31 VSAM

October 1998

update

In this issue

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Subscriptions and back-issues
A year's subscription to *VSAM Update*, comprising four quarterly issues, costs £115.00 in the UK; $170.00 in the USA and Canada; £121.00 in Europe; £128.00 in Australasia and Japan; and £125.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the April 1991 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Editorial panel
Articles published in *VSAM Update* are reviewed by our panel of experts. Members of the panel include John Bradley (UK), Ernie Ishman (USA), and Rem Perretta (UK).

Editor
Fiona Hewitt

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

Contributions
Articles published in *VSAM Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

VSAM Update on-line
Code from *VSAM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

REXX extensions for VSAM – continued

Here, we continue the article, begun in the last issue, which looks at VSAM file processing, date processing, and data conversion.

$SHOWRPL

```
+-------------------------------------------------------------+
! code=$SHOWRPL(ddname, variable, parameter)                   !
+-------------------------------------------------------------+
```

This function assigns an area of the RPL of the VSAM file with the DDNAME ‘ddname’ to the variable ‘variable’.

For the returned value, see the note in the section on the $CLOSE function (in the previous issue).

The ‘parameter’ argument specifies which area of the RPL you want to obtain. The following values are possible:

- ‘RBA’ – RBA of the processed record.
- ‘RECLEN’ – record length.
- ‘KEYLEN’ – length of the generic key.
- ‘FDBK’ – reason code.
- ‘AREA’ – address of the work area.
- ‘AREALEN’ – length of the work area.
- ‘ARG’ – address of the area containing the search argument.
- ‘FTNCD’ code of the function in error.

More details can be obtained from the IBM documentation for the SHOWCB RPL=macro, FIELDS= keyword.

Data is returned into the variable in binary format, and is four bytes long.

An example of the $SHOWRPL function is given below:

```
CODE=$SHOWRPL('SYSUT1','VAR1','AREALEN')
```

$STOW

```
+-------------------------------------------------------------+
! code=$STOW(ddname, member, operation <, parameter>)          !
+-------------------------------------------------------------+
```

This function updates the directory of the partitioned dataset with the DDNAME ‘ddname’. The member processed is the one with the name ‘member’.

This function returns the return code from the STOW macro (see IBM documentation for further details).

The ‘parameter’ argument depends on the requested ‘operation’, which must be one of the following:

- ‘A’. The member is added into the directory. All previously written data is part of this member. Data supplied in ‘parameter’ is stored in the directory entry for this member.
- ‘R’. The specified member is replaced. Data supplied in ‘parameter’ is stored in the directory entry for this member.
- ‘D’. The member is deleted. Data supplied in ‘parameter’ is ignored.
- ‘C’. The member name is changed. The ‘parameter’ argument must contain the new name.

One entry of a directory for a PDS processed by ISPF is 42-bytes long, and contains the member name, the TTR, and the byte containing the length of this entry in halfword, as well as the following data (30 bytes in length):

- Offet 0 – 1 byte containing the version number (VV) in binary.
- Offset 1 – 1 byte containing the modification number (MM) in binary.
- Offset 2 – 3 bytes '000000'X.

- Offset 5 – 3 bytes containing the creation date in packed decimal, in the YYQQQ format. The sign must be 'F'X.
- Offset 8 – 1 byte '00'X.
- Offset 9 – 3 bytes containing the modification date in packed decimal, in the YYQQQ format. The sign must be 'F'X.
- Offset 12 – 2 bytes containing the modification hour in packed decimal without any sign, in the HHMM format.
- Offset 14 – 2 bytes containing the number of records, in binary.
- Offset 16 – 2 bytes containing the initial number of records, in binary.
- Offset 18 – 2 bytes containing the modification number, in binary.
- Offset 20 – 7 bytes containing the name of the user who made the last modification.
- Offset 27 – 3 bytes '404040'X.

An example of the $STOW function is given below:

```
CODE=$STOW('SYSUT1','NEWMEMB','A')
```

Sample programs for files processing

The following programs are designed to help you to get started with REXXPLUS/MVS files processing. They also show how to use the various parameters for $OPEN and $MODRPL. You should use the same method as that used in Assembler.

