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

© Xephon plc 2002
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.
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.

```plaintext
* Name:       JUPEXIT                                    *
* Author:     Andy Wright                                *
* Date:       29th October 2001                          *
* Copyright:  IBM Corporation 2001                      *
* 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. *
************************************************************************** *
DFHREDS                        Establish register equates
JUPEXIT  CSECT

        STM     R14,R12,12(R13)        Save the registers on entry
        BASR    R3,0                    Establish base register
        USING   *,R3                    Tell the assembler
        ICM     R4,15,0(R1)             Address record
        BZ      EOF                     If plist empty then eof
        CLC     0(0,R4),=CL4'&DFH'     Test for block hdr record
```

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 IXGDELETE 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 than 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) © IBM 2002

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
PPPPPP CC LL CC
PP CC LL CC
PP CCCCCC LLLLLL CCCCCC    Printer....: PR53
Font..............: 1  1-Courier 2- LetterGothic
Pitch............: 11 . 00 01.00 to 30.00
```
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)'.

*  
01 SLINE-INS PIC S9(8) COMP VALUE +73.
01 SLINE-ACQ PIC S9(8) COMP VALUE +69.
01 SLINE-CRE PIC S9(8) COMP VALUE +67.
01 PRT-STATUS PIC S9(8) COMP.
01 IMPRESSORA PIC X(4).
*  
01 CONFIG-STRING.
  02 INIT-STRING PIC X(O7) VALUE X'50506F66C6CD7'.
  02 FILLER PIC X(O3) VALUE X'6CF1C2'.
  02 FONTYP-STRING.
    03 FILLER PIC X(O4).
    03 PITCH1-STRING PIC X(O2).
    03 FILLER PIC X(O1).
    03 PITCH2-STRING PIC X(O2).
    03 FILLER PIC X(I0).
  02 FILLER PIC X(O5) VALUE X'6CF1C25093'.
  02 MARSUP-STRING PIC X(O3).
  02 FILLER PIC X(O1) VALUE 'Z'.
  02 FILLER PIC X(O5) VALUE X'6CF1C25093'.
  02 MARLEF-STRING PIC X(O3).
  02 FILLER PIC X(O1) VALUE 'U'.
  02 FILLER PIC X(O5) VALUE X'6CF1C25093'.
  02 ORIENT-STRING PIC X(O1).
  02 FILLER PIC X(O1) VALUE '0'.
  02 FILLER PIC X(O5) VALUE X'6CF1C25093'.
  02 COMPVE-STRING PIC X(O3).
  02 FILLER PIC X(O1) VALUE 'C'.
  02 FINIS-STRING PIC X(O7) VALUE X'50506F66F0F0F0'.
  02 END-OF-MESSAGE PIC X(O1) VALUE X'19'.
*  
01 COMMAREA.
  02 PCLS001I.
*  
  05 FILLER PIC X(12).
  05 IMPRESL COMP PIC S9(4).
  05 IMPRESF PIC X(O1).
  05 IMPRESI PIC X(O4).
  05 FONTYPL COMP PIC S9(4).
  05 FONTYPF PIC X(O1).
  05 FONTPYI PIC X(O1).
  05 PITCH1L COMP PIC S9(4).
  05 PITCH1F PIC X(O1).
  05 PITCH1I PIC X(O2).
  05 PITCH2L COMP PIC S9(4).
  05 PITCH2F PIC X(O1).
  05 PITCH2I PIC X(O2).
  05 MARSUPL COMP PIC S9(4).
  05 MARSUPF PIC X(O1).
  05 MARSUPI PIC X(O3).
  05 MARLEFL COMP PIC S9(4).
05 MARLEFF PIC X(01).
05 MARLEFI PIC X(03).
05 ORIENTL COMP PIC S9(4).
05 ORIENTF PIC X(01).
05 ORIENTI PIC X(01).
05 COMPVEL COMP PIC S9(4).
05 COMPVEF PIC X(01).
05 COMPVFI PIC X(02).
05 ERRO1L COMP PIC S9(4).
05 ERRO1F PIC X(01).
05 ERRO1II PIC X(66).

02 PCLS001C REDEFINES PCLS001I.
05 FILLER PIC X(12).
05 FILLER PIC X(03).
05 IMPRESO PIC X(04).
05 FILLER PIC X(03).
05 FONTPRO PIC X(01).
05 FILLER PIC X(03).
05 PITCH10 PIC X(02).
05 FILLER PIC X(03).
05 PITCH20 PIC X(02).
05 FILLER PIC X(03).
05 MARSUPO PIC X(03).
05 FILLER PIC X(03).
05 MARLEFO PIC X(03).
05 FILLER PIC X(03).
05 ORIENTO PIC X(01).
05 FILLER PIC X(03).
05 COMPVEO PIC X(02).
05 FILLER PIC X(03).
05 ERRO10 PIC X(66).

