 3 TO INDICA
               GO TO MESSAGE-QUESTION
            ELSE
               GO TO CLEAR1
            END-IF
         END-IF
         IF EIBAID = DFHPF8 OR = DFHPF2Ø
            IF PAG = PAG-MAX
               GO TO ERRO-PAG-MAX
            END-IF
            ADD 1 TO PAG
            COMPUTE PAGOFFSET = (PAG - 1) * 256
            MOVE LOW-VALUES TO FILEC-OUT FILEH-OUT
            PERFORM FILEIN-TO-SCREENOUT
                    VARYING X FROM 1 BY 1 UNTIL X > 256
                    AFTER   Y FROM 1 BY 1 UNTIL Y > 256
            PERFORM LOAD-MAP1
                    VARYING K FROM 1 BY 1 UNTIL K > 16
         END-IF
         IF EIBAID = DFHPF7 OR = DFHPF19
            IF PAG = 1
               GO TO ERRO-PAG-MIN
            END-IF
            SUBTRACT 1 FROM PAG
            COMPUTE PAGOFFSET = (PAG - 1) * 256
            MOVE LOW-VALUES TO FILEC-OUT FILEH-OUT
            PERFORM FILEIN-TO-SCREENOUT
                    VARYING X FROM 1 BY 1 UNTIL X > 256
                    AFTER   Y FROM 1 BY 1 UNTIL Y > 256
            PERFORM LOAD-MAP1
                    VARYING K FROM 1 BY 1 UNTIL K > 16
         END-IF
         PERFORM SEND-MAP1-DATAONLY
         GO TO RETURN-TRANSID
      END-IF
 *
      IF INDICA = 3
         PERFORM RECEIVE-MAP1
         IF ANSWERI = 'yes' OR = 'YES'
            PERFORM DATASET-WRITE
            MOVE 'Record written' TO ERROØI
         END-IF
         GO TO CLEAR1
      END-IF.
 *===========================================================*
 *    Subroutines
 *===========================================================*
  DATASET-INQUIRE.
 *================*
      EXEC CICS HANDLE CONDITION FILENOTFOUND(ERRO-FILENOTFND)
      END-EXEC
      EXEC CICS INQUIRE FILE  (DDNAMEØI)
                DSNAME        (DSNAMEØI)
                OPENSTATUS    (WFILE-OPENSTATUS)
                ENABLESTATUS  (WFILE-ENABLESTATUS)
                READ          (WFILE-READ)
                UPDATE        (WFILE-UPDATE)
                ADD           (WFILE-ADD)
                BROWSE        (WFILE-BROWSE)
                DELETE        (WFILE-DELETE)
```

```
                TYPE          (WFILE-TYPE)
                RECORDSIZE    (WFILE-RECSIZE)
                KEYLENGTH     (WFILE-KEYLEN)
                KEYPOSITION   (WFILE-KEYPOS)
      END-EXEC
      IF WFILE-RECSIZE > RECLEN
         GO TO ERRO-RECSIZE
      END-IF.
      IF WFILE-OPENSTATUS = DFHVALUE(OPEN)
         MOVE 'Ope'  TO  STATUSI(1:3)
      ELSE
         GO TO ERRO-NOTOPEN
      END-IF.
      IF WFILE-ENABLESTATUS = DFHVALUE(ENABLED)
         MOVE 'Ena'  TO  STATUSI(5:3)
      ELSE
         GO TO ERRO-NOTENABLED
      END-IF.
      IF WFILE-TYPE NOT = DFHVALUE(KSDS)
         GO TO ERRO-NOTKSDS
      END-IF.
      IF WFILE-READ = DFHVALUE(READABLE)
         MOVE 'Rea'  TO  STATUSI(9:3)
      END-IF.
      IF WFILE-UPDATE = DFHVALUE(UPDATABLE)
         MOVE 'Upd'  TO  STATUSI(13:3)
      END-IF.
      IF WFILE-ADD = DFHVALUE(ADDABLE)
         MOVE 'Add'  TO  STATUSI(17:3)
      END-IF.
      IF WFILE-BROWSE = DFHVALUE(BROWSABLE)
         MOVE 'Bro'  TO  STATUSI(21:3)
      END-IF.
      IF WFILE-DELETE = DFHVALUE(DELETABLE)
         MOVE 'Del'  TO  STATUSI(25:3)
      END-IF.
      MOVE  WFILE-RECSIZE   TO  WFILE-NUMBER
      MOVE  WFILE-NUMBER    TO  RECSIZØI
      MOVE  WFILE-KEYLEN    TO  WFILE-NUMBER WFILE-KLEN
      MOVE  WFILE-NUMBER    TO  KEYLENØI
      MOVE  WFILE-KEYPOS    TO  WFILE-NUMBER
      MOVE  WFILE-NUMBER    TO  KEYPOSØI
      ADD WFILE-KEYPOS WFILE-KEYLEN GIVING WFILE-LEN-KEEP.
*
 DATASET-START-BROWSE.
*====================*
      IF STARTKCL NOT = Ø AND STARTKHL = Ø
         MOVE STARTKCI TO STARTKEY
      END-IF
      IF STARTKHL > Ø
         PERFORM CHECK-HEXCHARS-STARTKEYH
```