Copying a file into another file

Below is an example of a program copying one sequential file into another.

```
ADDRESS TSO
"ALLOC DD(F) DA('YOUR.INPUT.FILE') SHR"
"ALLOC DD(G) DA('YOUR.OUTPUT.FILE') SHR"
RC=$OPEN('F','INPUT'); SAY 'OPEN RETURN CODE F=' RC
RC=$OPEN('G','OUTPUT'); SAY 'OPEN RETURN CODE G=' RC
N=0; F_EOF=0
```

```
RC=$GET('F')
DO WHILE F_EOF=Ø
   N=N+1
   RC=$PUT('G',F_RECORD)
   RC=$GET('F')
END
RC=$CLOSE('F'); RC=$CLOSE('G')
SAY N ' RECORDS COPIED'
RETURN
```

Writing into a PDS

Below is an example of a program writing into a PDS. It asks the user for the records, and, when the record is empty, it asks the user for the member name into which he wants these records stored.

```
ADDRESS TSO
"ALLOC DD(F) DA('MY.PDS') SHR"
RC=$OPEN('F','BPAMO'); SAY 'OPEN RETURN CODE=' RC
DO FOREVER
   SAY 'GIVE A RECORD'; PULL ART
   DO WHILE ART¬=''
      RC=$PUT('F',ART); SAY 'PUT RETURN CODE=' RC
      SAY 'GIVE A RECORD'; PULL ART
   END
   SAY 'GIVE THE MEMBER NAME FOR THESE RECORDS'; PULL MBR
   IF MBR='' THEN DO; RC=$CLOSE('F'); RETURN
                  END
   RC=$STOW('F',MBR,'A',); SAY 'STOW RETURN CODE=' RC
END
```

Processing the members of a PDS

Below is an example of a program handling the members of a PDS. The program asks the user for a member name, and asks for the operation requested for this member.

```
ADDRESS TSO
"ALLOC DD(F) DA('MY.PDS') SHR"
RC=$OPEN('F','BPAMO'); SAY 'OPEN RETURN CODE=' RC
SAY 'MEMBER TO BE PROCESSED ?'; PULL MBR
DO WHILE MBR¬=''
   SAY 'GIVE A MEMBER NAME TO RENAME, OR NOTHING TO DELETE THE MEMBER
   PULL REP
   IF REP='' THEN CALL SUP
   IF REP¬='' THEN CALL REN
```

```
   SAY 'MEMBER TO BE PROCESSED?'; PULL MBR
END
RC=$CLOSE('F')
RETURN
SUP: RC=$STOW('F',MBR,'D'); SAY 'STOW DELETE RET CODE=' RC; RETURN
REN: RC=$STOW('F',MBR,'C',REP); SAY 'STOW CHANGE RET CODE=' RC; RETUR
```

Writing into an ESDS

Below is an example of a program processing a VSAM ESDS. It asks the user for records, and inserts them into the file until the supplied record is empty.

```
/* REXX - WRITES A VSAM ESDS */
ADDRESS TSO
"ALLOC DD(F) DA('MY.ESDS') SHR"
RC=$OPEN('F','VSAM','OUT'); SAY 'CODE RETOUR OUVERTURE=' RC
RC=$MODRPL('F','ADR'); SAY 'CODE RETOUR MODRPL=' RC
N=Ø
SAY 'GIVE A RECORD'
PULL REP
DO WHILE LENGTH(REP)¬=Ø
   RC=$MODRPL('F',,LENGTH(REP)); SAY 'MODRPL RETURN CODE=' RC
   RC=$PUT('F',REP); SAY 'PUT RETURN CODE=' RC
   SAY 'GIVE A RECORD'
   PULL REP
END
RC=$CLOSE('F')
RETURN
```

Reading an ESDS beginning at the end of the file

Below is an example of a program processing a VSAM ESDS. It reads the ESDS beginning at the end of the dataset, and displays the records.

```
/* REXX - BACKWARD READ OF AN ESDS */
ADDRESS TSO
"ALLOC DD(F) DA('MY.ESDS') SHR"
RC=$OPEN('F','VSAM'); SAY 'OPEN RETURN CODE=' RC
RC=$MODRPL('F','ADR,LRD,BWD'); SAY 'MODRPL RETURN CODE=' RC
RC=$POINT('F'); SAY 'POINT RETURN CODE=' RC
F_EOF=Ø
RC=$GET('F'); SAY '1ST GET RETURN CODE=' RC
DO WHILE F_EOF=Ø
   SAY F_RECORD;
   RC=$GET('F')
END
```

```
RC=$CLOSE('F')
RETURN
```