*==================================================================* LINKAGE SECTION. *==================================================================* 01 DFHCOMMAREA. 02 FILLER PIC X(1000). *==================================================================* PROCEDURE DIVISION. *==================================================================* FIRST-TIME-ONLY.
*--------------------*
IF EIBCALEN = 0
MOVE LOW-VALUES TO PCLS001I
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 ERR0-INPUT
   PERFORM SEND-ALARM
   GO TO RETURN-TRANSID
END-IF
IF FONTP1 NOT NUMERIC
   MOVE -1 TO FONTP1L
   PERFORM ERR0-NUMERICO
   PERFORM SEND-ALARM
   GO TO RETURN-TRANSID
END-IF
IF PITCH11 NOT NUMERIC
   MOVE -1 TO PITCH11L
   PERFORM ERR0-NUMERICO
   PERFORM SEND-ALARM
   GO TO RETURN-TRANSID
END-IF
IF PITCH21 NOT NUMERIC
   MOVE -1 TO PITCH21L
   PERFORM ERR0-NUMERICO
   PERFORM SEND-ALARM
   GO TO RETURN-TRANSID
END-IF
IF MARSUPI NOT NUMERIC
   MOVE -1 TO MARSUPL
PERFORM ERR0-NUMERIC
PERFORM SEND-ALARM
GO TO RETURN-TRANSID
END-IF
IF MARLEFI NOT NUMERIC
  MOVE -1 TO MARLEFL
  PERFORM ERR0-NUMERIC
  PERFORM SEND-ALARM
  GO TO RETURN-TRANSID
END-IF
IF ORIENTI NOT NUMERIC
  MOVE -1 TO ORIENTL
  PERFORM ERR0-NUMERIC
  PERFORM SEND-ALARM
  GO TO RETURN-TRANSID
END-IF
IF COMPVEI NOT NUMERIC
  MOVE -1 TO COMPVEL
  PERFORM ERR0-NUMERIC
  PERFORM SEND-ALARM
  GO TO RETURN-TRANSID
END-IF.
*
BUILD-COMMAND-STRING.
*==============================*
  IF FONTYP1 = 1
    MOVE FONTYP1-STRING TO FONTYP-STRING
  END-IF
  IF FONTYP2 = 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
    TERMIERR
  END-EXEC
  PERFORM INQUIRE-PRINTER
  IF PRT-STATUS EQUAL DFHVALUE(ACQUIRED)
    GO TO ACQUIRE-PRINTER-EXIT
  END-IF
  EXEC CICS SET TERMINAL (IMPRESI)
    CREATESS (DFHVALUE(CREATE))
    SERVSTATUS (DFHVALUE(INSERVICE))
    ACQSTATUS (DFHVALUE(ACQUIRED))
**END-EXEC**
*   ACQSTATUS (SLINE-ACQ)
**EXEC CICS DELAY INTERVAL (10)**
**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-LEN)
   TERMID (IMPRESI)**
**END-EXEC.**
**LOAD-CONFIG1.**
**================================================================**
**MOVE  1 TO FONTYP1**
**MOVE 11 TO PITCH11**
**MOVE 00 TO PITCH21**
**MOVE 100 TO MARSUPI**
**MOVE 300 TO MARLEFI**
**MOVE 0 TO ORIENTI**
**MOVE 07 TO COMPVEI.**
**LOAD-CONFIG2.**
**================================================================**
**MOVE  2 TO FONTYP1**
**MOVE 18 TO PITCH11**
**MOVE 50 TO PITCH21**
**MOVE 100 TO MARSUPI**
**MOVE 300 TO MARLEFI**
**MOVE 0 TO ORIENTI**
**MOVE 07 TO COMPVEI.**
**RECEIVE-MAP.**
**================================================================**
**EXEC CICS HANDLE CONDITION**
**   MAPFAIL (CLEAR)**
**END-EXEC**
**EXEC CICS RECEIVE MAP('PCLS001')**
**END-EXEC.**
**ERRO-NUMERICO.**
**================================================================**
**MOVE LOW-VALUES TO ERROII**
MOVE 'VALUE MUST BE NUMERIC' TO ERRO11.
ERROR-INPUT.

*================================*
MOVE LOW-VALUES TO ERRO11
MOVE 'PLEASE ENTER THE CURSOR FIELD' TO ERRO11.
SENT-Inicial.

*================================*
EXEC CICS SEND MAP ('PCLS001')
   ERASE
END-EXEC.
SEND-DATAONLY.

*================================*
EXEC CICS SEND MAP ('PCLS001')
   DATAONLY
END-EXEC.
SEND-ALARM.

*================================*
EXEC CICS SEND MAP ('PCLS001')
   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. PCLP002.
ENVIRONMENT DIVISION.
DATA DIVISION.
*==============================================================================*
WORKING-Storage SECTION.
*==============================================================================*
77 COMANDOS PIC X(100).
77 COMANDOS-LENG PIC S9(4) COMP VALUE +71.
77 CTLCHARH PIC X VALUE 'H'.
*==============================================================================*
PROCEDURE DIVISION.
*==============================================================================*
EXEC CICS IGNORE CONDITION
LENERR