```
                   VARYING R FROM 1 BY 1 UNTIL R > 5Ø
           MOVE LOW-VALUES TO STARTKEY
           PERFORM TRANSLATE-TO-CHAR-STARTKEYH
                   VARYING R FROM 1 BY 1 UNTIL R > 25
                                            OR R > KEYLENØI
                   AFTER   Y FROM 1 BY 1 UNTIL Y > 256
        END-IF
        EXEC CICS HANDLE CONDITION ENDFILE(DATASET-END-BROWSE)
        END-EXEC
        EXEC CICS HANDLE CONDITION NOTFND(ERRO-NOT-FOUND)
        END-EXEC
        EXEC CICS IGNORE CONDITION LENGERR
        END-EXEC
        EXEC CICS STARTBR DATASET   (DDNAMEØI)
                          RIDFLD    (STARTKEY)
                          KEYLENGTH (WFILE-KLEN)
        END-EXEC
        MOVE LOW-VALUES TO KEYLINES
        MOVE -1  TO SELCØL(1)
        MOVE 'Z' TO DDNAMEØF
        PERFORM READ-NEXT-DATASET
                VARYING X FROM 1 BY 1 UNTIL X > 18.
*
 DATASET-END-BROWSE.
*===================*
        EXEC CICS ENDBR DATASET (DDNAMEØI)
        END-EXEC.
*
 READ-NEXT-DATASET.
*==================*
        MOVE WFILE-LEN-KEEP TO WFILE-LEN
        EXEC CICS READNEXT DATASET (DDNAMEØI)
                           RIDFLD  (STARTKEY)
                           INTO    (WFILE-INPUT)
                           LENGTH  (WFILE-LEN)
        END-EXEC
        IF X < 18
           MOVE STARTKEY TO FILEKEY(X)
           MOVE 'A' TO SELCØF(X)
           PERFORM TRANSLATE-TO-HEXA-KEY
                   VARYING Y FROM 1 BY 1 UNTIL Y > 24
                                            OR Y > KEYLENØI
                   AFTER   Z FROM 1 BY 1 UNTIL Z > 256
        END-IF.
*
```

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

Systems Programmer (Portugal) © Xephon 2002

CICS and the CICSPlex SM Dynamic Workload Management model

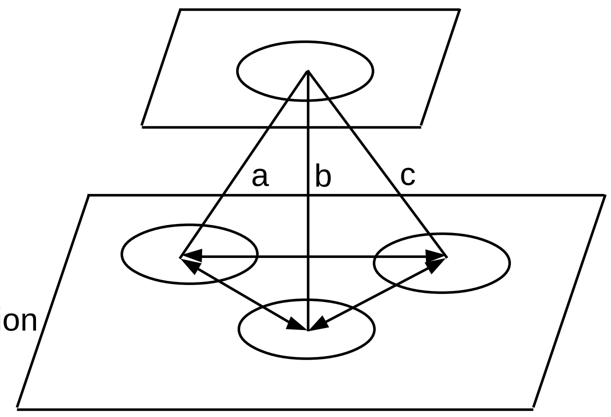
This article will cover the CICS routing models and the CICSPlex SM model for dynamic workload management of work within CICS. It also explains the mechanics of affinity management and abend avoidance provided by CICSPlex SM.

THE CICS MODEL

The routing facilities provided by CICS are outlined in Figure 1. Routing facilities have been enhanced across various releases, and one can now route dynamic transactions and programs,

DFHDYPDS
- function code
- tioaCopy
- Comarea
- Userarea
- sysid
- netname
- remtranid
- program
- queue
- mropri
- invocationCount
- invokeOnTermination
- routeErrorCode
- dtrtranIndicator
- dtrtranRefect
- MVSServiceClass Token
- AbendCode
- abendCopntinueIndicator
- CBTS data (process/activity)
- component(RT/SH)
- requesttype

DFHSIT
- DTRPGM=myexit
- DSRTPGM=myexit
- DTRTRAN=mytran



CICS 3.3
- transaction routing

CICS 4.1
- DTRTRAN
- Static ATI
- abend

CICS 5.3
- DPL
- CICS BTS
- START

CICS 6.1
- Enterprise Java Bear

Figure 1: The CICS model

START requests with or without data, START TERMID requests, CICS Business Application Services activities and processes, Enterprise Java Beans requests, and dynamic bridge requests.