Reading a KSDS sequentially beginning at a key

Below is an example of a program reading a VSAM KSDS. After asking the user for a key, it reads the file beginning at this key, and displays the records. Although, in this example, the key length must be 8, any other key length can be processed using the $SHOWACB instruction (where the KEYLEN parameter allows you to obtain the key length of any VSAM KSDS).

```
/* REXX - SEQUENTIAL READ FROM A KEY IN A KSDS */
ADDRESS TSO
"ALLOC DD(F) DA('MY.KSDS') SHR"
RC=$OPEN('F','VSAM'); SAY 'OPEN RETURN CODE=' RC
CLE='A'
DO WHILE CLE¬=''
   SAY 'GIVE THE KEY'
   PULL CLE
   IF CLE='' THEN LEAVE
   CLE=SUBSTR(CLE!!'        ',1,8)
   RC=$POINT('F',CLE); SAY 'POINT RETURN CODE=' RC
   IF RC=8 & REASON=16 THEN DO
                               SAY '(RECORD NOT FOUND)'; ITERATE
                            END
   F_EOF=Ø
   RC=$GET('F'); SAY '1ST GET RETURN CODE=' RC
   DO WHILE F_EOF=Ø
      SAY F_RECORD
      RC=$GET('F')
   END
END
RC=$CLOSE('F')
RETURN
```

Reading, writing, or deleting records in a KSDS

Below is an example of a program processing a VSAM KSDS. It asks the user for a record key, and asks for the desired operation, namely read, write, or delete.

```
/* REXX - READ, WRITE, DELETE A KSDS, BY KEY */'
ADDRESS TSO
"ALLOC DD(F) DA('MY.KSDS') SHR"
RC=$OPEN('F','VSAM','DIR,IN ,OUT'); SAY 'OPEN RETURN CODE=' RC
```

```
RC=$SHOWACB('F',LCLE,'KEYLEN')
SAY 'THE LENGTH OF THE KEY FOR THIS FILE IS ' LCLE
RC=$SHOWACB('F',DEP,'RKP')
DEP=DEP
SAY 'THIS KEY HAS AN OFFSET OF ' DEP
MINI=LCLE+DEP
RC=$MODRPL('F','DIR'); SAY 'MODRPL RETURN CODE=' RC
SAY 'GIVE THE KEY'
PULL REP
DO WHILE REP¬=''
   IF LENGTH(REP)<MINI THEN REP=REP!!COPIES(' ',MINI-
    LENGTH(REP))
   SAY 'READ (R), DELETE (D) OR WRITE (W) ?'
   PULL TRAIT
   SELECT
      WHEN TRAIT='R' THEN CALL LIRE
      WHEN TRAIT='D' THEN CALL SUPP
      WHEN TRAIT='W' THEN CALL ECRI
      OTHERWISE NOP
   END
   SAY 'GIVE A KEY'
   PULL REP
END
RC=$CLOSE('F')
RETURN
SUPP: /* DELETE THIS RECORD */
   RC=$MODRPL('F','UPD'); SAY 'MODRPL RETURN CODE=' RC
   RC=$GET('F',REP); SAY 'GET RETURN CODE=' RC
   IF RC=Ø THEN SAY 'THIS RECORD IS:'F_RECORD
   RC=$ERASE('F'); SAY 'ERASE RETURN CODE=' RC
   IF RC=Ø THEN SAY 'RECORD HAS BEEN DELETED'
   RETURN
LIRE: /* READ THIS RECORD */
   RC=$MODRPL('F','NUP'); SAY 'MODRPL RETURN CODE=' RC
   RC=$GET('F',REP); SAY 'GET RETURN CODE=' RC
   IF RC=Ø THEN SAY 'THIS RECORD IS:'F_RECORD
   RETURN
ECRI: /* WRITE THIS RECORD */
   SAY '(LENGTH OF THIS RECORD IS' LENGTH(REP) ')'
   RC=$MODRPL('F','NUP',LENGTH(REP)); SAY 'MODRPL RETURN CODE='
    RC
   RC=$PUT('F',REP); SAY 'PUT RETURN CODE=' RC
   RETURN
```