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=NO,EXTATT=MAPONLY

PCLS001 DFHMDI SIZE=(24,80)
DFHMDF POS=(03,08),LENGTH=26,ATTRB=(ASKIP,PROT), *
COLOR=YELLOW,
INITIAL='PPPPPP CCCCCC LL CCCCCC'
DFHMDF POS=(03,40),LENGTH=30,ATTRB=(ASKIP,PROT), *
COLOR=PINK,
INITIAL='Send PCL commands to a printer'
DFHMDF POS=(04,08),LENGTH=22,ATTRB=(ASKIP,PROT), *
COLOR=WHITE,
INITIAL='PP PP CC LL CC'
DFHMDF POS=(05,08),LENGTH=22,ATTRB=(ASKIP,PROT), *
COLOR=WHITE,
INITIAL='PPPPPP CC LL CC'
DFHMDF POS=(06,08),LENGTH=22,ATTRB=(ASKIP,PROT), *
COLOR=WHITE,
INITIAL='PP CC LL CC'
DFHMDF POS=(07,08),LENGTH=26,ATTRB=(ASKIP,PROT), *
COLOR=WHITE,
INITIAL='PP CCCCC LLLLL CCCCC'
DFHMDF POS=(07,40),LENGTH=12,ATTRB=(ASKIP,PROT), COLOR=NEUTRAL,
INITIAL=' Printer....:'

IMPRESI DFHMDF POS=(07,53),LENGTH=04,ATTRB=(UNPROT,FSET,IC), COLOR=NEUTRAL
DFHMDF POS=(07,58),LENGTH=01,ATTRB=(ASKIP,PROT)
DFHMDF POS=(11,10),LENGTH=18,ATTRB=(ASKIP,PROT), COLOR=TURQUOISE,
INITIAL='Font.........':

FONTYP DFHMDF POS=(11,29),LENGTH=01,ATTRB=(NUM,FSET), COLOR=RED
DFHMDF POS=(11,31),LENGTH=01,ATTRB=(ASKIP,PROT)
DFHMDF POS=(11,39),LENGTH=26,ATTRB=(ASKIP,PROT), COLOR=BLUE,
INITIAL='1-Courier 2- LetterGothic'
DFHMDF POS=(12,10),LENGTH=18,ATTRB=(ASKIP,PROT), COLOR=TURQUOISE,
INITIAL='Pitch.........'

PITCH1 DFHMDF POS=(12,29),LENGTH=02,ATTRB=(NUM,FSET), COLOR=RED
DFHMDF POS=(12,32),LENGTH=01,ATTRB=(ASKIP,PROT), COLOR=RED,
INITIAL=':.'
PITCH2 DFHMDF POS=(12,34),LENGTH=02,ATTRB=(NUM,FSET), COLOR=RED
DFHMDF POS=(12,37),LENGTH=01,ATTRB=(ASKIP,PROT)
DFHMDF POS=(12,39),LENGTH=14,ATTRB=(ASKIP,PROT), COLOR=BLUE,
INITIAL='01.00 to 30.00'
DFHMDF POS=(13,10),LENGTH=18,ATTRB=(ASKIP,PROT), COLOR=TURQUOISE,
INITIAL='Upper margin.....'

MARSUP DFHMDF POS=(13,29),LENGTH=03,ATTRB=(NUM,FSET), COLOR=RED
DFHMDF POS=(13,33),LENGTH=01,ATTRB=(ASKIP,PROT)
DFHMDF POS=(13,39),LENGTH=19,ATTRB=(ASKIP,PROT), COLOR=BLUE,
INITIAL='1 to 999 decipoints'
DFHMDF POS=(14,10),LENGTH=18,ATTRB=(ASKIP,PROT), COLOR=TURQUOISE,
INITIAL='Left margin.....'

MARLEF DFHMDF POS=(14,29),LENGTH=03,ATTRB=(NUM,FSET), COLOR=RED
DFHMDF POS=(14,33),LENGTH=01,ATTRB=(ASKIP,PROT)
DFHMDF POS=(14,39),LENGTH=19,ATTRB=(ASKIP,PROT), COLOR=BLUE,
INITIAL='1 to 999 decipoints'
DFHMDF POS=(15,10),LENGTH=18,ATTRB=(ASKIP,PROT), COLOR=TURQUOISE,
INITIAL='Orientation......:'

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 REXXX 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.PTEST29.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 60 of 677
Command ===> Scroll ===> CSR
(Search: F object value object: entry\tran\trans\desc\plan\group)
DB2 system ===> DBT SQL output dsn ===>CICSTS12.CICS.PTEST29.DFHCSD
Line cmds: S-Select

<table>
<thead>
<tr>
<th>S</th>
<th>ENTRY</th>
<th>TRAN</th>
<th>TRANS DESC</th>
<th>PLAN</th>
<th>GROUP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ENBA01</td>
<td>B013</td>
<td>CURRENT ACCOUNTS</td>
<td>BANK0013 DB2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENBA02</td>
<td>B016</td>
<td>UPDATING VALUES IN TBPS</td>
<td>BANK0016 DB2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRBA0202</td>
<td>A2R7</td>
<td>UPDATING FOREIGN ACCOUNTS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
The panel for updating DB2 transactions looks like:

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

Db2Tran  TRBA0404
Group     DB2____
Description UPDATING USER NUMBERS_________
Entry     ENBA04__

DB2 system ===> DBT SQL output dsn ===> CICSTS12.CICS.PTEST29.DFHCSD

The panel for updating DB2 entries looks like:

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

Db2Entry  ENBA04__
Group     DB2____
Description DB2ENTRY FOR BANK___________________
Transaction ________________
Plan       BANK0079
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)
RCTDDL

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