The determination of the target is via exit invocation, passing the DFHDYPDS data area that contains information relevant to the routing request, and it also acts to return to CICS information on how to process the request. Additional information is contained in the CICS System Initialization Table (DFHSIT). The user exit invoked by CICS depends on the type of routing request.

CICS provides two routing models – dynamic transaction routing program model (DTRPGM), introduced in CICS 3.3; and distributed routing model (DSRTPGM), introduced in CICS TS 1.3.

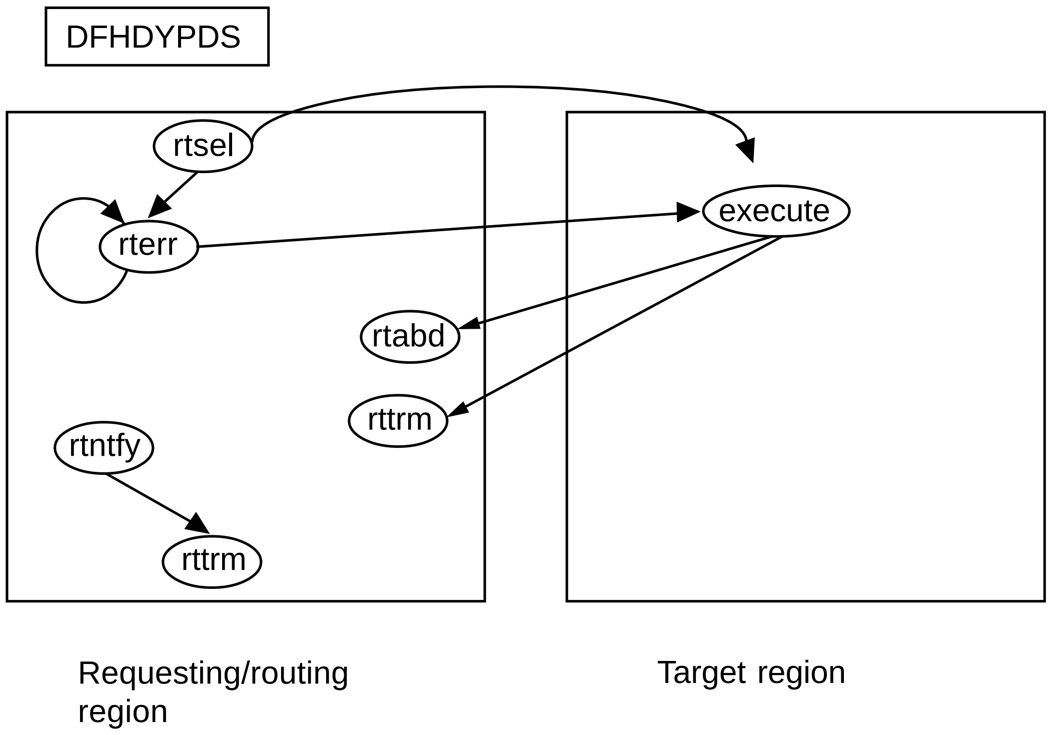


Figure 2: DTRPGM calls

DTRPGM MODEL

DTRPGM is the 'traditional' routing mode introduced in CICS TS 1.3. It applies to the routing of transactions, programs, START TERMID, and dynamic bridge requests.

CICS invokes routing for a specific request and the routing exit returns to CICS the APPLID or SYSID of the target region. CICS then routes the request, executes it in the target, and then reinvokes the exit in the router for termination processing.

The calls shown in Figure 2 are as follows:

- rtsel – route select.
- rterr – route error (unable to get to selected target; a retry to an alternative is allowed).
- rtabd – route abend (abended in the target region).
- rttrm – route terminate (successful completion).
- rtntfy – route notify (no selection possible. Notification so that the effect on load on the target can be established).

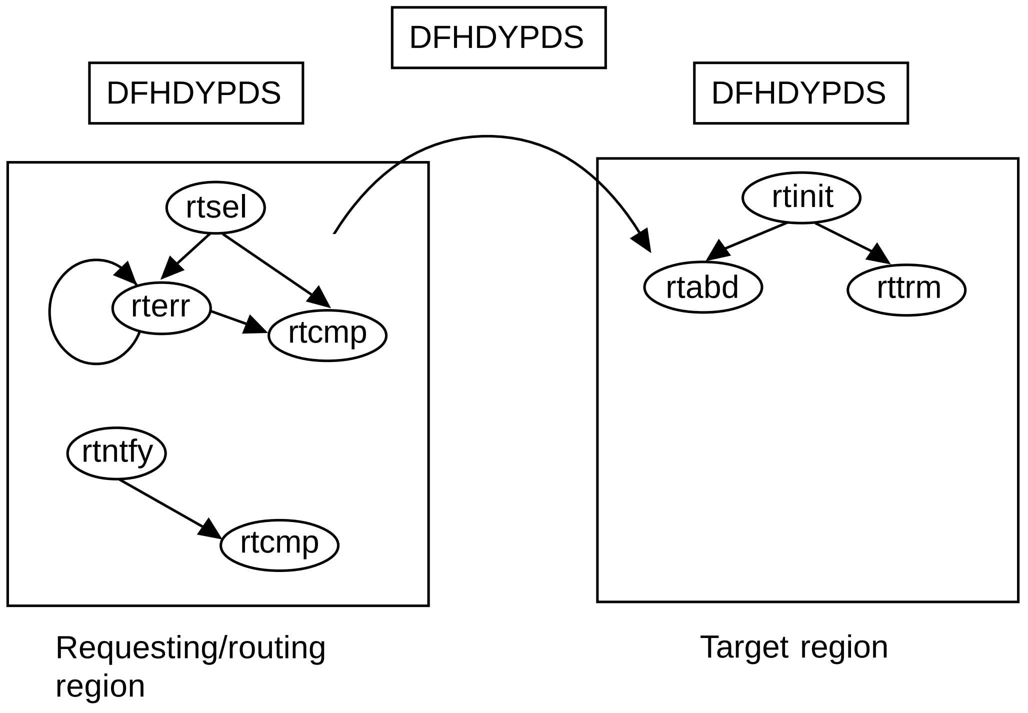


Figure 3: DSRTPGM calls

DSRTPGM MODEL

The DSRTPGM model was introduced in CICS TS 1.3. It now supports CBTS activities and processes, STARTs with or without data, and EJBs. The principal difference is that whilst the request

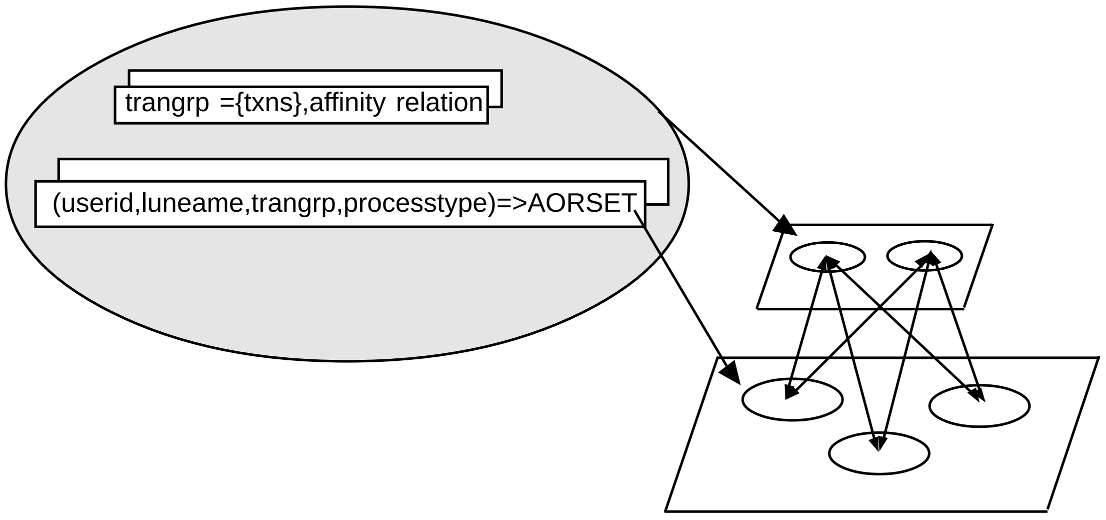

trangrp ={txns},affinity relation
(userid,luneame,trangrp,processtype)=>AORSET

Figure 4: CICSPlex SM Dynamic routing model

is initiated in the requestor, the flow of control never returns to the requestor. The calls are the same as in Figure 2 but with the following additional invocations specific to this model:

- rtcmp – route complete – the request has been successfully queued at the target system.
- rtinit – route initialize – about to dispatch this work item.

This is illustrated in Figure 3.

Note that although the calls are similar, the address spaces in which they are invoked are not necessarily. Furthermore, a copy of the user area in DFHDYPDS is copied over to the target region. Of course, we are talking roles here. For a local route, the routing and target region are the same physical region.

CICSPlex SM provides a version of the DSRTPGM and DTRPGM exits (EYU9XLOP) and associated administration and operations for workloads. This is described in Figure 4.

The administrative constructs to define the workload management criteria for balancing, separation, and affinity management are as follows.

TRANGRPS

TRANGRPS define the affinities that exist between transactions. They are also used to define applications for workload separation.