Loading a KSDS

Below is a skeleton of a program loading a KSDS. In this example, the

keys must be in ascending order.

```
/* REXX - LOADING A KSDS */
RC=$OPEN('F','VSAM','OUT')
DO I=1 TO NUMBER_OF_RECORDS
   RC=$MODRPL('F',,length_of_ith_record)
   RC=$PUT('F',record_i)
END
```

Reading a KSDS by generic key

Below is a skeleton of a program reading a KSDS by generic key. The read record is the first one whose key begins with the supplied generic key.

```
/* REXX - READ BY GENERIC KEY */
RC=$OPEN('F','VSAM','DIR')
RC=$MODRPL('F','DIR,GEN',length_of_generic_key)
RC=$GET('F',generic_key)                                        !
```

Reading a KSDS by key higher or equal

Below is a skeleton of a program reading a KSDS by key higher or equal.

```
/* REXX - READING BY KEY >= */
RC=$OPEN('F','VSAM','DIR')
RC=$MODRPL('F','DIR,KGE')
RC=$GET('F',key)
```

Reading a KSDS beginning at the end

Below is an example of a program reading a KSDS sequentially, beginning at the end of the file.

```
/* REXX - BACKWARD READ FOR A KSDS*/
ADDRESS TSO
"ALLOC DD(F) DA('MY.KSDS') SHR"
RC=$OPEN('F','VSAM'); SAY 'OPEN RETURN CODE=' RC
RC=$MODRPL('F','LRD,BWD'); SAY 'MODRPL RETURN CODE=' RC
RC=$POINT('F') /* POSITION AT THE END */
F_EOF=Ø
RC=$GET('F')
DO WHILE F_EOF=Ø
   SAY 'RECORD='F_RECORD
   RC=$GET('F')
```

```
END
RC=$CLOSE('F')
RETURN                                                          !
```

Reading an ESDS or KSDS by RBA

Below is a program reading an ESDS or KSDS by RBA. It asks the user for an RBA, reads the record, and displays it.

```
/* REXX - GET BY RBA(ESDS OR KSDS)*/
ADDRESS TSO
"ALLOC DD(F) DA('MY.ESDS') SHR"
RC=$OPEN('F','VSAM','ADR,DIR'); SAY 'OPEN RETURN CODE=' RC
RC=$MODRPL('F','ADR,DIR'); SAY 'MODRPL RETURN CODE=' RC
SAY 'GIVE A RBA'
PULL REP
DO WHILE REP¬=''
   REP=D2C(REP,4)
   RC=$GET('F',REP)
   IF RC¬=Ø THEN SAY 'GET RETURN CODE=' RC ' REASON=' REASON
   IF RC=Ø THEN SAY F_RECORD
   SAY 'GIVE A RBA'
   PULL REP
END
RC=$CLOSE('F')
RETURN
```

Reading and writing an RRDS

Below is a program processing a VSAM RRDS. It asks the user for the desired operation, namely read or write.

```
/* REXX - READ/WRITE ON A RRDS */'
ADDRESS TSO
"ALLOC DD(F) DA('MY.RRDS') SHR"
RC=$OPEN('F','VSAM','IN ,OUT,DIR'); SAY 'OPEN RETURN CODE=' RC
RC=$MODRPL('F','UPD,DIR',8Ø); SAY 'MODRPL RETURN CODE=' RC
SAY 'READ (R) OR WRITE (W) ?'; PULL REP
DO WHILE REP¬=''
   SELECT
      WHEN REP='R' THEN CALL LECT
      WHEN REP='W' THEN CALL ECRI
      OTHERWISE NOP
   END
   SAY 'READ (R) OR WRITE (W) ?'; PULL REP
END
```

```
RC=$CLOSE('F')
RETURN
LECT:
   SAY "RECORD NUMBER ?"; PULL NUM
   NUM=D2C(NUM,4)
   RC=$MODRPL('F','NUP'); SAY 'MODRPL RETURN CODE=' RC
   RC=$GET('F',NUM)
   IF RC¬=Ø THEN SAY 'GET RETURN CODE=' RC ' REASON=' REASON
   IF RC=Ø THEN SAY 'RECORD='F_RECORD
   RETURN
ECRI:
   SAY "RECORD NUMBER ?"; PULL NUM
   NUM=D2C(NUM,4)
   SAY 'GIVE YOUR RECORD'; PULL ART
   RC=$MODRPL('F','NUP'); SAY 'MODRPL RETURN CODE=' RC
   RC=$PUT('F',ART,NUM); SAY 'PUT RETURN CODE=' RC
   RETURN
```