```
-DROP TABLESPACE DSNDB04.TSCICS01;
-COMMIT;
CREATE TABLESPACE TSCICS01
  IN DSNDB04
  USING STOGROUP SGDP002
  PRIQTY 500
  SECQTY 50
  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 DSNDB04.TSCICS01;
CREATE UNIQUE INDEX CICS.XDB2EI
ON CICS.DB2ENTRY (DB2ENTRY, RDOGROUP)
USING STOGROUP SGXP002
  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 DSNDBO4.TSCICS01;
CREATE UNIQUE INDEX CICS.XDB2TI
ON CICS.DB2TRAN (DB2TRAN, RDOGROUP)
USING STOGROUP SGXP002
PRIQTY 48
SECVTY 12
ERASE NO
BUFFERPOOL BP5
CLOSE NO;
ALTER TABLE CICS.DB2TRAN
FOREIGN KEY RCTR0 (ENTRY, RDOGROUP) REFERENCES CICS.DB2ENTRY
ON DELETE CASCADE;
GRANT ALL ON CICS.DB2TRAN TO PUBLIC;

RCTR0

/* REXX - RCTR0 *************************************************************************/
/* TITLE : APPLICATION START UP REXX */
/* ******************************************************************************/
address ispexec 'select panel(rctpl)'

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 DFHCS2D
2. Generate JCL to define DB2 entries and transactions
   (for migration purposes)
3. Maintenance of DB2 entries and transactions

)`INIT
  .ZVARS = '(ZCMD  ZSCR  DSNBSSID OCSD  )'
&ZSCR = CSR
.CURSOR = ZCMD
VGET (OCSD DSNBSSID) SHARED
IF (&DSNBSSID = &Z)
  &DSNBSSID = DBT
&OCSD = CICSTS12.CICS.PTEST29.DFHCSD
)PROC
VER (&ZCMD,NB,LIST,1,2,3)
&S = &ZCMD
VER (&DSNBSSID,NB,LIST,DSN,DBT)
IF (&DSNBSSID = DBT)
  &OCSD = CICSTS12.CICS.PTEST29.DFHCSD
ELSE
  &OCSD = CICSTS12.CICS.DFHCSD
VPUT (OCSD DSNBSSID 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(TEXT) CAPS(ON)
  ] 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\trans\esp\plan\group) +DB2 system ====<Z [ +SQL output dsn ====<Z [ [ %Line cmds:+S-Select [ +--------------------------------------------[
  %S ENTRY [% TRAN [% TRANS DESC [ % PLAN [ %
GROUP [ +--------------------------------------------[
)MODEL
RCTP3

