196

CICS update

March 2002

In this issue

3 CICS log record formatting

10 Sending PCL commands to a printer

20 Maintenance of CICS DB2 entries and transactions

42 Exit-enabling program

43 CICS questions and answers

44 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 1998 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 record formatting

BACKGROUND

CICS log records are produced by CICS Transaction Server in response to certain events taking place in the system. They are generated for system-recovery activity (such as automatic logging of recoverable changes made to CICS-managed resources), and for other general events (such as recording audit trails for terminal I/O events or for BTS processes and their constituent activities, for forward recovery logging and autojournalling of CICS file control operations, or for explicit writes to user journals). Their destinations can be the CICS system log (DFHLOG and DFHSHUNT logstreams) or general logs defined for user-purposes (eg DFHJ02). Some general logs are used by CICS for system-related activity, such as the logging of CICS file control forward recovery log records.

The format of the log records varies between system log and general log destinations. Similarly, log record formats used in CICS Transaction Server are different from those used for the equivalent logging of events in CICS/ESA 4.1.0 and earlier CICS/ESA versions.

The area of log record formatting, conversion between the CICS/ESA 4.1.0 and CICS Transaction Server record formats, and the choices and options available when reading logstreams, has led to some confusion in the user community. This article describes the different approaches available in handling the log data written by CICS, and the effects of specifying various options on the batch jobs used to process CICS logs.

CICS/ESA 4.1.0 LOG FORMAT OVERVIEW

CICS/ESA 4.1.0 used BSAM to support logging to system and user journals. Typically, customers would define dual dataset support for each journal being used by the system. These would be the A and B extents; for example, DFHJ02A and DFHJ02B were the two dataset extents for user journal DFHJ02. CICS journal control processing provided buffering of the log records within in-core memory areas, and initiated I/O events to BSAM as appropriate (when a buffer was too full to accommodate a new record to be written, or if a synchronized journal control command was being processed, for example).

The CICS/ESA 4.1.0 journal dataset extents were defined to BSAM as Undefined (U) Record Format. RECFM=U was used to give CICS control of when I/O events to DASD were made. It also allowed CICS control over the record positioning within each log block. For example, CICS could then ensure that a journal Label Record was present at the start of each block. Despite the dataset extents themselves being defined as RECFM=U, the records and blocks were written in Variable Blocked (VB) format. CICS achieved the appearance of its journal data being written in RECFM=VB format by ensuring that LLBB fields were added to the start of each record, and at the start of the log block being constructed in the in-core buffer.

Typically, when a dataset extent filled up this would be archived prior to reuse, whilst logging continued on the new extent just switched to. CICS/ESA 4.1.0 included automatic archiving support, to submit a batch job when a journal extent switch occurred, which then archived this dataset and marked it ready for reuse. In this way, the dataset extents were written to, archived, and reused as CICS filled them and switched from one to the other, and back again.

DFHJUP was provided as the batch utility program for journal record formatting. It allowed the log records to be read from the target journal, filtered, and then printed or copied to another dataset. If the DD statement for the journal dataset in the DFHJUP job were to specify a RECFM=U (or defaulted to such), each log block would be returned as an undefined structure. If RECFM=VB were specified as an override to the DCB definition for the target dataset, BSAM would deblock the records into individual entries, as per their preceding LLBB values.

CICS TRANSACTION SERVER LOG FORMAT OVERVIEW

CICS Transaction Server replaced DFHJCP and the journal control component of CICS with a restructured, object-oriented Domain called the CICS Log Manager component. This writes CICS log records to blocks of data managed by the MVS System Logger subsystem (IXGLOGR). MVS Logger holds its data on logstreams, which can reside on structures within a Coupling Facility, or on staging datasets for DASD-only logging support. There is no longer the concept of an extent switch, since the logstream appears to be continuous. Therefore there is no convenient point to archive data as there was for CICS/ESA 4.1.0. Typically logstreams are copied as they are being written to.

CICS Log Manager writes log records in a different format from those used in CICS/ESA 4.1.0. The record headers are structured differently, and the contents and lengths of equivalent records between CICS/ESA 4.1.0 and CICS Transaction Server are not the same. Since CICS Transaction Server has no need to recover task or system failures by making use of the old-style log data, there is no need for CICS run-time code to be sensitive to or aware of the CICS/ESA 4.1.0 format of log records; such records are not encountered by CICS Transaction Server systems.

Conversely, user journal data (such as audit trails, terminal I/O records, etc) is of interest to other batch or online programs that run in conjunction with a CICS Transaction Server environment. These programs may have been converted from their CICS/ESA 4.1.0 counterparts, and be ‘logstream aware’; that is, they have been modified to know that their target ‘dataset’ is in fact a logstream, and so they issue the appropriate MVS System Logger API calls to access the journal records there. If so, they will be able to format and handle log records written in this new style. For those vendor packages and user programs that still handle CICS/ESA 4.1.0 format records, however, there needs to be a transparent way of giving them access to a logstream and of returning the log data into the older style of records, both without the program being aware that it is not accessing a BSAM dataset.

SUBSYS OPTIONS

CICS provides a Subsystem Interface (SSI) exit to allow batch programs to access CICS-managed logstreams. The exit is called DFHLGCNV; it (and its partner module DFHGTCNV) resides in SDFHLINK within the MVS Linklist. The batch program JCL can specify a DD card for SUBSYS options. This names the exit program to be invoked when access method service requests are made against the target dataset. If the batch program were to issue Open and Get macros against what it believed was a CICS journal dataset managed by BSAM, these would then be intercepted by the CICS-supplied exit and dynamically converted into the appropriate MVS System Logger API calls. The record data would then be passed back from the exit to the application program. In this way, the fact that the target dataset was in fact a logstream is shielded from the application; its logic does not need to be changed to directly invoke the MVS System Logger API.

This approach solves one of the requirements for existing applications. In order to have them recognize log records in CICS/ESA 4.1.0 format, the SUBSYS option provides support for formatting options. These are COMPAT41 and COMPAT41V. If specified, they return the log data in CICS/ESA 4.1.0 format. Note: these options are only available for general logs; they do not convert CICS system log records from DFHLOG or DFHSHUNT into their equivalent syncpoint records on CICS/ESA 4.1.0. Note also that COMPAT41V was provided for COBOL programs which expect the data in RECFM=VB format. By default, logstreams are deemed to be Undefined (RECFM=U). With RECFM=VB, COBOL would present the record to an application, having removed the preceding LLBB. Since a logstream is deemed Undefined, the LLBB would not be removed by COBOL, and so the application receives a record with an unexpected 4 byte LLBB prefix. To avoid this, COMPAT41V can therefore be specified to make the DFHLGCNV SSI exit remove the LLBB instead, and thereby maintain compatibility for existing COBOL applications that expect to read back log data in CICS/ESA 4.1.0 format.

DFHJUP OPTION NEWDCB

The CICS Transaction Server version of DFHJUP supports a new option on the COPY command. This is NEWDCB – if used it overrides the input dataset’s DCB characteristics when determining how to set up the output dataset’s DCB. This is useful if the output destination is to store records copied in CICS/ESA 4.1.0 format. Since a logstream has a RECFM=U, such records would be copied one per block on the output dataset. By specifying NEWDCB, and supplying DCB characteristics via JCL on the DD statement for the target dataset, output dataset space can be more optimally managed by copying the records in RECFM=VB format rather than RECFM=U.

It is possible that customers may wish to copy new-style CICS log records to an output dataset, and utilize the NEWDCB option to allow the log records to be blocked by BSAM. This would make optimal use of space on the copy destination. There is a problem with this approach, however. Unlike CICS/ESA 4.1.0 log records (or those returned from CICS Transaction Server by means of the COMPAT41 option) new-style log records do not start with an LLBB field. Without such a fullword, they cannot be passed to BSAM to be blocked to an output destination with a DCB attribute of RECFM=VB.

One solution to this is to use the DFHJUP exit option. By specifying 'EXITR=' on the DFHJUP OPTION statement, a user-supplied exit program will be invoked for every record read back from the input logstream. The exit can analyse the record and modify it if required; it can also instruct DFHJUP to ignore the record and proceed to the next one on the logstream, without copying the record to the output dataset.

Such a sample exit program is given in Example 1 below. It is intended as a guide for record manipulation; users would be expected to modify it to serve their own purposes as appropriate. The example exit rejects Block Header Records (which are CICS-internal information, not relevant to user data on the journal); for other record types it modifies the initial fullword.

```
***********************************************************************
* Name:      JUPEXIT                                                  *
* Author:    Andy Wright                                              *
* Date:      29th October 2ØØ1                                        *
* Copyright: IBM Corporation 2ØØ1                                     *
* Purpose:   Provide a sample exit routine for DFHJUP, to parse log   *
*            records and reject block header records. Adjust remaining*
*            records to have an LLBB at their start. This means that  *
*            the remaining CICS TS log records returned by DFHJUP are *
*            in RECFM=VB format, and so are eligible for blocking by  *
*            BSAM if COPY is used with NEWDCB for the output dataset. *
***********************************************************************
         DFHREGS                      Establish register equates
JUPEXIT  CSECT
         STM   R14,R12,12(R13)        Save the registers on entry
         BASR  R3,Ø                   Establish base register
         USING *,R3                   Tell the assembler
         ICM   R4,15,Ø(R1)            Address record
         BZ    EOF                    If plist empty then eof
         CLC   Ø(Ø,R4),=CL4'>DFH'     Test for block hdr record
```

```
BE    EXITREJ                If so, reject record
L     R5,Ø(,R4)              Pick up record length
SLL   R5,16                  Convert to LLBB format
ST    R5,Ø(,R4)              Store back at start of record          *
WTO   'DFHJUP EXIT RAN'      Diagnostic message
B     EXIT                   Leave the program
EOF      DS    ØH
*        WTO   'EOF ENCOUNTERED'      Diagnostic message
EXIT     DS    ØH
LM    R14,R12,12(R13)        Restore the registers on exit
SR    R15,R15                Set a good return code
BR    R14                    Return to DFHJUP
EXITREJ  DS    ØH
LM    R14,R12,12(R13)        Restore the registers on exit
LA    R15,1                  Set a bad rc (no R3 base anymore)
BR    R14                    Return to DFHJUP
DROP  R3                     Tell the assembler
LTORG                        Define the literal pool
END
```

Note: it does not cater for the possibility of a record exceeding 32,767 bytes in length. One solution to this would be to cap such record lengths, and so avoid BSAM failures when writing long records to the RECFM=VB output dataset.