ERROR MESSAGES

The following error messages may occur:

- $IRX001E $DATE: YEAR < 1583. This means that a date provided to the $DATE function contains a year lower than 1583. The Gregorian calendar began in 1582, and REXXPLUS/MVS does not compute dates for the Julian calendar. Check to see whether you have misused the $DATEREF subroutine.
- $IRX002E $DATE: MONTH <1 OR >12. This is self-explanatory.
- $IRX003E $DATE: DAY IN THE YEAR <1 OR > 365 OR 366. This is self-explanatory.
- $IRX004E $DATE: YEAR < 1601. A date provided to the $DATE function contains a year lower than 1601. 1601 is the first year of the first 400-year cycle after 1582 (the first year of our calendar). Check to see whether you have misused the $DATEREF subroutine.
- $IRX005E $DATE: DATE TOO HIGH. A date provided to the $DATE function contains a year greater than the maximum allowed, which is much greater than 10,000. Check to see whether you have misused the $DATEREF subroutine.

- $IRX006E $DATE: DATE < 0. A date provided to the $DATE function is lower than 1 January 1601.
- $IRX007E $DATE: DAY IN THE WEEK < 1 OR > 7. This is self-explanatory.
- $IRX010E $DATE: DAY < 1 OR > 28 OR 29 OR 30 OR 31. This is self-explanatory.
- $IRX011E $DATE: ARGUMENT 4 <=0 OR > 7. The fourth argument of the $DATE function must contain an expression representing a whole number between one (for Monday) and seven (for Sunday). Exponential notation cannot be used.
- $IRX012E $DATE: ARGUMENT 5 <=0 OR > 4 OR 5. The fifth argument of the $DATE function must contain an expression representing a whole number between one and four or five – there are only four or five Mondays in a month, four or five Tuesdays, and so on. Exponential notation cannot be used.
- $IRX013E $DATE: DATE NOT NUMERIC OR LENGTH <1 OR >8. This is self-explanatory. Exponential notation cannot be used.
- $IRX014E $DATE: UNKNOWN FORMAT. The code provided to the $DATE function for the input or the output format is invalid. See the description for this function (above). Note that the codes must be in uppercase.
- $IRX015E NUMBER OF ARGUMENTS INVALID. This is self-explanatory.
- $IRX016E LENGTH OF DDNAME INVALID. This is self-explanatory.
- $IRX017E TYPE OF OPEN INVALID. The parameter used with the $OPEN function is invalid. See the description for the $OPEN function (above). Note that the parameters must be in uppercase.
- $IRX018E FILE ALREADY OPEN. A file with the same DDNAME is already open.

- $IRX019E GET FOR OUTPUT FILE, OR PUT FOR INPUT FILE. This is self-explanatory.
- $IRX020E UNKNOWN FILE. No file with this DDNAME has been opened.
- $IRX022E RECFM INVALID FOR BPAM. REXXPLUS/MVS can process the PDS with RECFM F, FB, or U, but not with either V or VB.
- $IRX023E VSAM INSTRUCTION USED ON NON-VSAM. This is self-explanatory.
- $IRX024E UNKNOWN KEYWORD FOR VSAM. This is self-explanatory.
- $IRX027E $SHOWACB: 4TH ARGUMENT NOT D(ATA) NOR I(NDEX). This is self-explanatory.
- $IRX028E $POINT: WRONG PARAMETER. The second argument of the $POINT function is invalid. For a VSAM RRDS, it must be a four-byte binary number, representing a RRN.
- $IRX029E BLDL ON NON-BPAM, OR STOW ON NON-BPAM, OR BPAM INPUT. This is self-explanatory.
- $IRX030E ERROR AT OPEN. This is self-explanatory.
- $IRX031E GET BEYOND END OF FILE. This is self-explanatory.
- $IRX032E GET ON BPAM FILE WITHOUT PREVIOUS FIND. You must execute a $FIND function before you can read a PDS.
- $IRX033E GET RRN OR RBA, LNG OF RRN OR RBA ¬4. This is self-explanatory.
- $IRX034E PUT WITH RRN ON NON-RRDS, OR W/O RRN ON RRDS. This is self-explanatory.
- $IRX035E FIND ON NON-BPAM FILE. This is self-explanatory.
- $IRX036E INVALID MEMBER NAME. This is self-explanatory.