ATT
% 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[[@delete U-update I-insert]

Db2Entry <Z [ ]

Group <Z [ ]
Description +Z [ ]
Transaction +Z [ ]
Plan +Z [ ]
Accountrec +Z [ ] (NONE\TASK\TXID\UOW)
Authid +Z [ ]
Authype +Z [ ] (USERID\OPID\GROUP\SIGN\TERM\TX)
Drollback +Z [ ] (YES\NO)
Planexit +Z [ ]
Priority +Z [ ] (HIGH\MEDIUM\LOW)
Protectnum +Z [ ] (0\value)
Threadlimit +Z [ ] (0\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 DSNBSSID 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 (&TPRIORITY,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 = PF03 & &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 ===><$[(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 = PF03 & &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 DSNBSSID) */
/* */
/* **************************************************** */
/* TRACE I */
/* signal on error */
/* ************************************************** */
/* init values */
/* **************************************************** */
ADDRESS ISPEXC "VGET (DSNBSSID OCSD S) SHARED"
userid = userid()
tick = ' '
csd = OCSD
forout = userid||'.FOROUT.PRIV'
outdsn = tick||userid||'.LOGCSD.CNTL'||tick
DB2V = DSNBSSID
/* ************************************************** */
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) lrec(132) blks(27984)"
end
select
when S = 1 then do
  ADDRESS ISPEXC "FTOPEN"
  ADDRESS ISPEXC "FTINCL RCTS1"
  ADDRESS ISPEXC "FTCLOSE"
end
when S = 2 then do
  queue '//'||userid||'.X JOB MSGCLASS=X,CLASS=A,NOTIFY='||userid
  queue '="/****** DEFINE DB2E AND DB2T */
  queue '="/****** EXTRACT EXEC PGM=DFHCSDDUP,REGION=1M */
  queue '="/****** STEPLIB DD DISP=SHR,DSN=CICSTS12.CICS.SDFHLDS */
  queue '="/****** DFHCSD DD DISP=SHR,DSN='||csd
  queue '="/****** SYSUT1 DD UNIT=SYSDA,SPACE=(1024,(100,100)) */
  queue '="/****** SYSPRINT DD SYSOUT="* */
  queue '="/****** SYSIN DD */
"execicio 10 diskw ispfile"

/* --------------------------------------------------------------- */
SQLQUERY = "SELECT A.DB2ENTRY, ",
    "A.RDOGROUP, ",
    "A.DESCRIPTION, ",
    "A.TRANSID, ",
    "A.ACCOUNTREC, ",
    "A.AUTHID, ",
    "A.AUHTYPE, ",
    "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(SQLISP);"
if rc <> 0 then say 'Error'
if _nrows = 0 then say 'No record found'
db2entry_m = ''
do i = 1 to _nrows
    if strip(value(_vn1".strip(i,1,'0'))) = db2entry_m then do
        db2entry_m = strip(value(_vn1".strip(i,1,'0')))
    if i > 1 then do
        queue '*'
    "execicio 1 diskw ispfile"
end
queue 'DEFINE DB2ENTRY('||,
    strip(value(_vn1".strip(i,1,'0')))||')'
"execicio 1 diskw ispfile"
do k = 2 to 14
    if strip(value(_vn.k".strip(i,1,'0'))) = '' then do
        vn_m = _vn.k
    if _vn.k = 'RDOGROUP' then vn_m = 'GROUP'
        queue ' '||vn_m||'( '||,
            strip(value(_vn.k".strip(i,1,'0')))||')'
    "execicio 1 diskw ispfile"
end
end
else do
    queue 'DEFINE DB2TRAN('||,
        strip(value(_vn15".strip(i,1,'0')))||')'
"execio 1 diskw ispfile"
do k = 16 to _vn.0
  if strip(value(_vn.k)."strip(i,1,')'")) = ' ' 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,1,')'"))||')'
  "execio 1 diskw ispfile"
end
end
end