COPYING OPTIONS

DFHJUP can be used to copy or print logstream contents, and options such as COMPAT41 can return the log data in old-style format as required. Since this old format was preceded by an LLBB, the log data could be copied or printed in either RECFM=U or RECVM=VB formats.

Log data written in new-style CICS Transaction Server format is not preceded by an LLBB value. As such, it is manipulated at the record level; this is because the SSI exit returns individual records to the batch program reading the logstream, and these are then treated as RECFM=U formatted records if copied to an output dataset.

The CICS *Operations and Utilities Guide* discusses the use of DFHJUP in more detail.

Another method of copying or printing back blocks of raw log data is to use the IDCAMS utility, targeted at the Offload datasets for the logstream. These datasets will be retained until the logstream data is physically deleted. Deletion can occur in several ways. Explicit deletion can take place via an IXGDELET command issued from a logstream-aware program or by a DELETE option on the SUBSYS card, eg a DFHJUP or IEFBR14 job. This will mark the log data as logically deleted. The MVS System Logger will retain the data until a retention period has elapsed. This is specified by the RETPD option on the logstream definition. Implicit deletion can occur when the logstream is defined to use AUTODELETE(YES) and the retention period has been reached.

IDCAMS prints or copies of the logstream Offload datasets are raw unformatted 4KB blocks of data. However, they do provide a quick snapshot of log record contents. This may be useful when reviewing user journal log data used for audit trail purposes, for example.

The IEBGENER utility can also be used against a logstream if so required. If the job provides a SUBSYS DD statement for the logstream, IEBGENER will process it as if it were a sequential dataset. However, this will also reflect the inefficiency of the output dataset being one record per block, because of the logstream’s RECFM=U format mentioned above.

Be wary when examining a RECFM=VB dataset, such as a copy of a logstream that was taken with DFHJUP specifying COMPAT41 on the SUBSYS card and NEWDCB on the OPTION statement. Utilities such as IDCAMS, or online methods such as browsing the sequential dataset under ISPF on TSO, will deblock the dataset before returning the records. This is because it is defined as RECFM=VB, and hence the software is trying to shield the fact that records within the dataset were blocked by the access method. To see the data in its true format on the dataset, you need to override the dataset attribute by specifying a DCB on the JCL of the job, with RECFM=U. This will ensure that the data is returned in blocked format, rather that unblocked to the individual record level.

As with the use of DFHJUP and other offline utilities, security considerations for dataset access should be reviewed to ensure that only the correct personnel have read access to user data.

Details of the structure and content of CICS Transaction Server ‘new-style' log record formats for general logs can be found in the CICS *Customization Guide*, part 5 *CICS journaling, monitoring, and statistics*. This publication also describes the corresponding formats of 'old-style' log records formatted using the COMPAT41 SSI option.

I hope that this article has helped clarify the way in which CICS log records are written and formatted by offline utilities such as DFHJUP. Readers wishing to discuss this article further may e-mail me at andy_wright@uk.ibm.com.

Andy Wright
CICS Change Team (UK)

Sending PCL commands to a printer

We have a laser printer that we often use under CICS for various purposes: program testing, printing source code, CICS output, and so on. The printer uses A4 paper and is connected to VTAM by an AX-CORBA unit. Since we use the printer for so many purposes, it is not always easy to have it properly configured in terms of font size, line spacing, margins, sheet orientation, and so on, since many of the things that we print are just plain text without any embedded printer configuration.

For that reason, I created an application to send it a set of PCL commands, thus programming the printer for whatever we send it afterwards.

This application consists of two programs, PCLP001 and PCLP002, and two associated transactions, respectively PCL1 and PCL2, and also a BMS map, PCLS001.

Transaction PCL1 presents you with the following screen:

```
PPPPPP CCCCCC LL    CCCCCC      Send PCL commands to a printer
PP  PP CC     LL    CC
PPPPP  CC     LL    CC
PP     CC     LL    CC
PP     CCCCCC LLLLL CCCCCC      Printer....: PR53

  Font............: 1            1-Courier  2- LetterGothic
  Pitch...........: 11 . ØØ     Ø1.ØØ to 3Ø.ØØ
```

```
        Upper margin.....: 100        1 to 999 decipoints
        Left margin......: 300        1 to 999 decipoints
        Orientation......: 0          0-Vertical 1-Horizontal
        Vertical compress: 07         1 (compressed) to 99 (expanded)


  Pre-set: F4 80 columns   F5 132 columns   ENTER Send command   F3 Exit
```

Here you can configure several items, either by introducing the appropriate values or by loading one of the two pre-configurations (the ones we use most) by pressing PF4 or PF5. Just pressing one of these PFs does not actually send anything to the printer, it simply presents some pre-set values on the screen. These two configurations are appropriate to print 72 lines per page vertically with 80 columns or 132 columns of text, and they suit most of our needs. Normally, we only switch between 80 and 132 columns, but occasionally we also need to print in landscape, for which we must also adjust the margins and line spacing. To send the commands to the printer, just press *Enter*. The transaction exits and a message appears on your screen informing you that the command has been sent.

Note that you won't make the printer do anything just by sending this command string. Afterwards, you must send it some text to print, either by CICS or by other means, so that you actually see the result of the formatting orders. If you are not satisfied, try adjusting the values until they suit your needs.