- $IRX037E $STOW: ERROR TYPE OF STOW (3RD ARGUMENT). This is self-explanatory.
- $IRX038E PUT WITH KEY RRDS: KEY NOT NUMERIC. This is self-explanatory.
- $IRX039E $STOW: MEMBER NAME IS INVALID. This is self-explanatory.
- $IRX040E $RPL: RPL NUMBER IS INVALID. The RPL number must be a whole number between zero and four.
- $IRX041E $MODRPL: LENGTH (3RD ARGUMENT) NOT NUMERIC. This is self-explanatory.
- $IRX042E $OBTAIN SEEK: LENGTH OF SEEK ADDRESS ¬= 5. This is self-explanatory.
- $IRX043E $D2P: ARG NOT WHOLE NUMBER, OR TOO LONG. The argument in $D2P must be a whole number less than 32-bytes long.
- $IRX044E $P2D: ARGUMENT IS TOO LONG. The argument in $P2D must have a length less than or equal to 17 bytes.
- $IRX048E ERROR WHEN STORING IN REXX VARIABLE. This is self-explanatory. The most frequent reason is a lack of memory.
- $IRX051E FLOAT. WITH LNG¬=4 OR DOUBLE FLOAT. WITH LNG¬=8. This is self-explanatory.
- $IRX052E $SHIFT: STRING LENGTH IS > 32 BYTES. This is self-explanatory.
- $IRX053E $SHIFT: ARG 2 NOT NUMERIC, OR <0 OR >8. This is self-explanatory.
- $IRX055E CONVERSION OF FLOATING POINT: INVALID NUMBER. Data is not numeric, or the number is too large.
- $IRX056E $CLOCKDT: ARGUMENT LENGTH ¬=8. This is self-explanatory.

- $IRX057E $OPTION: UNKNOWN OPTION. This is self-explanatory.
- $IRX059E $DATE: 4TH ARG TOO LONG OR NOT NUMERIC. This is self-explanatory.

PROGRAMMING NOTES

Searching for a string in a file

In this version, you can't say 'GET the next record which has, or which does not have, this string in this column', but the CSECT SY751, in $IRXFI1, is almost ready for this ...

WORKEXT_USERFIELD

$IRX is the main table in REXXPLUS/MVS. This table is pointed to by WORKEXT_USERFIELD, in the REXX work block extension. When I wrote REXXPLUS/MVS, some years ago, the Name/Token service (which appears in MVS/SP 4.2.2 and subsequent releases) did not exist. Now, however, it offers a standard way of controlling an anchor point. Nevertheless, my method works.

Floating point data

Since it is not obvious how to convert an extended decimal number with a large exponent (eg 12.3E+25) to (or from) an internal binary floating point number, I used the $IRXCNV2 COBOL program.