"execio 1 diskw ispfile"
"execio 0 diskw ispfile(finis"
end
otherwise
end
signal off error
"free fil(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 DSNBSSID) */
/* ****************************************** */
/* TRACE I */
/* ****************************************** */
address ISPEEXEC
'CONTROL ERRORS RETURN '
'VGET (DSNBSSID OCSD) SHARED'
DB2V = DSNBSSID
panel = 'rctp2' /* panel to display the ispf table */
tbnum = 'rcttl' /* ispf table */
libdd = 'ISPPROF' /* ddname for ispf table */
tvars = 'TENTRY TTRAN TTRANS TDDESC TPLAN TGROUP',
  'TACCOUNT TAUTHID TAUTHTYPE TDROLLBA'"
' TPLANEXI TPRORIT TPROTECT TTHREADL'\',
'TTHREADW TENTRY1 TSORT1 TSORT2 TSORT3'
cnt_f = 0
cnt_ft = '.
msg = '.
csrrow = 1
cursor = 'OPT'
/* ********************************************** */
userid = userid()
csd = OCSD
tick = '....'
outsdn = tick||userid||'.LOGCSD.CNTL'||tick
outsdn1 = tick||userid||'.LOGCSD1.CNTL'||tick
if DSN8SSID = 'DSN' then CICSNAME = 'CIC29ACT'
else CICSNAME = 'PSTEST29'
/* ********************************************** */
ADDRESS ISPsEXEC,
'TBERASE 'tnam' LIBRARY('libdd')'
'TBOPEN 'tnam' WRITE SHARE LIBRARY('libdd')'
if rc > 0 then do
   'TBCREATE 'tnam' NAMES('tvars') WRITE SHARE LIBRARY('libdd') '
   if rc = 0 then call sql_error rc
call add_tabrows
   'TBTOP 'tnam'
end
disprc = 0
do while (disprc < 8)
opt = '.
   'TBQUERY 'tnam' ROWNUM(rowcnt)'
   if csrrow <= 0 then csrrow = 1
   'TBDISPL 'tnam' 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 = 0
      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(''|TENTRY1'||',','|',
                         TGROUP||'','|',
                         TDESC||'','|',
                         TTRANS||'','|',
                         TACCOUNT||'','|',
                         TAUTHID||'','|',
                         TAUTHYPE||'','|',
                         TDROLLBA||'','|',
                         TPLAN||'','|',
                         TPLANEXI||'','|',
                         TPRIORITY||'','|',
                         TPROTECT||'','|',
                         TTHREAD||'','|',
                         TTHREAD1||'','|',
                        "");
    ADDRESS ISPEXEC "SELECT PGM(SQNSPF)";
    if rc = 0 then call sql_error rc
    call cics_defaftentry '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||'','|',
                        "");
    ADDRESS ISPEXEC "SELECT PGM(SQNSPF)";
    if rc = 0 then call sql_error rc
    call cics_defaftentry 'DEF'
      TTRAN          = ' ';
      TENTRY1        = ' ';
      TSORT1         = TENTRY
      TSORT2         = TTRAN
      TSORT3         = TTRANS
    call add_row
  end
TTRANS|'"')";
ADDRESS ISPEXEC "SELECT PGM(SQLISP)"
if rc ≠ 0 then call sql_error rc
call cics_defaltttrans 'DEF'
ENTRY = ' ' 
PLAN = ' ' 
ACCOUNT = ' ' 
THID = ' ' 
AUTHYTP = ' ' 
TDROLLBA = ' ' 
PLANEXI = ' ' 
TPRIORIT = ' ' 
TPROTECT = ' ' 
THREADL = ' ' 
THREADW = ' ' 
TSORT1 = TENTRY1 
TSORT2 = TTRAN 
TSORT3 = TTRAN 
call add_row
end 'TSORT 'tnam' 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||'":"||
"TRANID = '"||TTRANS||'":"||
"ACCTOUNTREC = '"||ACCOUNT||'":"||
"AUTHID = '"||AUTHYTP||'":"||
"AUTHYTP = '"||AUTHYTP||'":"||
"DROLLBACK = '"||TDROLLBA||'":"||