PCLP001 COBOL SOURCE

```
        IDENTIFICATION DIVISION.
        PROGRAM-ID. PCLP001.
        ENVIRONMENT DIVISION.
        DATA DIVISION.
       *==============================================================*
        WORKING-STORAGE SECTION.
       *=======================*
        77   TRANS-1             PIC X(4)       VALUE 'PCL1'.
        77   TRANS-2             PIC X(4)       VALUE 'PCL2'.
        77   X                   PIC S9(8) COMP VALUE +0.
        77   CONFIG-STRING-LENG  PIC S9(4) COMP VALUE +71.
        77   MSG-LENG            PIC S9(4) COMP VALUE +40.
        77   MSG1                PIC X(40)
             VALUE 'Configuration sent to printer'.
        77   MSG2                PIC X(40)
             VALUE 'Printer not acquired'.
        77  FONTYP1-STRING  PIC X(19) VALUE '(s0p11.00h0s0b4099T'.
        77  FONTYP2-STRING  PIC X(19) VALUE '(s0p18.00h0s3b4102T'.
*
 Ø1  SLINE-INS   PIC S9(8) COMP VALUE +73.
 Ø1  SLINE-ACQ   PIC S9(8) COMP VALUE +69.
 Ø1  SLINE-CRE   PIC S9(8) COMP VALUE +67.
 Ø1  PRT-STATUS  PIC S9(8) COMP.
 Ø1  IMPRESSORA  PIC X(4).
*
 Ø1  CONFIG-STRING.
     Ø2    INIT-STRING    PIC X(Ø7)  VALUE X'5Ø5Ø6F6F6C6CD7'.
     Ø2    FILLER         PIC X(Ø3)  VALUE X'6CF1C2'.
     Ø2    FONTYP-STRING.
       Ø3  FILLER         PIC X(Ø4).
       Ø3  PITCH1-STRING  PIC X(Ø2).
       Ø3  FILLER         PIC X(Ø1).
       Ø3  PITCH2-STRING  PIC X(Ø2).
       Ø3  FILLER         PIC X(1Ø).
     Ø2    FILLER         PIC X(Ø5)  VALUE X'6CF1C25Ø93'.
     Ø2    MARSUP-STRING  PIC X(Ø3).
     Ø2    FILLER         PIC X(Ø1)  VALUE 'Z'.
     Ø2    FILLER         PIC X(Ø5)  VALUE X'6CF1C25Ø93'.
     Ø2    MARLEF-STRING  PIC X(Ø3).
     Ø2    FILLER         PIC X(Ø1)  VALUE 'U'.
     Ø2    FILLER         PIC X(Ø5)  VALUE X'6CF1C25Ø93'.
     Ø2    ORIENT-STRING  PIC X(Ø1).
     Ø2    FILLER         PIC X(Ø1)  VALUE 'O'.
     Ø2    FILLER         PIC X(Ø5)  VALUE X'6CF1C25Ø93'.
     Ø2    COMPVE-STRING  PIC X(Ø3).
     Ø2    FILLER         PIC X(Ø1)  VALUE 'C'.
     Ø2    FINIS-STRING   PIC X(Ø7)  VALUE X'5Ø5Ø6F6FFØFØFØ'.
     Ø2    END-OF-MESSAGE PIC X(Ø1)  VALUE X'19'.
*
 Ø1  COMMAREA.
     Ø2 PCLSØØ1I.
*
           Ø5     FILLER           PIC   X(12).
           Ø5     IMPRESL   COMP   PIC   S9(4).
           Ø5     IMPRESF          PIC   X(Ø1).
           Ø5     IMPRESI          PIC   X(Ø4).
           Ø5     FONTYPL   COMP   PIC   S9(4).
           Ø5     FONTYPF          PIC   X(Ø1).
           Ø5     FONTYPI          PIC   X(Ø1).
           Ø5     PITCH1L   COMP   PIC   S9(4).
           Ø5     PITCH1F          PIC   X(Ø1).
           Ø5     PITCH1I          PIC   X(Ø2).
           Ø5     PITCH2L   COMP   PIC   S9(4).
           Ø5     PITCH2F          PIC   X(Ø1).
           Ø5     PITCH2I          PIC   X(Ø2).
           Ø5     MARSUPL   COMP   PIC   S9(4).
           Ø5     MARSUPF          PIC   X(Ø1).
           Ø5     MARSUPI          PIC   X(Ø3).
           Ø5     MARLEFL   COMP   PIC   S9(4).
           Ø5     MARLEFF          PIC   X(Ø1).
           Ø5     MARLEFI          PIC   X(Ø3).
           Ø5     ORIENTL   COMP   PIC   S9(4).
           Ø5     ORIENTF          PIC   X(Ø1).
           Ø5     ORIENTI          PIC   X(Ø1).
           Ø5     COMPVEL   COMP   PIC   S9(4).
           Ø5     COMPVEF          PIC   X(Ø1).
           Ø5     COMPVEI          PIC   X(Ø2).
           Ø5     ERRO1L    COMP   PIC   S9(4).
           Ø5     ERRO1F           PIC   X(Ø1).
           Ø5     ERRO1I           PIC   X(66).
    Ø2  PCLSØØ1O   REDEFINES  PCLSØØ1I.
           Ø5     FILLER           PIC   X(12).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     IMPRESO          PIC   X(Ø4).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     FONTYPO          PIC   X(Ø1).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     PITCH1O          PIC   X(Ø2).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     PITCH2O          PIC   X(Ø2).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     MARSUPO          PIC   X(Ø3).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     MARLEFO          PIC   X(Ø3).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     ORIENTO          PIC   X(Ø1).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     COMPVEO          PIC   X(Ø2).
           Ø5     FILLER           PIC   X(Ø3).
           Ø5     ERRO1O           PIC   X(66).
*==========================================================*
 LINKAGE SECTION.
*==========================================================*
 Ø1  DFHCOMMAREA.
     Ø2     FILLER     PIC X(1ØØØ).
*==========================================================*
 PROCEDURE DIVISION.
*==========================================================*
 FIRST-TIME-ONLY.
*================*
     IF EIBCALEN = Ø
        MOVE LOW-VALUES TO PCLSØØ1I
        MOVE 123        TO EIBCALEN
        PERFORM LOAD-CONFIG1
        PERFORM SEND-INICIAL
        GO TO RETURN-TRANSID
     END-IF.
*
 OTHER-TIMES.
*============*
```

```
     MOVE DFHCOMMAREA TO COMMAREA
     PERFORM RECEIVE-MAP
     IF EIBAID = 3 OR EIBAID = 'C'
        GO TO CLEAR
     END-IF
     IF EIBAID = 4 OR EIBAID = 'D'
        PERFORM LOAD-CONFIG1
        PERFORM SEND-DATAONLY
        GO TO RETURN-TRANSID
     END-IF
     IF EIBAID = 5 OR EIBAID = 'E'
        PERFORM LOAD-CONFIG2
        PERFORM SEND-DATAONLY
        GO TO RETURN-TRANSID
     END-IF
     PERFORM VALIDATE-INPUT
     PERFORM BUILD-COMMAND-STRING
     PERFORM ACQUIRE-PRINTER THRU ACQUIRE-PRINTER-EXIT
     PERFORM START-TRANSACTION-2
     PERFORM SEND-MSG1
     GO TO RETURN-EXIT.
*==============================================================*
*
 VALIDATE-INPUT.
*===============*
     IF IMPRESI = SPACES OR = LOW-VALUES
        MOVE -1 TO IMPRESL
        PERFORM ERRO-INPUT
        PERFORM SEND-ALARM
        GO TO RETURN-TRANSID
     END-IF
     IF FONTYPI  NOT NUMERIC
        MOVE -1 TO FONTYPL
        PERFORM ERRO-NUMERICO
        PERFORM SEND-ALARM
        GO TO RETURN-TRANSID
     END-IF
     IF PITCH1I  NOT NUMERIC
        MOVE -1 TO PITCH1L
        PERFORM ERRO-NUMERICO
        PERFORM SEND-ALARM
        GO TO RETURN-TRANSID
     END-IF
     IF PITCH2I  NOT NUMERIC
        MOVE -1 TO PITCH2L
        PERFORM ERRO-NUMERICO
        PERFORM SEND-ALARM
        GO TO RETURN-TRANSID
     END-IF
     IF MARSUPI  NOT NUMERIC
        MOVE -1 TO MARSUPL
```

```
             PERFORM ERRO-NUMERICO
             PERFORM SEND-ALARM
             GO TO RETURN-TRANSID
          END-IF
          IF MARLEFI  NOT NUMERIC
             MOVE -1 TO MARLEFL
             PERFORM ERRO-NUMERICO
             PERFORM SEND-ALARM
             GO TO RETURN-TRANSID
          END-IF
          IF ORIENTI  NOT NUMERIC
             MOVE -1 TO ORIENTL
             PERFORM ERRO-NUMERICO
             PERFORM SEND-ALARM
             GO TO RETURN-TRANSID
          END-IF
          IF COMPVEI  NOT NUMERIC
             MOVE -1 TO COMPVEL
             PERFORM ERRO-NUMERICO
             PERFORM SEND-ALARM
             GO TO RETURN-TRANSID
          END-IF.
      *
       BUILD-COMMAND-STRING.
      *======================*
          IF FONTYPI = 1
             MOVE FONTYP1-STRING TO FONTYP-STRING
          END-IF
          IF FONTYPI = 2
             MOVE FONTYP2-STRING TO FONTYP-STRING
          END-IF

          MOVE PITCH1I TO PITCH1-STRING
          MOVE PITCH2I TO PITCH2-STRING
          MOVE MARSUPI TO MARSUP-STRING
          MOVE MARLEFI TO MARLEF-STRING
          MOVE ORIENTI TO ORIENT-STRING
          MOVE COMPVEI TO COMPVE-STRING.
      *
       ACQUIRE-PRINTER.
      *================*
          EXEC CICS IGNORE CONDITION
                           TERMIDERR
          END-EXEC
          PERFORM INQUIRE-PRINTER
          IF PRT-STATUS EQUAL DFHVALUE(ACQUIRED)
             GO TO ACQUIRE-PRINTER-EXIT
          END-IF
          EXEC CICS SET TERMINAL   (IMPRESI)
                        CREATESESS (DFHVALUE(CREATE))
                        SERVSTATUS (DFHVALUE(INSERVICE))
                        ACQSTATUS  (DFHVALUE(ACQUIRED))
```

```
     END-EXEC
*                  ACQSTATUS  (SLINE-ACQ)
     EXEC CICS DELAY INTERVAL (1Ø)
     END-EXEC
     PERFORM INQUIRE-PRINTER
     IF PRT-STATUS NOT EQUAL DFHVALUE (ACQUIRED)
        PERFORM SEND-MSG2
        GO TO RETURN-EXIT
     END-IF.
 ACQUIRE-PRINTER-EXIT.
*======================*
     EXIT.
 INQUIRE-PRINTER.
*=================*
     EXEC CICS INQUIRE TERMINAL  (IMPRESI)
                      ACQSTATUS (PRT-STATUS)
     END-EXEC.
 START-TRANSACTION-2.
*====================*
     EXEC CICS START TRANSID (TRANS-2)
                     FROM    (CONFIG-STRING)
                     LENGTH  (CONFIG-STRING-LENG)
                     TERMID  (IMPRESI)
     END-EXEC.
 LOAD-CONFIG1.
*==============*
     MOVE   1   TO FONTYPI
     MOVE  11   TO PITCH1I
     MOVE  ØØ   TO PITCH2I
     MOVE 1ØØ   TO MARSUPI
     MOVE 3ØØ   TO MARLEFI
     MOVE   Ø   TO ORIENTI
     MOVE  Ø7   TO COMPVEI.
 LOAD-CONFIG2.
*==============*
     MOVE   2   TO FONTYPI
     MOVE  18   TO PITCH1I
     MOVE  5Ø   TO PITCH2I
     MOVE 1ØØ   TO MARSUPI
     MOVE 3ØØ   TO MARLEFI
     MOVE   Ø   TO ORIENTI
     MOVE  Ø7   TO COMPVEI.
 RECEIVE-MAP.
*=============*
     EXEC CICS HANDLE CONDITION
                      MAPFAIL (CLEAR)
     END-EXEC
     EXEC CICS RECEIVE MAP('PCLSØØ1')
     END-EXEC.
 ERRO-NUMERICO.
*===============*
     MOVE  LOW-VALUES TO ERRO1I
     MOVE 'VALUE MUST BE NUMERIC' TO ERRO1I.
 ERRO-INPUT.
*===========*
     MOVE  LOW-VALUES TO ERRO1I
     MOVE 'PLEASE ENTER THE CURSOR FIELD' TO ERRO1I.
 SEND-INICIAL.
*=============*
     EXEC CICS SEND MAP ('PCLSØØ1')
                    ERASE
     END-EXEC.
 SEND-DATAONLY.
*==============*
     EXEC CICS SEND MAP ('PCLSØØ1')
                    DATAONLY
     END-EXEC.
 SEND-ALARM.
*===========*
     EXEC CICS SEND MAP ('PCLSØØ1')
                    DATAONLY
                    CURSOR
                    ALARM
                    FREEKB
     END-EXEC.
 SEND-MSG1.
*==========*
     EXEC CICS SEND FROM   (MSG1)
                    LENGTH (MSG-LENG)
                    ERASE
     END-EXEC.
 SEND-MSG2.
*==========*
     EXEC CICS SEND FROM   (MSG2)
                    LENGTH (MSG-LENG)
                    ERASE
     END-EXEC.
 RETURN-TRANSID.
*===============*
     EXEC CICS RETURN TRANSID  (TRANS-1)
                      COMMAREA (COMMAREA)
                      LENGTH   (EIBCALEN)
     END-EXEC.
 RETURN-EXIT.
*============*
     EXEC CICS RETURN
     END-EXEC.
 CLEAR.
*=====*
     EXEC CICS SEND CONTROL ERASE
     END-EXEC
     EXEC CICS RETURN
     END-EXEC
     GOBACK.
```

PCLP002 COBOL SOURCE

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PCLPØØ2.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
*==============================================================*
 WORKING-STORAGE SECTION.
*=======================*
 77   COMANDOS       PIC X(1ØØ).
 77   COMANDOS-LENG  PIC S9(4) COMP VALUE +71.
 77   CTLCHARH       PIC X          VALUE 'H'.
*==============================================================*
 PROCEDURE DIVISION.
*==============================================================*
     EXEC CICS IGNORE CONDITION
                     LENGERR
     END-EXEC
     EXEC CICS RETRIEVE INTO   (COMANDOS)
                       LENGTH (COMANDOS-LENG)
     END-EXEC
     EXEC CICS SEND FROM    (COMANDOS)
                    LENGTH  (COMANDOS-LENG)
                    CTLCHAR (CTLCHARH)
     END-EXEC
     EXEC CICS RETURN
     END-EXEC
     GOBACK.
```

PCLS001 BMS SOURCE