```
trangroupName = {tranid},affinity relation and lifetime
```

WLMDEFS

WLMDEFS specify workload separation criteria.

```
(trangroupName, Luname, Userid, processtype) -> CandidateSet
```

WLMGROUPS

WLMGROUPS provide a simple administrative construct.

WLMSPECS

WLMSPECS defines default routing criteria, the routing algorithm to be used, and whether abend avoidance is activated. The workload criteria are made known to the relevant regions by association of the WLMSPEC with those regions (typically via a system group).

Health data and RTA thresholding

When routing the transaction to a given set of CICS targets, it obviously makes sense to send that work to the regions most capable of executing that work. In RTA we saw SAM identify various conditions that might exist in the CICS regions which might make it less eligible to process work. The user can also specify an RTA event with which they can effect routing into a target. The full power of RTA can therefore be employed in the routing decision. These correspond to the 'Health factor' that is used by the balancing algorithms.

REALTIME DATA

Routing algorithms usually calculate a weight for each candidate system based on a set of weights derived from various criteria. CICSPlex SM takes account of various properties of the candidate set systems when deducing the target system. Basic health indicators from the Real Time Analysis component of CPSM are used (System Availability Monitoring events and any user defined RTA event).

These criteria are:

- Maxtask – the region is at maxtask.
- SOS – the region is short on storage.
- Dumping – the region is processing a transaction dump.
- Stalled – potential stalled situation (see next section).
- Down – the candidate region is shut down.
- Inactive – the candidate region is inactive. Inactive is a property of CPSM dynamic workload management. You can quiesce a target from dynamic routing, eg for applying maintenance.
- RTAevent – the user-specified RTA event is active (the severity is also taken into account).
- Linkdown – the link between router and target is not available.
- Lostcontact – the management software has lost contact with the target system, its state is unknown.
- Nolink – there isn't a link between the router and the target!
- Abendhealth – see abend avoidance.

Most of these are straightforward enough. Stalled, whilst intuitively obvious, is harder to define.

STALL INDICATORS (SAM)

The detection of a stalled state is based on SPI made available in CICS 4.1. Part of the heartbeat process is to build a table of active tasks versus suspend type and CPU used by utilizing the following SPI command:

```
  EXEC CICS INQUIRE TASK(TASK)
        RUNSTATUS(RUNSTATUS)
        SUSPENDTIME(SUSPENDTIME)
        SUSPENDTYPE(SUSPENDTYPE)
        RESNAME(RESNAME)
```

If, on successive inquires, no CPU has been used, the task must have remained in its previous state. After the specified number of intervals a stalled state is raised. These intervals are defined via EYUPARMS.

For convenience, the suspendtype is consolidated down to several classes.

EYUPARMS (COUNT OF STALLED TASKS, NUMBER OF TIMES TRUE)

Storage

```
STALLSTGTSK
STALLSTGCNT
     CDSA,ECDSA,ERDSA,ESDSA,EUDSA,RDSA,SDSA,
RDSA,SDSA,UDSA
```

DB control

```
STALLDBCTSK
STALLDBCCNT
     DBCTL
```

DLI

```
STALLDLITSK
STALLDLICNT
     DLI
```

Task

```
STALLTSKTSK
```

```
STALLTSKCNT
     EKCWAIT,KCCOMPAT
```

Enqueue

```
STALLENQTSK
STALLENQCNT
     KC_ENQ
```

Journal

```
STALLJNLTSK
STALLJNLCNT
     JCTAPE2,JASUBTAS,JCAVLECB,JCBUFFER,JRNL,
JCCLDONE,JCDETACH,JCFLBUFF,JCIOBLOK,
JCIOCOMP,JCJAGET,JCJAPUT,JCLASTBK,JCOPDONE
JCREADY,JCRQDONE,JCSWITCH
```

TD queue

```
STALLTDQTSK
STALLTDQCNT
               Any_MBCB,Any_MRCB,TDEPLOCK,TD_INIT,
MBCB_xxx,MRCB_xxx,TDIPLOCK
```

TS queue

```
STALLTSQTSK
STALLTSQCNT
     TSAUX,TSBUFFER,TSEXTEND,TSIO,TSOPEN4B,
TSQUEUE,TSSTRING,TSUT,TSWBUFFR
```

File

```
STALLFLETSK
STALLFLECNT
FCBFWAIT,FCDWWAIT,FCFSWAIT,FCIOWAIT,
FCPSWAIT,FCRBWAIT,FCSRSUSP,FCTISUSP,
FCXCWAIT
```

Interval