"PLAN = '"||PLAN||'":"||
"PLANEXITNAME = '"||PLANEXI||'":"||
"PRIORITY = '"||TPRIORIT||'":"||
"PROTECTNUM = '"||TPROTECT||'":"||
"THREADLIMT = '"||THREADL||'":"||
"THREADWAIT = '"||THREADW||'":"||
"WHERE DB2ENTRY = '"||TENTRY||'" AND ",
"RDOGROUP = '"||TGROUP||'":"||
ADDRESS ISPEXEC "SELECT PGM(SQLISP)"
if rc ≠ 0 then call sql_error rc
call cics_defaltentry 'ALT'
'TBPUT 'tnam
end else do /* db2t */
SQLQUERY = "UPDATE CICS.DB2TRAN ",
"SET DESCRIPTION = '"||TDESC||'":"||
"ENTRY = '"||TENTRY1||'":"||
"TRANID = '"||TTRANS||'":"||
WHERE DB2TRAN = '"'||TTRAN||'"' AND ",
"RDOGROUP = '"'||TGROUP||'"'';
ADDRESS ISPEXEC "SELECT PGM(SQLISP)"
if rc ≠ 0 then call sql_error rc
call cics_defalttrans 'ALT'
TSORT1 = TENTRY1
TSORT3 = TTRANS
'TBPUT 'tnam
end
'TBSORT 'tnam' 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(SQLISP)"
if rc ≠ 0 then call sql_error rc
call cics_delentry1
dimi = 1
arrd.dimi = TENTRY
'TBDELETE 'tnam
'TBSKIP 'tnam
do while (rc = 0 & TENTRY = ' ')
call cics_deltrans1
dimi = dimi + 1
arrd.dimi = TTRAN
'TBDELETE 'tnam
'TBSKIP 'tnam
end
call cics_delentry2
end
else do /* db2t */
SQLQUERY = "DELETE FROM CICS.DB2TRAN ",
"WHERE DB2TRAN = '"'||TTRAN||'"' AND ",
"RDOGROUP = '"'||TGROUP||'"'';
ADDRESS ISPEXEC "SELECT PGM(SQLISP)"
if rc ≠ 0 then call sql_error rc
call cics_deltrans
'TBDELETE 'tnam
end
return
/* Process the search */
tab_search:
'TBVCLEAR 'tnam
tab_field = 'T' || word(strip(ZCMD), 2)
obj = 'T'||word(strip(ZCMD), 2)
interpret obj ' = word(strip(ZCMD), 3)'
/* 'TBTOP 'tnam */
if cnt_f ≠ 0 then 'TBSKIP 'tnam' NUMBER('csrrow') NOREAD'
'TBSARG 'tnam' NEXT NAMECOND('tab_field',EQ)"
'TBSCAN 'tbnam' NOREAD POSITION('crpname')'
csrrow = crpname
drop crpname
if rc /= 0 then do
  zed1msg = 'Not found.'
  'setmsg msg(ISRZ000)' '
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.RODGROUP,",
               "B.DB2TRAN,",
               "B.DESCRIPTION AS DEST,",
               "B.RODGROUP 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.RODGROUP = B.RODGROUP",
               "ORDER BY A.DB2ENTRY, A.TRANSID, B.DB2TRAN";
ADDRESS ISPEXEC "SELECT PGM(SQLISPF)";
if _nrows = 0 then do
  zed1msg = 'No record found'
  'setmsg msg(ISRZ000)' '
  exit 20
end
db2entry_m = ''
do i = 1 to _nrows
  if strip(value(_vn.1).strip(i,1,'0')) = db2entry_m then do
    db2entry_m = strip(value(_vn.1).strip(i,1,'0'))
    call add_row_entry
    if strip(value(_vn.19).strip(i,1,'0')) <> '-'
      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 = ' AUTHID('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_dentry1:
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_dentry2:
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
end
call alloc_free 'FREE' 'outdsn'
return
/* JCL to delete DB2T */
cics_deltrans:
call alloc_free 'ALLOC' 'outdsn'
ADDRESS ISPXEC "FTINCL RCTS2"
STRCMD = 'DELETE DB2TRAN('TTRAN')'
ADDRESS ISPXEC "FTINCL RCTS3"
STRCMD = ' ' GROUP('TGROUP')'
ADDRESS ISPXEC "FTINCL RCTS3"
call alloc_free 'FREE' 'outdsn'
call alloc_free 'ALLOC' 'outdsn1'
ADDRESS ISPXEC "FTINCL RCTS4"
ADDRESS ISPXEC "FTINCL RCTS5"
STRCMD = ' // F "CICSNAME",'CEMT',
"DISCARD DB2T("TTRAN")'
ADDRESS ISPXEC "FTINCL RCTS3"
call alloc_free 'FREE' 'outdsn1'
return