```
MAPSET   DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB),              *
               LANG=COBOL,TIOAPFX=YES,EXTATT=MAPONLY
PCLSØØ1  DFHMDI SIZE=(24,8Ø)
         DFHMDF POS=(Ø3,Ø8),LENGTH=26,ATTRB=(ASKIP,PROT),            *
               COLOR=YELLOW,
               INITIAL='PPPPPP CCCCCC LL    CCCCCC'
         DFHMDF POS=(Ø3,4Ø),LENGTH=3Ø,ATTRB=(ASKIP,PROT),            *
               COLOR=PINK,
               INITIAL='Send PCL commands to a printer'
         DFHMDF POS=(Ø4,Ø8),LENGTH=22,ATTRB=(ASKIP,PROT),            *
               COLOR=YELLOW,
               INITIAL='PP  PP CC     LL    CC'
         DFHMDF POS=(Ø5,Ø8),LENGTH=22,ATTRB=(ASKIP,PROT),            *
               COLOR=YELLOW,                                         *
               INITIAL='PPPPP  CC     LL    CC'
         DFHMDF POS=(Ø6,Ø8),LENGTH=22,ATTRB=(ASKIP,PROT),            *
               COLOR=YELLOW,                                         *
               INITIAL='PP     CC     LL    CC'
         DFHMDF POS=(Ø7,Ø8),LENGTH=26,ATTRB=(ASKIP,PROT),            *
               COLOR=YELLOW,                                         *
               INITIAL='PP     CCCCCC LLLLL CCCCCC'
         DFHMDF POS=(Ø7,4Ø),LENGTH=12,ATTRB=(ASKIP,PROT),              *
               COLOR=NEUTRAL,                                          *
               INITIAL='Printer....:'
IMPRESI  DFHMDF POS=(Ø7,53),LENGTH=Ø4,ATTRB=(UNPROT,FSET,IC),          *
               COLOR=NEUTRAL
         DFHMDF POS=(Ø7,58),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(11,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Font............:'
FONTYP   DFHMDF POS=(11,29),LENGTH=Ø1,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(11,31),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(11,39),LENGTH=26,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='1-Courier  2- LetterGothic'
         DFHMDF POS=(12,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Pitch...........:'
PITCH1   DFHMDF POS=(12,29),LENGTH=Ø2,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(12,32),LENGTH=Ø1,ATTRB=(ASKIP,PROT),              *
               COLOR=RED,                                              *
               INITIAL='.'
PITCH2   DFHMDF POS=(12,34),LENGTH=Ø2,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(12,37),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(12,39),LENGTH=14,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='Ø1.ØØ to 3Ø.ØØ'
         DFHMDF POS=(13,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Upper margin.....:'
MARSUP   DFHMDF POS=(13,29),LENGTH=Ø3,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(13,33),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(13,39),LENGTH=19,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='1 to 999 decipoints'
         DFHMDF POS=(14,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Left margin......:'
MARLEF   DFHMDF POS=(14,29),LENGTH=Ø3,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(14,33),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(14,39),LENGTH=19,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='1 to 999 decipoints'
         DFHMDF POS=(15,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Orientation......:'
```

```
ORIENT   DFHMDF POS=(15,29),LENGTH=Ø1,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(15,31),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(15,39),LENGTH=23,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='Ø-Vertical 1-Horizontal'
         DFHMDF POS=(16,1Ø),LENGTH=18,ATTRB=(ASKIP,PROT),              *
               COLOR=TURQUOISE,                                        *
               INITIAL='Vertical compress:'
COMPVE   DFHMDF POS=(16,29),LENGTH=Ø2,ATTRB=(NUM,FSET),                *
               COLOR=RED
         DFHMDF POS=(16,32),LENGTH=Ø1,ATTRB=(ASKIP,PROT)
         DFHMDF POS=(16,39),LENGTH=31,ATTRB=(ASKIP,PROT),              *
               COLOR=BLUE,                                             *
               INITIAL='1 (compressed) to 99 (expanded)'
ERRO1    DFHMDF POS=(19,Ø8),LENGTH=66,ATTRB=(ASKIP,PROT),              *
               COLOR=NEUTRAL
         DFHMDF POS=(22,Ø1),LENGTH=75,ATTRB=(ASKIP,PROT),              *
               COLOR=DEFAULT,                                          *
               INITIAL='Pre-set: F4 8Ø columns   F5 132 columns        *
                ENTER Send command  F3 Exit'
         DFHMSD TYPE=FINAL
         END
```

Systems Programmer
(Portugal) © Xephon 2002

Maintenance of CICS DB2 entries and transactions

At our installation we have CICS Transaction Server for OS/390 Version 1.2 and DB2 Version 5.1. We have developed a tool for the administration of DB2 entries and transactions. The REXX EXECs store information from CSD files into DB2 tables, prepare jobs for migration purposes, and use generated ISPF tables allowing the online update of CICS resource definitions.

This tool is designed according to our internal standards, so you should tailor the names to fit your installation standards.

The first step (member RCTDDL), which includes the creation of a table space and tables for each DB2 subsystem, is executed from SPUFI.

On the main menu, the target DB2 subsystem has to be specified and consequently the name of the corresponding CSD file is generated. The main menu is shown below:

```
------------------------- RCT MAIN MENU ----------------------------

 Command ===>                                            Scroll ===>
CSR


 DB2 system ===> DBT          CSD dsn ===>
CICSTS12.CICS.PSTEST29.DFHCSD
               (DSN\DBT)






              1. Generate JCL to load DB2 tables from DFHCSD

              2. Generate JCL to define DB2 entries and transactions
                            (for migration purposes)

              3. Maintenance of DB2 entries and transactions
```

The first option is to load DB2 tables with data from CSD files using the CICS utility DFHCSDUP, with the user exit program DFH$FORA (which produces sequential files for loading tables).

The second option generates JCL streams for migrating DB2 entries and transactions. We use program SQLISPF, published in the July and August 2001 issues of *DB2 Update* (code provided in this article).

The third option enables users to update (insert, delete, update) CSD files. It suppresses the ability to create duplicate transaction IDs in a CSD file. Program SQLISPF is also used in this option.

The panel generated when choosing the third option displays DB2 entries and transactions:

```
------------------ DB2E and DB2T preview ----- Row 46 to 6Ø of 677
 Command ===>                                           Scroll ===> CSR
 (Search: F object value     object: entry\tran\trans\desc\plan\group)
 DB2 system ===> DBT  SQL output dsn  ===>CICSTS12.CICS.PSTEST29.DFHCSD
 Line cmds: S-Select
 ----------------------------------------------------------------------
 S  ENTRY    TRAN    TRANS DESC                         PLAN    GROUP
 ----------------------------------------------------------------------
    ENBAØ1           BØ13 CURRENT ACCOUNTS              BANKØØ13 DB2
    ENBAØ2           BØ16 UPDATING VALUES IN TBPS       BANKØØ16 DB2
            TRBAØ2Ø2 AZRN UPDATING FOREIGN ACCOUNTS              DB2
```

```
ENBAØ3          BØ46 VIEW CURRENT ACCOUNTS          BANKØØ46 DB2
ENBAØ4               DB2ENTRY FOR BANK              BANKØØ79 DB2
        TRBAØ4Ø1 ALTA ALTERNATIVE ADDRESSES                  DB2
        TRBAØ4Ø2 BØ83 HISTORY OF CURRENT ACCOUNTS            DB2
        TRBAØ4Ø3 MATP HISTORY OF FOREIGN ACCOUNTS            DB2
        TRBAØ4Ø4 PLBR UPDATING USER NUMBERS                  DB2
        TRBAØ4Ø5 RACU GIRO ACCOUNTS                          DB2
        TRBAØ4Ø6 ST17 AUTHORIZATIONS                         DB2
ENDBØ1          DBTT DB2 TEST APPLICATION           DBTTEST  DB2
ENDBØ2          DBRG DB2 REORG TS/IX                DB2REORG DB2
ENDBØ3          SAPU SAPIENS ATTACHMENT             DB2SAPC  DB2
ENDIØ2               DB2ENTRY FOR CREDITS           DIMRØ445 DB2
```

The panel for updating DB2 entries looks like:

```
-------------------- DB2E edit ------------------------

 Command ===> _ (D-delete U-update I-insert)

           Db2Entry     ENBAØ4__

           Group        DB2_____
           Description  DB2ENTRY FOR BANK_________________
           Transaction  ____
           Plan         BANKØØ79
           Accountrec   TXID        (NONE\TASK\TXID\UOW)
           Authid       ________
           Authtype     USERID      (USERID\OPID\GROUP\SIGN\TERM\TX)
           Drollback    YES         (YES\NO)
           Planexit     ________
           Priority     HIGH_       (HIGH\MEDIUM\LOW)
           Protectnum   Ø___        (Ø\value)
           Threadlimit  1___        (Ø\value)
           Threadwait   YES_        (POOL\YES\NO)


 DB2 system ===> DBT  SQL output dsn ===> CICSTS12.CICS.PSTEST29.DFHCSD
```

The panel for updating DB2 transactions looks like:

```
---------------------- DB2T edit -------------------------

 Command ===> _ (D-delete U-update I-insert)


           Db2Tran      TRBAØ4Ø4

           Group        DB2_____

           Description  UPDATING USER NUMBERS___________
           Entry        ENBAØ4__
           Transid      PLBR

DB2 system ===> DBT   SQL output dsn ===> CICSTS12.CICS.PSTEST29.DFHCSD
```

RCTDDL

```
   — Creation of table space and tables.
   — This member should be executed from SPUFI for each DB2 subsystem.

   —DROP TABLESPACE DSNDBØ4.TSCICSØ1;
   —COMMIT;
   CREATE TABLESPACE TSCICSØ1
     IN DSNDBØ4
     USING STOGROUP SGDPØØ2
       PRIQTY 5ØØ
       SECQTY 5Ø
     LOCKSIZE ANY
     BUFFERPOOL BP2
     CLOSE NO;
   COMMIT;
   CREATE TABLE CICS.DB2ENTRY
      (DB2ENTRY      CHAR(8)   NOT NULL,
       RDOGROUP      CHAR(8)   NOT NULL,
       DESCRIPTION   CHAR(58)  NOT NULL WITH DEFAULT,
       TRANSID       CHAR(4)   NOT NULL WITH DEFAULT,
       ACCOUNTREC    CHAR(4)   NOT NULL WITH DEFAULT,
       AUTHID        CHAR(8)   NOT NULL WITH DEFAULT,
       AUTHTYPE      CHAR(6)   NOT NULL WITH DEFAULT,
       DROLLBACK     CHAR(3)   NOT NULL WITH DEFAULT,
       PLAN          CHAR(8)   NOT NULL WITH DEFAULT,
       PLANEXITNAME  CHAR(8)   NOT NULL WITH DEFAULT,
       PRIORITY      CHAR(5)   NOT NULL WITH DEFAULT,
       PROTECTNUM    CHAR(4)   NOT NULL WITH DEFAULT,
       THREADLIMIT   CHAR(4)   NOT NULL WITH DEFAULT,
       THREADWAIT    CHAR(4)   NOT NULL WITH DEFAULT,
      PRIMARY KEY (DB2ENTRY, RDOGROUP))
      IN DSNDBØ4.TSCICSØ1;
   CREATE UNIQUE INDEX CICS.XDB2EI
   ON CICS.DB2ENTRY (DB2ENTRY, RDOGROUP)
   USING STOGROUP SGXPØØ2
      PRIQTY 48
      SECQTY 12
      ERASE NO
   BUFFERPOOL BP5
   CLOSE NO;
   GRANT ALL ON CICS.DB2ENTRY TO PUBLIC;
   COMMIT;
   CREATE TABLE CICS.DB2TRAN
      (DB2TRAN       CHAR(8)   NOT NULL,
       RDOGROUP      CHAR(8)   NOT NULL,
       DESCRIPTION   CHAR(58)  NOT NULL WITH DEFAULT,
        ENTRY         CHAR(8)   NOT NULL WITH DEFAULT,
        TRANSID       CHAR(4)   NOT NULL WITH DEFAULT,
       PRIMARY KEY (DB2TRAN, RDOGROUP))
       IN DSNDBØ4.TSCICSØ1;
   CREATE UNIQUE INDEX CICS.XDB2TI
   ON CICS.DB2TRAN (DB2TRAN, RDOGROUP)
   USING STOGROUP SGXPØØ2
      PRIQTY 48
      SECQTY 12
      ERASE NO
   BUFFERPOOL BP5
   CLOSE NO;
   ALTER TABLE CICS.DB2TRAN
     FOREIGN KEY RCTCE (ENTRY, RDOGROUP) REFERENCES CICS.DB2ENTRY
     ON DELETE CASCADE;
   GRANT ALL ON CICS.DB2TRAN TO PUBLIC;
```

RCTR0