```
STALLITVTSK
STALLITVCNT
     ICGTWAIT,ICWAIT
```

Terminal

```
STALLTRMTSK
```

```
STALLTRMCNT
     ZCIOWAIT,ZCZGET,ZCZNAC
```

Session

```
STALLSESTSK
STALLSESCNT
     ALLOCATE,IRLINK
```

Dispatcher

```
STALLDSPTSK
STALLDSPCNT
     DS_HELD
```

Program

```
STALLPGMTSK
STALLPGMCNT
     PROGRAM
```

XRF

```
STALLXRFTSK
STALLXRFCNT
     XRGETMSG,XRPUTMSG
```

Lock

```
STALLLCKTSK
STALLLCKCNT
     XM_HELD
```

DB2

```
STALLDB2TSK
STALLDB2CNT
     CDB2RDYQ,CDB2TCB,DB2_INIT,DB2
```

AFFINITY MANAGEMENT

Affinity creation and deletion

Affinities are created when a transaction is routed for which an affinity relation is defined. On route selection, the tranid is resolved to a trangroup, and if an affinity is defined, but does not yet exist, then an affinity element is created. Note that the affinity defined for a transaction is a declaration of future intent to create an affinity. Once this transaction is routed, it must continue to be routed to the same target until the affinity is destroyed. Depending on the type of affinity, the affinity element may need to be ‘broadcast’ across the CICSplex. If this is required, the CMASs involved in the workload must be active.

Affinity deletion depends upon the lifetime of the affinity. Some affinities terminate when the transaction terminates (pconv terminates on EXEC CICS RETURN), some when they return to an overseer/menu transaction (DELIMIT). Others last until the AOR terminates (SYSTEM), others until the user signs off or logs off (detected via SIGNOFF and LOGOFF user exits). Others last beyond the lifetime of the CICS system for recoverable resources (PERM).

Transaction traces can exist within one transaction group or across transaction groups. Consider the following:

```
Definition
trangroup(A) = a,b,c
trangroup(B) = x,y,z
```

Both have affinity defined.

Runtime trace is across trangroups:

a = create affinity (say CICS1)

b = honour affinity to CICS1

x = create affinity (say CICS5)

y = honour affinity to CICS5

c = honour affinity to CICS1.

Some affinities are worse than others. On affinity creation, depending on the relation and lifetime, affinities can require a CICSplex-wide lock to be obtained (ie affinity elements need broadcasting across the CMAS network). If any CMAS in the network is down, the affinity element cannot be created and the transaction cannot be executed. If the affinity element already exists, it can execute in the absence of the CMAS. Other forms of affinity require the same incarnation of the AOR to be available (pconv, logon, signon, delimit). These are illustrated in Figure 5.

Broadcast	Permanent	System	Logon	Signon	Delimit	Pconv	activity	process
Global	Yes	Yes	N/A	N/A	N/A	N/A	N/A	N/A
Luname	Yes	Yes	No	N/A	No	No	N/A	N/A

Figure 5: Affinities

AOR incarnations

Consider a pseudo-conversational transaction. A request in the TOR routes to AOR1. In AOR1, state data is created. Whilst the end user is contemplating what to do, the AOR shuts down and restarts. Having finished his contemplations, the end user enters some data and hits *Enter*. What should happen now? If we route to the AOR (it is, after all, active) the state data could be gone. If we route to another, there will be no state data. The only option is to reject the routing request and let the operator sort out the mess (and delete the affinity element).

ABEND AVOIDANCE

When dynamically routing to a set of CICS regions, it is possible that the transaction could abend in some regions but not others (maybe owing to database accessibility). In such circumstances, you may want to preferentially route to the successful regions. CICSPlex SM provides such a capability via its abend avoidance mechanism.

When a transaction abends, an abend probability is associated

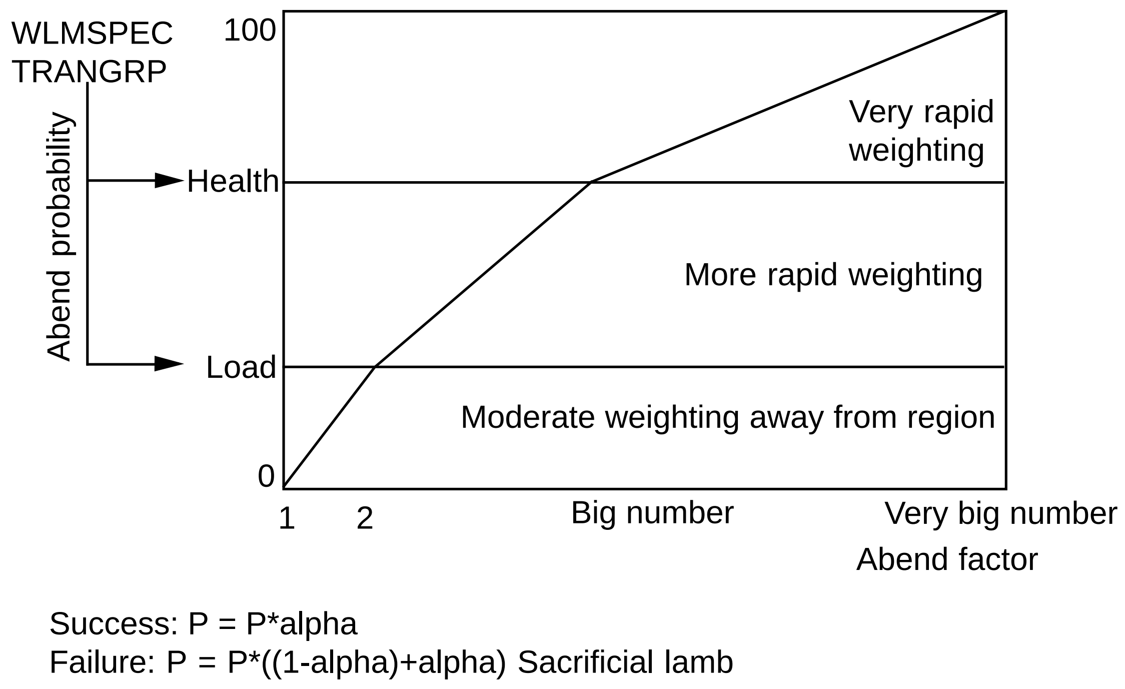


Figure 6: Abend probability graph

with the transaction (or transaction group if affinity exists). This abend probability is used to calculate an abend factor for use in the balancing algorithm. The initial abend probability is approximately 100%.

The calculation is controlled by user specification of two values – abendHealth and abendLoad (these are specified in WLMSPEC and TRANGRP). These two values allow a graph of abend factor to abend probability to be generated. From this graph, an abend probability can be converted into an abend factor – see Figure 6.

Since the initial abend probability is 100%, the abend factor is very large, and so routing to that CICS region is very unlikely.

So far we have biased routing away from the region. How do we decide when to try the region again?

Basically what happens is that we simulate successful completions of the failed transaction and recalculate the abend probability (and associated abend factor). At some point, the abend factor will have become so low that we choose the failing region to route a real request (the so-called sacrificial lamb). If the transaction executes successfully, the probability continues to reduce, and routing is re-established to *that* region without any operator intervention. If it fails, the probability increases and routing is biased away from that region again, and we repeat the process.

The probabilities are calculated as follows (0>alpha>1)

$$\text{AbendProbability}_{\text{Success}} = \text{AbendProbability}_{\text{Previous}} * \text{alpha}$$

$$\text{AbendProbability}_{\text{Failure}} = \text{AbendProbability}_{\text{Previous}} * ((1-\text{alpha}) + \text{alpha})$$

NEXT ARTICLE

In the next article we will look at the balancing algorithms that CICSPlex SM uses.

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

QMF goes to the Web

INTRODUCTION

There are a lot of ways to access DB2 from the Web – Net.Data, WAS, etc. But new interfaces means creating new programs. If you have thousands of QMF procedures, it could be interesting to put them directly on the Web without (too much) modification and without spending time and money on writing new programs.

THE PRINCIPLE

As shown in Figure 1, when a user types a URL (1), the HTTP request handler starts a transaction (2) running the QMF callable interface. This transaction reads the parameters from the request, initializes the QMF session, executes a procedure (3), and exports the result in HTML in a temporary storage (TS) queue (4). The program then reads the TS queue and puts it in the DFHCOMMAREA (5). The HTTP request handler sends it back to the user (6).

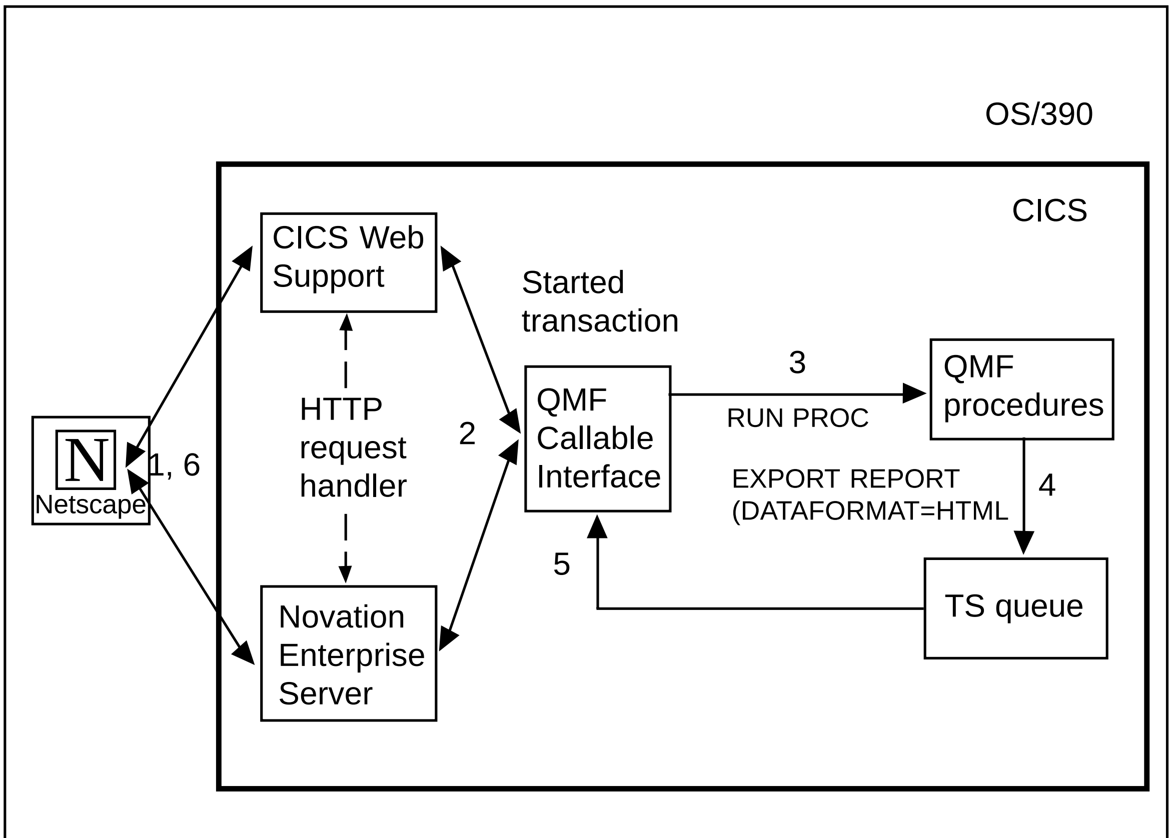


Figure 1: How it works

The URL http://myhost.mydomain:10004/cics/CWBA/PGMX doesn't look very friendly to the end user. That's why it's interesting to build an HTML page with an array of links corresponding to each report. It's very easy to generate such a page with a QMF report executing a query on the QMF object table, Q.OBJECT_DATA. A problem arises when you want to pass parameters to the QMF reports. In this case, for each report, you have to build an intermediary HTML page requesting those parameters. This could be very tedious if you have thousands of QMF queries. The solution is to write a batch program that reads the QMF object table, find the parameters delimited by an '&' in the queries and build the corresponding HTML pages.

As you can see in Figure 1, you may choose the HTTP request handler – either using CICS Web support or using a Web server like Novation Enterprise Server from GT Software. We have chosen the second solution because:

- It's a true Web server running in CICS.
- By using the HFS file system, you can organize your Web site(s) in a directory structure, like any other Web server running on other platforms.
- It supports scripting languages like LUA.

More information is available from http://www.gtsoftware.com.

Whatever you choose, your HTTP request handler must start a CICS transaction with a correct DB2Entry or DB2 RCT definition, pointing to the QMF plan.

IN PRACTICE

This started transaction runs a program that does the following.

Decodes the HTTP request

The HTTP request received by the program looks like this:

```
REQUEST=POST /CGI-BIN/GTICGLNK/EXAMPLE1 HTTP/1.Ø REQUEST_METHOD=POST
DOCUMENT_URI=/CGI-BIN/GTICGLNK/EXAMPLE1 REQUEST_URL=/CGI-BIN/GTICGLNK/
EXAMPLE1 DOCUMENT_NAME=EXAMPLE1 REQUEST_PROTOCOL=HTTP/1.Ø
HTTP_REFERER=http://testy.bogus.com:8Ø8Ø/QuickStartCGI.htm
HTTP_CONNECTION=Keep-Alive HTTP_USER_AGENT=Mozilla/4.7 [en] (Win95; U)
HTTP_HOST=testy.bogus.com:8Ø8Ø HTTP_ACCEPT=image/gif, image/x-xbitmap,
image/jpeg, image/pjpeg, image/png, */* HTTP_ACCEPT_ENCODING=gzip
HTTP_ACCEPT_LANGUAGE=en HTTP_ACCEPT_CHARSET=iso-8859-1,*,UTF8
```

```
CONTENT_TYPE=APPLICATION/X-WWW-FORM-URLENCODED CONTENT_LENGTH=22
HTTP_FIELD1=something typed POST_STRING=field1=something typed
SERVER_NAME=Novation TM SERVER_PROTOCOL=HTTP/1.Ø
SERVER_SOFTWARE=Novation TM Enterprise Web Server for CICS
GATEWAY_INTERFACE=CGI/1.Ø PATH=gt:/www/;gt:/lua/ PATH_INFO=/CGI-BIN/
GTICGLNK/EXAMPLE1 PATH_TRANSLATED=GT:/WWW/CGI-BIN/GTICGLNK/EXAMPLE1
LOG_LISTENER=CSMT LOG_SUCCESS=CSMT LOG_ERRORS=CSMT HOST_CDPG=37
REMOTE_CDPG=819 SERVER_ROOT=GT:/WWW/ SECURE_SOCKET=Ø
SERVER_ADDR=1Ø.8.17.1ØØ SERVER_HOST=1Ø.8.17.1ØØ SERVER_PORT=8Ø8Ø
REMOTE_ADDR=1Ø.8.17.16 REMOTE_HOST=1Ø.8.17.16 REMOTE_PORT=1623
DATE_LOCAL=Tuesday, 16-Nov-99 13:42:54 DATE_GMT=Tuesday, 16-Nov-99
18:42:54 SERVICE_NAME=link TPL_BLOCK=Ø62e9914
HTTP_ASPOID=EXAMPLE1_1261296941 HTTP_ASPOID_INDEX=1 HTTP_ASPSOCKET=-1
```

The program must extract the parameters to pass to the QMF procedure from the HTTP request, using the Novation Parser or the EXEC CICS WEB READ/STARTBROWSE/READNEXT/ ENDBROWSE FORMFIELD statements for the CICS Web support. You can even pass the name of the QMF procedure to run as a parameter of the request.

Issues the following commands through the QMF Callable interface

For more information on the Callable interface, see the *Developing QMF Applications Guide*.

START QUERY

The START QUERY statement initializes the QMF session and sets some command keywords. In our case, the most important are:

- DSQALANG – E – determines the presiding language for the session you are starting. E specifies English.
- DSQSBSTG – 50000 – tells QMF how many bytes of storage to use for report generation.
- DSQSMODE – B – tells QMF which mode you want to work in. B specifies batch mode.
- DSQSSPQN – ? – specifies the name of the CICS temporary storage queue that is used for QMF spill data. It must be a unique name.

SET GLOBAL

Having extracted the parameters and their values from the HTTP request, it's time now to set them prior to the execution of your QMF procedure. Be aware that numeric variables must be set apart from alphanumeric variables. For more information, see the *SET GLOBAL* paragraph in the *Developing QMF Applications Guide*.

RUN PROC

You must specify a full QMF procedure name, ie OWNER.PROC_NAME, and issue the RUN PROC command.

Security

Because the started transaction isn't associated with a terminal, no signon is possible, although you may perform an EXEC CICS VERIFY PASSWORD. (See *CICS Transaction Server for OS/390 V…R… CICS Application Programming Reference* for the difference between EXEC CICS VERIFY PASSWORD and EXEC CICS SIGNON.) This means that your QMF procedures must be shared and that the userid of the CICS region must have at least SELECT authority on *all* the tables of your query.

EXPORT REPORT

To generate the HTML page from your report, use the following command:

```
EXPORT REPORT TO XXX (QUEUEUTYPE=TS CONFIRM=NO DATAFORMAT=HTML
```

where XXX is the name of a TS queue (don't use the same name for this queue and for the DSQSSPQN value).

The HTML generated by QMF is very poor, just enough to produce a black and white page. If you want to enhance the HTML page of your report, you may:

- Copy the actual form to 'HTML_form'.
- Modify the form by inserting all the HTML tags needed. The original form remains unchanged.

- Issue a DISPLAY FORM HTML_form command in the Callable Interface after the RUN PROC. This HTML_form is used only when displaying QMF reports (mainframe users won't ever see it!) on the Web and there is no need to modify the existing QMF procedure or query.
- Issue a PRINT REPORT (QUEUENAME=XXX QUEUEUTYPE=TS CONFIRM=NO instead of the EXPORT REPORT.

END QUERY

The END QUERY statement ends the QMF session using the callable interface.

Reads the TS queue containing the HTML

Using a simple loop, it reads the TS queue records and concatenates them in the DFHCOMMAREA. It's a good idea to put a X'15' (line feed) at the end of each TS record, so that the HTML source can more easily be viewed on the PC.

CHARTS AND GIFS

Although you may create and export chart objects in QMF, it isn't possible to create an online GIF. Indeed, QMF can export only those objects to a GDDM file, and the ADMUGIF conversion utility between the GDDM picture format (GDF) and GIF is not supported in CICS. See *GDDM Base Application Programming Reference* for more details.

THE 32KB BARRIER

The CICS maximum DFHCOMMAREA length is 32,767 bytes and the HTML report generated by QMF must enforce this rule. But CICS Web Support and Novation Enterprise Server can run in pointer mode – instead of passing a COMMAREA, you may pass a pointer to it using the template manager. For more details, see *CICS Internet Guide* or the Novation documentation.

CONCLUSIONS

This way of accessing DB2 data might not be the best (dynamic SQL, black and white HTML pages …), but it enables you to put existing reports on the Web without any coding modification. This is one of the main advantages of this solution.

Pierre Delaunoy
Director
Ministère de la Communauté Française (Belgium)

CICS questions and answers

Q We have started to use CICS Transaction Gateway/390 and therefore the EXCI interface. We were wondering whether it is possible to workload balance the requests across multiple CICS regions?

A The DFHXCURM User Replaceable Module allows you to do this. You could code an alias for a group of regions (eci_system_name=alias) on the ECI/EXCI request and use the URM to change this alias to a specific region. You can code CICSPlex/SM APIs in this URM and use CPSM to manage the alias look-up and workload management. Be aware that this module is called only for a new connection. CTG/390 doesn't free the existing ones, it re-uses them. So in a low activity system the first connection for an alias will get the majority of the work. If RRS is in use then this exit must also ensure that the selected region is on the same LPAR.

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.

CICS news

QED Business Systems has announced FireXML for CICS, which allows mainframe applications to become Web services. Running solely in CICS on z/OS, OS/390, and VSE mainframes, FireXML for CICS is the middleware that interfaces between the mainframe world of COBOL, PL/I, C and Assembler, and either the Web world of Web services or the outside world of remote systems communicating via messaging protocols like MQ Series.

FireXML for CICS allows the Web services model to incorporate the enterprise business logic contained in mainframe applications, resulting in cost savings in development, testing, infrastructure, deployment, and training. Using FireXML for CICS, browser-based user interfaces written in standard Web server languages, such as ASP, VB, Perl, and Java provide the front-end presentation logic to mainframe business logic.

For further information contact:
QED Business Systems, QED House, 430 Bath Road, Slough, Berks SL1 6BB, UK.
Tel: (01628) 660025.
URL: http://www.qedbusiness.com/firexml.htm.

* * *

Seagull has announced Transidiom 3.0, a module in its LegaSuite. It now includes a CICS Connector that enables integration from any J2EE or .NET application server to non-screen-based CICS transactions.

It provides direct access to CICS transactions without dependency on conversational access through the 3720 data stream or map files. The non-invasive connector automates development of programmatic interfaces and Web services.

Specifically, it lets users integrate CICS or mainframe-based client/server applications into Java and .NET environments, enabling them to deliver CICS functions with XML, COM, Java, SOAP, or MQSeries interfaces.

For further information contact:
Seagull, 3340 Peachtree Road NE, Atlanta, GA 30326, USA.
Tel: (404) 760 1560.
URL: http://www.seagullsw.com/products/transidiom.html.

* * *

IBM has launched CICS Performance Monitor for z/OS V1.1, a comprehensive on-line performance monitoring and management tool for CICS Transaction Server for OS/390 V1.3 and CICS Transaction Server for z/OS V2.2. It provides facilities for the display and management of user-defined thresholds and for the display of detailed real-time performance data. Information is displayed at a Windows workstation.

CICS PM provides Single System Image (SSI) access to the resources in all CICS regions, allows early detection and correction of performance problems, provides intuitive cross-system navigation for fast problem determination, is built on standard APIs and proven CICS TS system management technologies, and is said to be easy to install, set up, learn, and use.

CICS Performance Monitor for z/OS Version 1.1 (CICS PM) provides real-time performance monitoring and management facilities for CICS Transaction Server on OS/390 and z/OS.

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

xephon