cics_deltrans1:
STRCMD = 'DELETE DB2TRAN('TTRAN')'
ADDRESS ISPXEC "FTINCL RCTS3"
STRCMD = ' ' GROUP('TGROUP')'
ADDRESS ISPXEC "FTINCL RCTS3"
return

sql_error:
parse arg errcode
zedlmsg = 'Error 'errcode
'setmsg msg(ISRZ000)'
exit 20
return

add_row_entry:
TENTRY = strip(value(_vn.1"strip(i,1,'\0')))
TTRAN = '
TTRANS = strip(value(_vn.2"strip(i,1,'\0')))
TDESC = strip(value(_vn.3"strip(i,1,'\0')))
TPLAN = strip(value(_vn.4"strip(i,1,'\0')))
TGROUP = strip(value(_vn.5"strip(i,1,'\0')))
TACOUNT = strip(value(_vn.10"strip(i,1,'\0')))
TAUTHID = strip(value(_vn.11"strip(i,1,'\0')))
TAUTHYTP = strip(value(_vn.12"strip(i,1,'\0')))
TDROLLBA = strip(value(_vn.13"strip(i,1,'\0')))
TPLANEXI = strip(value(_vn.14"strip(i,1,'\0')))
TPRIORIT = strip(value(_vn.15"strip(i,1,'\0')))
TPROTECT = strip(value(_vn.16"strip(i,1,'\0')))
TTHREAD1 = strip(value(_vn.17"strip(i,1,'\0')))
TTHREADW = strip(value(_vn.18"strip(i,1,'\0')))
TENTRY1 = '
TSORT1 = TENTRY
TSORT2 = TTRAN
TSORT3 = TTRANS

call add_row

return

add_row_tran:
  TENTRY = ''
  TTRAN = strip(value(_vn.6"strip(i,1,'\0')"))
  TTRANS = strip(value(_vn.9"strip(i,1,'\0')"))
  TDESC = strip(value(_vn.7"strip(i,1,'\0')"))
  TPLAN = ''
  TGROUP = strip(value(_vn.8"strip(i,1,'\0')"))
  TACCOUNT = ''
  TAUTH = ''
  TAUTHID = ''
  TROLLBA = ''
  TPLANEXI = ''
  TPRORIT = ''
  TPROTECT = ''
  TTHREAD = ''
  TTHREADW = ''
  TENTRY1 = strip(value(_vn.19"strip(i,1,'\0')"))
  TSORT1 = TENTRY1
  TSORT2 = TTRAN
  TSORT3 = TTRANS

  call add_row

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 = 0
ADDRESS ISPEEXEC "SELECT PGM(SQLISP) MODE(FSCR)";
if rc <> 0 then call sql_error rc
do i = 1 to _nrows
    n = n + _VN1.i
end
if n > 0 then do
    zed1msg = 'Error: Transid 'TTRANS' already exists.'
    'setmsg msg(ISRZ000)'
    exit 20
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) lrec(132) blksiz(27984)"
end
ADDRESS ISPEEXEC "FTOPEN"
end
else do
    ADDRESS ISPEEXEC "FTCLOSE"
    ADDRESS TSO "free fi(ispfile)"
    ADDRESS ISPEEXEC "edit dataset("value(dsn)")"
    ADDRESS ISPEEXEC "Imerase 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) © Xephon 2002
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 'PLTPN PROGRAM TO ENABLE &EXIT EXIT &EXPGM'
&PGMID  AMODE ANY
&PGMID  RMODE ANY
*
   DFHREGS,                EQUATE REGISTERS
*
   R3:  (CODEREG)         
*
   R10:  (DATAREG)         
*
   R11:  (EIBREG)          
*
DFHEISTG DSECT
WRESP DS F
*
&PGMID  DFHEIENT CODEREG=R3,DATAREG=R10,EIBREG=R11
BEGIN
   DC  C'&PGMID &SYSDATE &SYSTIME'
BEGIN DS 0H
   EXEC  CICS HANDLE ABEND LABEL(RETURN)
   EXEC  CICS WRITE OPERATOR TEXT(MSG001) NOHANDLE
   EXEC  CICS ENABLE PROGRAM('&EXPGM') EXIT('&EXIT') START
       RESP(WRESP)
   CLC  WRESP,DFHRESP(NORMAL)
   BNE  NOENABLE
   EXEC  CICS WRITE OPERATOR TEXT(MSG002) NOHANDLE
   B   RETURN
NOENABLE DS 0H
   EXEC  CICS WRITE OPERATOR TEXT(MSG101) NOHANDLE
RETURN DS 0H
   EXEC  CICS RETURN
LTORG
MSG001 DC  C'EEXIT.001I PROGRAM(&EXPGM) EXIT(&EXIT) TO BE ENABLED'
MSG002 DC  C'EEXIT.002I PROGRAM(&EXPGM) EXIT(&EXIT) ENABLED'
MSG101 DC  C'EEXIT.101I PROGRAM(&EXPGM) EXIT(&EXIT) NOT ENABLED'
END
```

Taras Wolansky
Technical Consultant (USA)  © Xephon 2002
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:

```assembly
DFHAFC TYPE=LOCATE
 LTR 15,15
 BZ not cics
```

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

```assembly
USING DFHAFC,15
LA R1,AFLSTBEG
AH R1,AFLNGL
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.

© Xephon 2002
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.


* * *

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.


* * *

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.