```
/* REXX - RCTRØ ****************************************************/
/* TITLE    :  APPLICATION START UP REXX                           */
/* *****************************************************************/
address ispexec 'select panel(rctp1)'
```

RCTP1

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- RCT MAIN MENU -$-$-[

%Command ===><Z[                                  %Scroll ===><Z   [


+DB2 system ===><Z   [         +CSD dsn ===>!Z                     [
               (DSN\DBT)






                 1. Generate JCL to load DB2 tables from DFHCSD

                 2. Generate JCL to define DB2 entries and transactions
                          (for migration purposes)
)INIT
   .ZVARS = '(ZCMD     ZSCR     DSN8SSID OCSD    )'
  &ZSCR   = CSR
  .CURSOR = ZCMD
  VGET (OCSD DSN8SSID) SHARED
  IF (&DSN8SSID = &Z)
    &DSN8SSID = DBT
    &OCSD = CICSTS12.CICS.PSTEST29.DFHCSD
)PROC
  VER  (&ZCMD,NB,LIST,1,2,3)
  &S = &ZCMD
  VER  (&DSN8SSID,NB,LIST,DSN,DBT)
  IF (&DSN8SSID = DBT)
    &OCSD = CICSTS12.CICS.PSTEST29.DFHCSD
  ELSE
    &OCSD = CICSTS12.CICS.DFHCSD
  VPUT (OCSD DSN8SSID S) SHARED
  &ZSEL = TRANS(TRUNC(&ZCMD,'.')
               1,'CMD(RCTR1)'
               2,'CMD(RCTR1)'
               3,'CMD(RCTR2)'
               '  ',' '
               *,'?')
)END
```

RCTP2

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) CAPS(ON)
+  TYPE(TEXT) INTENS(LOW)
]  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
!  TYPE(INPUT) INTENS(LOW) PADC(' ') CAPS(ON)
*  TYPE(OUTPUT) INTENS(LOW) COLOR(PINK)
;  TYPE(OUTPUT) INTENS(LOW) COLOR(TURQ)
¬  TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW)
)BODY DEFAULT(/,_)EXPAND($$)
%-$-$- DB2E and DB2T preview -$-$-[

%Command ===><Z                                    [%Scroll ===><Z   [
 (Search:%F object value[    object: entry\tran\trans\desc\plan\group)
+DB2 system ===>]Z   [  +SQL output dsn ===>]Z
[
%Line cmds:+S-Select [
+—————————————————————————————————————————[
%S  ENTRY [% TRAN [% TRANS DESC [                          % PLAN [ %
GROUP [
+—————————————————————————————————————————[
)MODEL
!Z *Z       ;Z       ¬Z   ]Z                                ]Z       ]Z
[
)INIT
   .ZVARS = '(ZCMD     ZSCR     DSN8SSID OCSD     OPT      TENTRY
TTRAN    +
              TTRANS   TDESC    TPLAN    TGROUP  )'
)END
```

RCTP3

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) PAD('_') CAPS(ON)
+  TYPE(INPUT) INTENS(LOW) PAD('_') CAPS(ON)
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- DB2E edit -$-$-[

%Command ===><Z[(D-delete U-update I-insert)

           Db2Entry    <Z       [

           Group       <Z       [
           Description +Z                         [
           Transaction +Z   [
           Plan        +Z       [
           Accountrec  +Z   [       (NONE\TASK\TXID\UOW)
           Authid      +Z       [
           Authtype    +Z     [     (USERID\OPID\GROUP\SIGN\TERM\TX)
           Drollback   +Z  [        (YES\NO)
           Planexit    +Z       [
           Priority    +Z    [      (HIGH\MEDIUM\LOW)
           Protectnum  +Z   [       (Ø\value)
           Threadlimit +Z   [       (Ø\value)
           Threadwait  +Z   [       (POOL\YES\NO)


%DB2 system ===>!Z   [  %SQL output dsn ===>!Z
[
)INIT
   .ZVARS = '(ZCMD     TENTRY   TGROUP   TDESC    TTRANS   TPLAN
TACCOUNT +
              TAUTHID  TAUTHTYP TDROLLBA TPLANEXI TPRIORIT TPROTECT
TTHREADL +
              TTHREADW DSN8SSID OCSD    )'
&ZSCR   = CSR
.CURSOR = ZCMD
)PROC
VER  (&TACCOUNT,NB,LIST,NONE,TASK,TXID,UOW)
VER  (&TAUTHTYP,NB,LIST,USERID,OPID,GROUP,SIGN,TERM,TX)
VER  (&TDROLLBA,NB,LIST,YES,NO)
VER  (&TPRIORIT,NB,LIST,HIGH,MEDIUM,LOW)
VER  (&TPROTECT,NUM)
VER  (&TTHREADL,NUM)
VER  (&TTHREADW,NB,LIST,POOL,YES,NO)
VER  (&ZCMD,NB,LIST,D,U,I)
IF (.PFKEY = PFØ3 & &ZCMD=D,U,I)
  &ZCMD = ' '
)END
```

RCTP4

```
)ATTR
%  TYPE(TEXT)
[  TYPE(TEXT) INTENS(LOW)
<  TYPE(INPUT) PAD('_') CAPS(ON)
+  TYPE(INPUT) INTENS(LOW) PAD('_') CAPS(ON)
!  TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
)BODY DEFAULT(]*;)EXPAND($$)
%-$-$- DB2T edit -$-$-[

%Command ===><Z[(D-delete U-update I-insert)



           Db2Tran     <Z       [



           Group       <Z       [

           Description +Z                            [

           Entry       +Z       [

           Transid     +Z   [





%DB2 system ===>!Z   [  %SQL output dsn ===>!Z
[
)INIT
   .ZVARS = '(ZCMD     TTRAN    TGROUP   TDESC    TENTRY1  TTRANS
DSN8SSID +
              OCSD    )'
&ZSCR   = CSR
.CURSOR = ZCMD
)PROC
VER  (&ZCMD,NB,LIST,D,U,I)
IF (.PFKEY = PFØ3 & &ZCMD=D,U,I)
```

```
  &ZCMD = ' '
)END
```

RCTR1

```
/* REXX - RCTR1 ****************************************************/
/* TITLE    :  JCL GENERATOR FOR LOAD AND MIGRATION                */
/* *************************************************************** */
/*     csd      - CSD file name (taken from panel as OCSD)         */
/*     forout   - input for load                                   */
/*     outdsn   - dataset name for JCL streams                     */
/*     DB2V     - DB2 subsystem name (taken from panel as DSN8SSID)  */
/*                                                                 */
/* *************************************************************** */
/* TRACE I                                                         */
  signal on error
/* *************************************************************** */
/* init values                                                     */
/* *************************************************************** */
  ADDRESS ISPEXEC "VGET (DSN8SSID OCSD S) SHARED"
  userid   = userid()
  tick     = ''''
  csd      = OCSD
  forout   = userid||'.FOROUT.PRIV'
  outdsn   = tick||userid||'.LOGCSD.CNTL'||tick
  DB2V = DSN8SSID
/* *************************************************************** */
  if sysdsn(outdsn) = 'OK' then
    "alloc fi(ispfile) da("outdsn") shr "
  else do
    "alloc fi(ispfile) da("outdsn") new ",
    "  dsorg(ps) space(1,1) tracks",
    "  recfm(F B) lrecl(132) blksize(27984)"
  end
select
 when S = 1 then do
  ADDRESS ISPEXEC "FTOPEN"
  ADDRESS ISPEXEC "FTINCL RCTS1"
  ADDRESS ISPEXEC "FTCLOSE"
 end
 when S = 2 then do
  queue '//'||userid||'X JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid
  queue '//**********************************************'
  queue '//* DEFINE DB2E AND DB2T                         '
  queue '//**********************************************'
  queue '//EXTRACT  EXEC PGM=DFHCSDUP,REGION=1M           '
  queue '//STEPLIB  DD DISP=SHR,DSN=CICSTS12.CICS.SDFHLOAD '
  queue '//DFHCSD   DD DISP=SHR,DSN='||csd
  queue '//SYSUT1   DD UNIT=SYSDA,SPACE=(1024,(100,100))   '
  queue '//SYSPRINT DD SYSOUT=*                           '
  queue '//SYSIN    DD *                                  '
```

```
"execio 1Ø diskw ispfile"
/* *************************************************** */
  SQLQUERY = "SELECT A.DB2ENTRY, ",
                     "A.RDOGROUP, ",
                     "A.DESCRIPTION, ",
                     "A.TRANSID, ",
                     "A.ACCOUNTREC, ",
                     "A.AUTHID, ",
                     "A.AUTHTYPE, ",
                     "A.DROLLBACK, ",
                     "A.PLAN, ",
                     "A.PLANEXITNAME, ",
                     "A.PRIORITY, ",
                     "A.PROTECTNUM, ",
                     "A.THREADLIMIT, ",
                     "A.THREADWAIT, ",
                     "B.DB2TRAN, ",
                     "B.RDOGROUP AS RDOT, ",
                     "B.DESCRIPTION AS DEST, ",
                     "B.ENTRY, ",
                     "B.TRANSID AS TRAT ",
              "FROM CICS.DB2ENTRY A LEFT OUTER JOIN CICS.DB2TRAN B ",
                   "ON A.DB2ENTRY = B.ENTRY AND ",
                      "A.RDOGROUP = B.RDOGROUP ",
              "ORDER BY A.DB2ENTRY, A.TRANSID, B.DB2TRAN";
  ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
  if  rc <> Ø then say 'Error'
  if  _nrows = Ø then say 'No record found'
  db2entry_m = ''
  do i = 1 to _nrows
    if strip(value(_vn.1"."strip(i,l,'Ø'))) ¬= db2entry_m then do
      db2entry_m = strip(value(_vn.1"."strip(i,l,'Ø')))
      if i > 1 then do
        queue '*'
        "execio 1 diskw ispfile"
      end
      queue 'DEFINE DB2ENTRY('||,
                    strip(value(_vn.1"."strip(i,l,'Ø')))||')'
      "execio 1 diskw ispfile"
      do k = 2 to 14
        if strip(value(_vn.k"."strip(i,l,'Ø'))) ¬= '' then do
          vn_m = _vn.k
          if _vn.k  = 'RDOGROUP' then vn_m = 'GROUP'
          queue '       '||vn_m||'('||,
                        strip(value(_vn.k"."strip(i,l,'Ø')))||')'
          "execio 1 diskw ispfile"
        end
      end
    end
    else do
      queue 'DEFINE DB2TRAN('||,
                    strip(value(_vn.15"."strip(i,l,'Ø')))||')'
        "execio 1 diskw ispfile"
        do k = 16 to _vn.Ø
          if strip(value(_vn.k"."strip(i,l,'Ø'))) ¬= '' then do
            select
              when _vn.k  = 'RDOT' then vn_m = 'GROUP'
              when _vn.k  = 'DEST' then vn_m = 'DESCRIPTION'
              when _vn.k  = 'TRAT' then vn_m = 'TRANSID'
              otherwise vn_m = _vn.k
            end
            queue '       '||vn_m||'('||,
                          strip(value(_vn.k"."strip(i,l,'Ø')))||')'
            "execio 1 diskw ispfile"
          end
        end
      end
    end
   queue '//'
   "execio 1 diskw ispfile"
   "execio Ø diskw ispfile(finis"
 end
 otherwise
end
signal off error
"free  fi(ispfile)"
"ispexec edit dataset("outdsn")"
signal on error
"ispexec lmerase dataset("outdsn")"
exit
error:
say 'error on line:' sigl '  ,rc:' rc
exit
```