IRXFLOC

IRXFLOC is a common interface between IBM REXX and REXXPLUS

```
IRXFLOC  CSECT
IRXFLOC  AMODE ANY
IRXFLOC  RMODE ANY
*********************************************************************
* IRXFLOC. AUTHOR: PATRICK LELOUP.
*********************************************************************
```

```
         DC    CL8'IRXFPACK'   FIXED BY IBM
         DC    FL4'24'         LNG OF HEADER
         DC    AL4(NBENT2)     NB OF ENTRIES
         DC    FL4'Ø'
         DC    FL4'32'         LNG OF 1 ENTRY
NBENT1   EQU   *
$GET     MACFD
$PUT     MACFD
$GETA    MACFD
*PUTA    MACFD
$OPEN    MACFD
$CLOSE   MACFD
$POINT   MACFD
$STOW    MACFD
$FIND    MACFD
$MODRPL  MACFD
$SHOWACB MACFD
$SHOWRPL MACFD
$ERASE   MACFD
$RPL     MACFD
$DATE    MACFD
$DATEREF MACFD
$LOCATE  MACFD
$OBTAIN  MACFD
$D2P     MACFD
$P2D     MACFD
$SF2D    MACFD
$DF2D    MACFD
$D2SF    MACFD
$D2DF    MACFD
$CLOCK   MACFD
$CLOCKDT MACFD
$SHIFT   MACFD
$OPTION  MACFD
NBENT2   EQU   (*-NBENT1)/32   NB OF ENTRIES
**********
* EACH FUNCTION MUST LOAD R3 WITH THE MODULE NUMBER TO LOAD: 1, 2,
* 3 etc. (SEE THE TABMODU TABLE: INSERT THE NAME OF THE MODULE TO
* LOAD IN THE RIGHT PLACE). TWO OR MORE DIFFERENT FUNCTIONS MAY HAVE
* THE SAME MODULE. EACH FUNCTION MUST LOAD R7 WITH A FUNCTION NUMBER
* WHICH MUST BE UNIQUE FOR A GIVEN MODULE.
$DATE    MACF  MODULE=1,FUNCTION=Ø    MODULE $IRXDAT (DATES)
$DATEREF MACF  MODULE=1,FUNCTION=4    "
$CLOSE   MACF  MODULE=2,FUNCTION=Ø    MODULE $IRXFI1 (FILES)
$ERASE   MACF  MODULE=2,FUNCTION=4    "
$FIND    MACF  MODULE=2,FUNCTION=8    "
$GET     MACF  MODULE=2,FUNCTION=12   "
$GETA    MACF  MODULE=2,FUNCTION=16   "
$MODRPL  MACF  MODULE=2,FUNCTION=2Ø   "
```

```
$OPEN    MACF  MODULE=2,FUNCTION=24   "
$POINT   MACF  MODULE=2,FUNCTION=28   "
$PUT     MACF  MODULE=2,FUNCTION=32   "
$PUTA    MACF  MODULE=2,FUNCTION=36   "
$SHOWACB MACF  MODULE=2,FUNCTION=4Ø   "
$SHOWRPL MACF  MODULE=2,FUNCTION=44   "
$STOW    MACF  MODULE=2,FUNCTION=48   "
$RPL     MACF  MODULE=2,FUNCTION=52   "
$LOCATE  MACF  MODULE=2,FUNCTION=56   "
$OBTAIN  MACF  MODULE=2,FUNCTION=6Ø   "
$D2P     MACF  MODULE=3,FUNCTION=Ø    MODULE $IRXCNV (CONVERSION)
$P2D     MACF  MODULE=3,FUNCTION=4    "
$SF2D    MACF  MODULE=3,FUNCTION=8    "
$DF2D    MACF  MODULE=3,FUNCTION=12   "
$D2SF    MACF  MODULE=3,FUNCTION=16   "
$D2DF    MACF  MODULE=3,FUNCTION=2Ø   "
$SHIFT   MACF  MODULE=3,FUNCTION=24   "
$CLOCK   MACF  MODULE=4,FUNCTION=Ø    MODULE $IRXTDC (TOD CLOCK)
$CLOCKDT MACF  MODULE=4,FUNCTION=4    "
$OPTION  MACF  MODULE=5,FUNCTION=Ø    MODULE $IRXOPT
************************************************************
COMMON   DS    ØH
         BASR  R1Ø,Ø           INIT BASE REGISTER
         USING *,R1Ø
         LR    R9,RØ           R9->ENVBLOCK
         GETMAIN R,LV=SAUVL
         ST    R1,8(,R13)
         ST    R13,4(,R1)
         LR    R13,R1
         LR    R6,R9           R6->ENVBLOCK
         USING ENVBLOCK,R6
         L     R6,ENVBLOCK_WORKBLOK_EXT R6->WORK BLOCK EXTENSION
         DROP  R6
         USING WORKBLOK_EXT,R6
         L     R8,WORKEXT_USERFIELD     R8->USER FIELD
         USING $IRX,R8
         LTR   R8,R8           OUR MAIN TABLE EXISTS ?
         BNZ   L2Ø             YES, B
         GETMAIN R,LV=$IRXL,SP=Ø,LOC=BELOW NO, GETMAIN IT
*              BELOW BECAUSE OF IBM MACROS GET, PUT, ETC
         ST    R1,WORKEXT_USERFIELD STORE ADDRESS OF OUR AREA
         LR    R8,R1           R8->OUR AREA