RCTR2

```
/* REXX - RCTR2 *****************************************************/
/* TITLE    :  DB2E and DB2T preview and edit                      */
/* **************************************************************** */
/*     csd      - CSD file name (taken from panel as OCSD)          */
/*     DB2V     - DB2 subsystem name (taken from panel as DSN8SSID)  */
/*                                                                  */
/* **************************************************************** */
/* TRACE I                                                          */
  address ISPEXEC
  'CONTROL ERRORS RETURN '
  'VGET (DSN8SSID OCSD) SHARED'
  DB2V = DSN8SSID
  panel = 'rctp2'      /* panel to display the ispf table */
  tbnam = 'rctt1'      /* ispf table */
  libdd = 'ISPPROF'    /* ddname for ispf table */
  tvars = 'TENTRY TTRAN TTRANS TDESC TPLAN TGROUP'||,
          ' TACCOUNT TAUTHID TAUTHTYP TDROLLBA'||,
         ' TPLANEXI TPRIORIT TPROTECT TTHREADL'||,
         ' TTHREADW TENTRY1 TSORT1 TSORT2 TSORT3'
  cnt_f = Ø
  cnt_ft = ' '
  msg  = ' '
  csrrow = 1
  cursor  = 'OPT'
/* ************************************************************** */
  userid   = userid()
  csd      = OCSD
  tick     = ''''
  outdsn   = tick||userid||'.LOGCSD.CNTL'||tick
  outdsn1  = tick||userid||'.LOGCSD1.CNTL'||tick
  if DSN8SSID = 'DSN' then CICSNAME = 'CIC29ACT'
  else CICSNAME = 'PSTEST29'
/* ************************************************************** */
  ADDRESS ISPEXEC ,
  'TBERASE 'tbnam' LIBRARY('libdd')'
  'TBOPEN 'tbnam' WRITE SHARE LIBRARY('libdd')'
  if rc > Ø then do
    'TBCREATE 'tbnam' NAMES('tvars') WRITE SHARE LIBRARY('libdd') '
    if rc ¬= Ø then call sql_error rc
    call add_tabrows
    'TBTOP 'tbnam
  end
  disprc = Ø
  do while (disprc < 8)
    opt = ' '
    'TBQUERY 'tbnam' ROWNUM(rowcnt)'
    if csrrow <= Ø then csrrow = 1
    'TBDISPL 'tbnam' PANEL('panel') CSRROW('csrrow') MSG('msg') ,
             CURSOR('cursor') AUTOSEL(NO)'
    disprc = rc
    if disprc < 8 then do
      if word(strip(ZCMD), 1) = 'F' &,
        (word(strip(ZCMD), 2) = 'ENTRY' |,
         word(strip(ZCMD), 2) = 'TRAN' |,
         word(strip(ZCMD), 2) = 'TRANS' |,
         word(strip(ZCMD), 2) = 'DESC' |,
         word(strip(ZCMD), 2) = 'PLAN' |,
         word(strip(ZCMD), 2) = 'GROUP') then do
        if cnt_ft ¬= word(strip(ZCMD), 3) then do
          cnt_ft = word(strip(ZCMD), 3)
          cnt_f  = Ø
        end
        else cnt_f = cnt_f + 1
        call tab_search
      end
      else do
        if opt = 'S' then call process_s   /* Process line command */
      end
    end
  end
  'TBCLOSE 'tbnam' library('libdd')'
EXIT
/* Select a row */
  process_s:
    'CONTROL DISPLAY SAVE    '
    if TENTRY = ' ' then 'display panel(rctp4)'
    else 'display panel(rctp3)'
    select
      when ZCMD = 'D' then call process_d
      when ZCMD = 'U' then call process_u
      when ZCMD = 'I' then call process_a
      otherwise
    end
    'CONTROL DISPLAY RESTORE '
  return
/* Add a new row */
  process_a:
    if TTRANS <> ' ' then call check_trans 'INS'
    if TENTRY ¬= ' ' then do   /* db2e */
      SQLQUERY = "INSERT INTO CICS.DB2ENTRY ",
                 "VALUES('"||TENTRY||"','"||,
                             TGROUP||"','"||,
                             TDESC||"','"||,
                             TTRANS||"','"||,
                             TACCOUNT||"','"||,
                             TAUTHID||"','"||,
                             TAUTHTYP||"','"||,
                             TDROLLBA||"','"||,
                             TPLAN||"','"||,
                             TPLANEXI||"','"||,
                             TPRIORIT||"','"||,
                             TPROTECT||"','"||,
                             TTHREADL||"','"||,
                             TTHREADW||"')";
      ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
      if rc ¬= Ø then call sql_error rc
      call cics_defaltentry 'DEF'
      TTRAN          = ' '
      TENTRY1        = ' '
      TSORT1         = TENTRY
      TSORT2         = TTRAN
      TSORT3         = TTRANS
      call add_row
    end
    else do                   /* db2t */
      SQLQUERY = "INSERT INTO CICS.DB2TRAN ",
                 "VALUES('"||TTRAN||"','"||,
                             TGROUP||"','"||,
                             TDESC||"','"||,
                             TENTRY1||"','"||,
                             TTRANS||"')";
      ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
      if rc ¬= Ø then call sql_error rc
      call cics_defalttrans 'DEF'
      TENTRY          = ' '
      TPLAN           = ' '
      TACCOUNT        = ' '
      TAUTHID         = ' '
      TAUTHTYP        = ' '
      TDROLLBA        = ' '
      TPLANEXI        = ' '
      TPRIORIT        = ' '
      TPROTECT        = ' '
      TTHREADL        = ' '
      TTHREADW        = ' '
      TSORT1          = TENTRY1
      TSORT2          = TTRAN
      TSORT3          = TTRANS
      call add_row
    end
    'TBSORT 'tbnam' FIELDS(TSORT1,C,A,TSORT2,C,A,TSORT3,C,A)'
  return
/* Update a row */
  process_u:
    if TTRANS <> ' ' then call check_trans 'UPD'
    if TENTRY ¬= ' ' then do   /* db2e */
      SQLQUERY = "UPDATE CICS.DB2ENTRY ",
                 "SET DESCRIPTION = '"||TDESC||"',"||,
                     "TRANSID = '"||TTRANS||"',"||,
                     "ACCOUNTREC = '"||TACCOUNT||"',"||,
                     "AUTHID = '"||TAUTHID||"',"||,
                     "AUTHTYPE = '"||TAUTHTYP||"',"||,
                     "DROLLBACK = '"||TDROLLBA||"',"||,
                     "PLAN = '"||TPLAN||"',"||,
                     "PLANEXITNAME = '"||TPLANEXI||"',"||,
                     "PRIORITY = '"||TPRIORIT||"',"||,
                     "PROTECTNUM = '"||TPROTECT||"',"||,
                     "THREADLIMIT = '"||TTHREADL||"',"||,
                     "THREADWAIT = '"||TTHREADW||"' ",
                 "WHERE DB2ENTRY = '"||TENTRY||"' AND ",
                       "RDOGROUP = '"||TGROUP||"'";
      ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
      if rc ¬= Ø then call sql_error rc
      call cics_defaltentry 'ALT'
      TSORT3          = TTRANS
      'TBPUT 'tbnam
    end
    else do                   /* db2t */
      SQLQUERY = "UPDATE CICS.DB2TRAN ",
                 "SET DESCRIPTION = '"||TDESC||"',"||,
                     "ENTRY = '"||TENTRY1||"',"||,
                     "TRANSID = '"||TTRANS||"' ",
                  "WHERE DB2TRAN = '"||TTRAN||"' AND ",
                        "RDOGROUP = '"||TGROUP||"'";
       ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
       if rc ¬= Ø then call sql_error rc
       call cics_defalttrans 'ALT'
       TSORT1          = TENTRY1
       TSORT3          = TTRANS
       'TBPUT 'tbnam
     end
     'TBSORT 'tbnam' FIELDS(TSORT1,C,A,TSORT2,C,A,TSORT3,C,A)'
   return
/* Delete a row */
   process_d:
     if TENTRY ¬= ' ' then do   /* db2e */
       SQLQUERY = "DELETE FROM CICS.DB2ENTRY ",
                  "WHERE DB2ENTRY = '"||TENTRY||"' AND ",
                        "RDOGROUP = '"||TGROUP||"'";
       ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
       if rc ¬= Ø then call sql_error rc
       call cics_delentry1
       dimi = 1
       arrd.dimi = TENTRY
       'TBDELETE 'tbnam
       'TBSKIP 'tbnam
       do while (rc = Ø & TENTRY = ' ')
         call cics_deltrans1
         dimi = dimi + 1
         arrd.dimi = TTRAN
         'TBDELETE 'tbnam
         'TBSKIP 'tbnam
       End
       call cics_delentry2
     end
     else do                  /* db2t */
       SQLQUERY = "DELETE FROM CICS.DB2TRAN ",
                  "WHERE DB2TRAN = '"||TTRAN||"' AND ",
                        "RDOGROUP = '"||TGROUP||"'";
       ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
       if rc ¬= Ø then call sql_error rc
       call cics_deltrans
       'TBDELETE 'tbnam
     end
   return
/* Process the search */
tab_search:
   'TBVCLEAR 'tbnam
   tab_field = 'T' || word(strip(ZCMD), 2)
   obj = 'T'||word(strip(ZCMD), 2)
   interpret obj ' = word(strip(ZCMD), 3)'
   /* 'TBTOP 'tbnam */
   if cnt_f > Ø then 'TBSKIP 'tbnam' NUMBER('csrrow') NOREAD'
   'TBSARG 'tbnam' NEXT NAMECOND('tab_field',EQ)'
```

```
  'TBSCAN 'tbnam' NOREAD POSITION('crpname')'
  csrrow = crpname
  drop crpname
  if rc ¬= Ø then do
    zedlmsg = 'Not found. '
    'setmsg msg(ISRZØØØ) '
  end
return
/* Insert a row */
  add_row:
    'TBADD 'tbnam''
  return
/* Add rows in table */
  add_tabrows:
    SQLQUERY = "SELECT A.DB2ENTRY, ",
                      "A.TRANSID, ",
                      "A.DESCRIPTION, ",
                      "A.PLAN, ",
                      "A.RDOGROUP, ",
                      "B.DB2TRAN, ",
                      "B.DESCRIPTION AS DEST, ",
                      "B.RDOGROUP AS RDOT, ",
                      "B.TRANSID AS TRAT, ",
                      "A.ACCOUNTREC, ",
                      "A.AUTHID, ",
                      "A.AUTHTYPE, ",
                      "A.DROLLBACK, ",
                      "A.PLANEXITNAME, ",
                      "A.PRIORITY, ",
                      "A.PROTECTNUM, ",
                      "A.THREADLIMIT, ",
                      "A.THREADWAIT, ",
                      "B.ENTRY ",
               "FROM CICS.DB2ENTRY A LEFT OUTER JOIN CICS.DB2TRAN B ",
                    "ON A.DB2ENTRY = B.ENTRY AND ",
                       "A.RDOGROUP = B.RDOGROUP ",
               "ORDER BY A.DB2ENTRY, A.TRANSID, B.DB2TRAN";
    ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
    if  _nrows = Ø then do
      zedlmsg = 'No record found'
      'setmsg msg(ISRZØØØ) '
      exit 2Ø
    end
    db2entry_m = ''
    do i = 1 to _nrows
      if strip(value(_vn.1"."strip(i,l,'Ø'))) ¬= db2entry_m then do
        db2entry_m      = strip(value(_vn.1"."strip(i,l,'Ø')))
        call add_row_entry
        if strip(value(_vn.19"."strip(i,l,'Ø'))) <> '-'
        then call add_row_tran
      end
      else call add_row_tran
    end
  return
/* JCL to define/alter DB2E */
  cics_defaltentry:
    parse arg choice
      call alloc_free 'ALLOC' 'outdsn'
      ADDRESS ISPEXEC "FTINCL RCTS2"
      if choice = 'DEF' then do
        STRCMD = 'DEFINE DB2ENTRY('TENTRY')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      else do
        STRCMD = 'ALTER  DB2ENTRY('TENTRY')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      STRCMD = '       GROUP('TGROUP')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      if TDESC <> ' ' then do
        STRCMD = '       DESCRIPTION('TDESC')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TTRANS <> ' ' then do
        STRCMD = '       TRANSID('TTRANS')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      else do
        if choice = 'ALT' then do
          STRCMD = '       TRANSID()'
          ADDRESS ISPEXEC "FTINCL RCTS3"
        end
      end
      if TACCOUNT <> ' ' then do
        STRCMD = '       ACCOUNTREC('TACCOUNT')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TAUTHID <> ' ' then do
        STRCMD = '       AUTHTID('TAUTHID')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TAUTHTYP <> ' ' then do
        STRCMD = '       AUTHTYPE('TAUTHTYP')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TDROLLBA <> ' ' then do
        STRCMD = '       DROLLBACK('TDROLLBA')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TPLAN <> ' ' then do
        STRCMD = '       PLAN('TPLAN')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TPLANEXI <> ' ' then do
```

```
        STRCMD = '       PLANEXITNAME('TPLANEXI')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TPRIORIT <> ' ' then do
        STRCMD = '       PRIORITY('TPRIORIT')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TPROTECT <> ' ' then do
        STRCMD = '       PROTECTNUM('TPROTECT')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TTHREADL <> ' ' then do
        STRCMD = '       THREADLIMIT('TTHREADL')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TTHREADW <> ' ' then do
        STRCMD = '       THREADWAIT('TTHREADW')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      call alloc_free 'FREE' 'outdsn'
      call alloc_free 'ALLOC' 'outdsn1'
      ADDRESS ISPEXEC "FTINCL RCTS4"
      ADDRESS ISPEXEC "FTINCL RCTS5"
      if choice = 'ALT' then do
        STRCMD = "// F "CICSNAME",'CEMT",
                 "SET DB2E("TENTRY") DISABLED'"
        ADDRESS ISPEXEC "FTINCL RCTS3"
        STRCMD = "// F "CICSNAME",'CEMT",
                 "DISCARD DB2E("TENTRY")'"
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      STRCMD = "// F "CICSNAME",'CEDA",
               "I DB2E("TENTRY") GR("TGROUP")'"
      ADDRESS ISPEXEC "FTINCL RCTS3"
      call alloc_free 'FREE' 'outdsn1'
  return
/* JCL to define/alter DB2T */
  cics_defalttrans:
    parse arg choice
      call alloc_free 'ALLOC' 'outdsn'
      ADDRESS ISPEXEC "FTINCL RCTS2"
      if choice = 'DEF' then do
        STRCMD = 'DEFINE DB2TRAN('TTRAN')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      else do
        STRCMD = 'ALTER  DB2TRAN('TTRAN')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      STRCMD = '       GROUP('TGROUP')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      if TDESC <> ' ' then do
```

```
        STRCMD = '       DESCRIPTION('TDESC')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TENTRY1 <> ' ' then do
        STRCMD = '       ENTRY('TENTRY1')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      if TTRANS <> ' ' then do
        STRCMD = '       TRANSID('TTRANS')'
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      call alloc_free 'FREE' 'outdsn'
      call alloc_free 'ALLOC' 'outdsn1'
      ADDRESS ISPEXEC "FTINCL RCTS4"
      ADDRESS ISPEXEC "FTINCL RCTS5"
      if choice = 'ALT' then do
        STRCMD = "// F "CICSNAME",'CEMT",
                 "DISCARD DB2T("TTRAN")'"
        ADDRESS ISPEXEC "FTINCL RCTS3"
      end
      STRCMD = "// F "CICSNAME",'CEDA",
               "I DB2T("TTRAN") GR("TGROUP")'"
      ADDRESS ISPEXEC "FTINCL RCTS3"
      call alloc_free 'FREE' 'outdsn1'
  return
/* JCL to delete DB2E */
  cics_delentry1:
      call alloc_free 'ALLOC' 'outdsn'
      ADDRESS ISPEXEC "FTINCL RCTS2"
      STRCMD = 'DELETE DB2ENTRY('TENTRY')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      STRCMD = '       GROUP('TGROUP')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
  return
  cics_delentry2:
      call alloc_free 'FREE' 'outdsn'
      call alloc_free 'ALLOC' 'outdsn1'
      ADDRESS ISPEXEC "FTINCL RCTS4"
      ADDRESS ISPEXEC "FTINCL RCTS5"
      do i = 1 to dimi
        if i = 1 then do
          STRCMD = "// F "CICSNAME",'CEMT",
                   "SET DB2E("arrd.i") DISABLED'"
          ADDRESS ISPEXEC "FTINCL RCTS3"
          STRCMD = "// F "CICSNAME",'CEMT",
                   "DISCARD DB2E("arrd.i")'"
          ADDRESS ISPEXEC "FTINCL RCTS3"
        end
        else do
          STRCMD = "// F "CICSNAME",'CEMT",
                   "DISCARD DB2T("arrd.i")'"
          ADDRESS ISPEXEC "FTINCL RCTS3"
        end
      end
      call alloc_free 'FREE' 'outdsn1'
  return
/* JCL to delete DB2T */
  cics_deltrans:
      call alloc_free 'ALLOC' 'outdsn'
      ADDRESS ISPEXEC "FTINCL RCTS2"
      STRCMD = 'DELETE DB2TRAN('TTRAN')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      STRCMD = '       GROUP('TGROUP')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      call alloc_free 'FREE' 'outdsn'
      call alloc_free 'ALLOC' 'outdsn1'
      ADDRESS ISPEXEC "FTINCL RCTS4"
      ADDRESS ISPEXEC "FTINCL RCTS5"
      STRCMD = "// F "CICSNAME",'CEMT",
               "DISCARD DB2T("TTRAN")'"
      ADDRESS ISPEXEC "FTINCL RCTS3"
      call alloc_free 'FREE' 'outdsn1'
  return
  cics_deltrans1:
      STRCMD = 'DELETE DB2TRAN('TTRAN')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
      STRCMD = '       GROUP('TGROUP')'
      ADDRESS ISPEXEC "FTINCL RCTS3"
  return
  sql_error:
    parse arg errcode
        zedlmsg = 'Error 'errcode
        'setmsg msg(ISRZØØØ) '
        exit 2Ø
  return
  add_row_entry:
        TENTRY          = strip(value(_vn.1"."strip(i,l,'Ø')))
        TTRAN           = ' '
        TTRANS          = strip(value(_vn.2"."strip(i,l,'Ø')))
        TDESC           = strip(value(_vn.3"."strip(i,l,'Ø')))
        TPLAN           = strip(value(_vn.4"."strip(i,l,'Ø')))
        TGROUP          = strip(value(_vn.5"."strip(i,l,'Ø')))
        TACCOUNT        = strip(value(_vn.1Ø"."strip(i,l,'Ø')))
        TAUTHID         = strip(value(_vn.11"."strip(i,l,'Ø')))
        TAUTHTYP        = strip(value(_vn.12"."strip(i,l,'Ø')))
        TDROLLBA        = strip(value(_vn.13"."strip(i,l,'Ø')))
        TPLANEXI        = strip(value(_vn.14"."strip(i,l,'Ø')))
        TPRIORIT        = strip(value(_vn.15"."strip(i,l,'Ø')))
        TPROTECT        = strip(value(_vn.16"."strip(i,l,'Ø')))
        TTHREADL        = strip(value(_vn.17"."strip(i,l,'Ø')))
        TTHREADW        = strip(value(_vn.18"."strip(i,l,'Ø')))
        TENTRY1         = ' '
        TSORT1          = TENTRY
        TSORT2          = TTRAN
      TSORT3          = TTRANS
      call add_row
return
add_row_tran:
      TENTRY          = ' '
      TTRAN           = strip(value(_vn.6"."strip(i,l,'Ø')))
      TTRANS          = strip(value(_vn.9"."strip(i,l,'Ø')))
      TDESC           = strip(value(_vn.7"."strip(i,l,'Ø')))
      TPLAN           = ' '
      TGROUP          = strip(value(_vn.8"."strip(i,l,'Ø')))
      TACCOUNT        = ' '
      TAUTHID         = ' '
      TAUTHTYP        = ' '
      TDROLLBA        = ' '
      TPLANEXI        = ' '
      TPRIORIT        = ' '
      TPROTECT        = ' '
      TTHREADL        = ' '
      TTHREADW        = ' '
      TENTRY1         = strip(value(_vn.19"."strip(i,l,'Ø')))
      TSORT1          = TENTRY1
      TSORT2          = TTRAN
      TSORT3          = TTRANS
      call add_row
return
check_trans:
  parse arg check
  if check = 'INS' then do
    SQLQUERY = "select count(*) ",
               "from cics.db2entry ",
               "where rdogroup like 'DB2%' AND ",
               "      transid = '"TTRANS"' ",
               "union ",
               "select count(*) ",
               "from cics.db2tran  ",
               "where rdogroup like 'DB2%' AND ",
               "      transid = '"TTRANS"' "
  end
  else do
    if TENTRY ¬= ' ' then do
      SQLQUERY = "select count(*) ",
                 "from cics.db2entry ",
                 "where rdogroup like 'DB2%' AND ",
                 "      transid = '"TTRANS"' AND ",
                 "      db2entry <> '"TENTRY"' ",
                 "union ",
                 "select count(*) ",
                 "from cics.db2tran  ",
                 "where rdogroup like 'DB2%' AND ",
                 "      transid = '"TTRANS"' "
    end
```

```
      else do
        SQLQUERY = "select count(*) ",
                   "from cics.db2entry ",
                   "where rdogroup like 'DB2%' AND ",
                   "      transid = '"TTRANS"' ",
                   "union ",
                   "select count(*) ",
                   "from cics.db2tran  ",
                   "where rdogroup like 'DB2%' AND ",
                   "      transid = '"TTRANS"' AND ",
                   "      db2tran <> '"TTRAN"' "
      end
    end
    n = Ø
    ADDRESS ISPEXEC "SELECT PGM(SQLISPF) MODE(FSCR)";
    if rc ¬= Ø then call sql_error rc
    do i = 1 to _nrows
      n = n + _VN1.i
    end
    if n > Ø then do
      zedlmsg = 'Error: Transid 'TTRANS' already exists.'
      'setmsg msg(ISRZØØØ) '
      exit 2Ø
    end
  return
  alloc_free:
    parse arg af dsn
    if af = 'ALLOC' then do
      if sysdsn(dsn) = 'OK' then
        ADDRESS TSO ,
        "alloc fi(ispfile) da("value(dsn)") shr "
      else do
        ADDRESS TSO ,
        "alloc fi(ispfile) da("value(dsn)") new ",
        "  dsorg(ps) space(1,1) tracks",
        "  recfm(F B) lrecl(132) blksize(27984)"
      end
      ADDRESS ISPEXEC "FTOPEN"
    end
    else do
      ADDRESS ISPEXEC "FTCLOSE"
      ADDRESS TSO "free  fi(ispfile)"
      ADDRESS ISPEXEC "edit dataset("value(dsn)")"
      ADDRESS ISPEXEC "lmerase dataset("value(dsn)")"
    end
  return
```

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

Nikola Lazovic, Gordana Kozovic, Ivan Bugarinovic
DB2 System Administrator
Postal Savings Bank (Yugoslavia)

Exit-enabling program

Just this morning we had a question about enabling and disabling exits, and I was looking at a little exit-enabling program I wrote. I think the feature of being able to enable just about any exit, merely by changing the three variables at the beginning, is very useful. Note that even the operator messages are automatically adjusted.

```
&PGMID   SETC  'ENXPCREQ'
&EXIT    SETC  'XPCREQ'
&EXPGM   SETC  'ZUXPCREQ'
&PGMID   TITLE 'PLTPI PROGRAM TO ENABLE &EXIT EXIT &EXPGM'
&PGMID   AMODE ANY
&PGMID   RMODE ANY
*
         DFHREGS ,                  EQUATE REGISTERS
*       R3:   (CODEREG)
*       R1Ø:  (DATAREG)
*       R11:  (EIBREG)
*
DFHEISTG  DSECT
WRESP    DS   F
*
&PGMID   DFHEIENT CODEREG=R3,DATAREG=R1Ø,EIBREG=R11
         B     BEGIN
         DC    C'&PGMID &SYSDATE &SYSTIME'
BEGIN    DS   ØH
         EXEC  CICS HANDLE ABEND LABEL(RETURN)
         EXEC  CICS WRITE OPERATOR  TEXT(MSGØØ1)     NOHANDLE
         EXEC  CICS ENABLE PROGRAM('&EXPGM') EXIT('&EXIT') START
               RESP(WRESP)
         CLC   WRESP,DFHRESP(NORMAL)
         BNE   NOENABLE
         EXEC  CICS WRITE OPERATOR  TEXT(MSGØØ2)     NOHANDLE
         B     RETURN
NOENABLE  DS   ØH
         EXEC  CICS WRITE OPERATOR  TEXT(MSG1Ø1)     NOHANDLE
RETURN    DS   ØH
         EXEC  CICS RETURN
         LTORG
MSGØØ1  DC  C'&EXIT.ØØ1I PROGRAM(&EXPGM) EXIT(&EXIT) TO BE ENABLED'
MSGØØ2  DC  C'&EXIT.ØØ2I PROGRAM(&EXPGM) EXIT(&EXIT) ENABLED'
MSG1Ø1  DC  C'&EXIT.1Ø1W PROGRAM(&EXPGM) EXIT(&EXIT) NOT ENABLED'
         END
```

Taras Wolansky
Technical Consultant (USA)

CICS questions and answers

Q Is there a way to determine whether I'm (my program is) in CICS or not?

A The C function iscics() provides this; alternatively, the following Assembler code will work:

```
DFHAFCD TYPE=LOCATE
     LTR   15,15
     BZ    not cics
```

As this works under the EXCI interface the following should be added:

```
     USING DFHAFCD,15
     LA    R1,AFLSTBEG
     AH    R1,AFLENG
     USING AFTSTART,R1
     TM    AFTFLG1,AFTEXCI
     BO    not exci (so plain cics)
```

Q How do you write a sequential file from a CICS transaction?

A CICS supports sequential file access via the Transient Data API calls: WRITEQ TD, READQ TD... You need to define the file to CICS via an Extra Transient Data Queue definition. Then define an Indirect Transient Data Queue definition to connect the API calls QUEUE(Indirect Q name) with the file (Extra definition).

An EXEC CICS WRITEQ TD(Indirect Q) will write a record to the dataset pointed to by the Extra Q.

Q EXCI and SYNCONRETURN forced or not? I'm confused. I've read about CTG/390 being able to support ECI_EXTEND – but how, if SYNCONRETURN is forced through EXCI ?

A You need to be running RRS (speak to your MVS SysProg) and then set RRMS=Yes in the target CICS region.

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

CICS news

IBM has announced Version 1.1 of its CICS Online Transmission Time Optimizer for z/OS, which identifies and removes repetitive data and compresses 3270 data streams. This is designed to improve 3270 network resource utilization and response time.

Specifically, it examines outgoing data streams and dynamically compresses them, eliminates repetitive characters, and ensures that only changed data is sent to the terminals.

It also optimizes messages to increase printer speed, enables exclusion and inclusion of terminals dynamically, operates transparently to users and applications, monitors its own operation, and provides an optional exit to change the data stream in the optimized message.

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

* * *

IBM has announced CICS VSAM Recovery Version 3 Release 1, which automates the recovery of lost or damaged VSAM files on OS/390 machines.

Among the new elements are CICSVR VSAM batch logging, which logs changes to VSAM data sets made by batch applications. Also new is the CICSVR server address space, which provides a communication vehicle for CICSVR, CICSVR VSAM batch logging, and other DFSMS components.

There's also change accumulation processing, to speed up the forward recovery process, complete data set forward recovery automation using DFSMSdss copies and dumps, and support of MVS system logger log streams and CICS/ESA forward recovery logs.

Also new are enhanced remote recovery site commands and instructions on how to maintain a remote disaster recovery site. New Recovery Control Data Set (RCDS) EXPORT and IMPORT commands allow users to copy the RCDS information, which can be sent to a remote recovery site and loaded from a previously-exported data set back into the RCDS.

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

* * *

Versata has announced the general availability of its Versata Logic Suite 5.5. The product introduces a set of features for creating and managing the business logic for enterprise applications and supports the WebSphere 4.0 app server.

Other enhancements to the Versata Logic Suite include integration with technologies such as JSP, XML, Web Services, Lotus Notes, CICS, and MQSeries through adaptable integration solutions, known as Versata Toolkits.

For further information contact:
Versata, 300 Lakeside Drive, Suite 1500, Oakland, CA 94612, USA.
Tel: (800) 984 7638.
URL: http://www.versata.com/versata.vjsp?pageid=366.