         LR    R14,R8          R14->AREA TO BE ZEROED
*                              INITIALIZE OUR MAIN TABLE $IRX
         LA    R15,$IRXL       R15:=LNG OF THE TABLE TO ZERO
         XR    R4,R4           ADDRESS OF EMISSION:=Ø
         XR    R5,R5           LNG OF EMISSION:=Ø
         MVCL  R14,R4          CLEAR THE TABLE
         ST    R8,$IRX$IRX     STORE ADDRESS OF THE TABLE
```

```
         LA    R15,$IRXL       R15:=LNG OF TABLE
         ST    R15,$IRXLNG     STORE INTO ITSELF
         MVC   $IRXID,ID       STORE ID
         ST    R9,$IRXENVB     STORE ADDR ENV BLOCK GIVEN BY REXX
         ST    R6,$IRXWBE      STORE ADR WORK BLOCK EXT UNDER WHICH
*                              THE TABLE $IRX WAS CREATED
         DROP  R6
*              STORE NUMBER OF ENTRIES TO BE FREED BY $IRXTERM:
         LA    R15,$IRXNPAL    R15:=ADDR OF LAST ENTRY + 1
         LA    RØ,$IRXTER      RØ:=ADDR 1ST ENTRY
         SR    R15,RØ          R15:=NO OF BYTES IN 1 ENTRY
         SRL   R15,3           /8=>R15:=NO OF ENTRIES TO FREE
         ST    R15,$IRXNPAL    STORE
*              INITS FOR IRXEXCOM (REXX VARIABLES ASSIGNMENT)
         MVC   $IRCOMNM,=CL8'IRXEXCOM' REXX NAME OF VAR MANIP MODULE
         MVI   $IRCODE,C'S'    CODE FOR "SET VARIABLE"
         LA    R15,$IRCOMNM    R15->1ST PARM FOR IRXEXCOM
         ST    R15,$IRCOMNA    STORE
         LA    R15,$IRNEXT     R15->4TH PARM FOR IRXEXCOM
         ST    R15,$IRADSHV    STORE
         OI    $IRADSHV,X'8Ø'  SET LEFT BIT TO 1 (LAST PARM)
*              INITS FOR IRXRLT (REQUEST FOR A LARGER EVALBLOCK)
         MVC   $IRLTG,=CL8'GETBLOCK' STORE FUNCTION NAME
         LA    R15,$IRLTG      R15->1ST PARM (->'GETBLOCK')
         ST    R15,$IRLTGA     STORE  E
         LA    R15,$IRLTBA     R15->2ND PARM (->Ø, RETURNS BLOCK
                                ADDR)
         ST    R15,$IRLTAA     STORE
         LA    R15,$IRVALL     R15->3RD PARM (LNG OF DESIRED RESULT)
         ST    R15,$IRLTLA     STORE
         OI    $IRLTLA,X'8Ø'   SET LEFT BIT TO 1 (LAST PARM)
*              CREATE THE TABLE OF MODULE ADDRESSES
         GETMAIN R,LV=MODUL,SP=Ø GETMAIN IT
         ST    R1,$IRXMODU     STORE
         LA    R15,MODUL       R15:=LNG OF THE MODULES TABLE
         ST    R15,$IRXMODL    STORE IN TABLE $IRX
         LR    R14,R1          R14->TABLE TO BE ZEROED
         LA    R15,MODUL       R15:=LNG OF THE TABLE
         XR    R4,R4           ADDR OF EMISSION:=Ø
         XR    R5,R5           LNG OF EMISSION:=Ø
         MVCL  R14,R4          CLEAR THE TABLE
L2Ø      EQU   *               HERE, R8->MAIN TABLE ($IRX)
         CLC   $IRXID,ID       IS THIS TABLE OUR TABLE ?
         BNE   ABEND1          NO, B ABEND
L21      EQU   *               COUNT THE NUMBER OF ARGUMENTS
         L     R1,4(,R13)      R1:=...
         L     R1,24(,R1)      R1 AT THE ENTRY OF IRXFLOC
         USING EFPL,R1
         L     R1,EFPLARG      R1->LIST OF ARGUMENTS
```

```
         USING ARGUM,R1
         XR    R15,R15          R15 WILL BE THE NUMBER OF ARG
ARGC     EQU   *
         CLC   ARGUM1P,FINARG   END OF ARGUMENTS ?
         BE    ARGC5            YES, B
         LA    R1,ARGUML(,R1)   ->NEXT ARGUMENT
         LA    R15,1(,R15)      +1 ARGUMENT
         B     ARGC             LOOP
         DROP  R1
ARGC5    EQU   *
         STC   R15,$IRNBARG     STORE NUMBER OF ARGUMENTS
*              LOAD EVENTUALY, AND CALL CONCERNED MODULE
         L     R4,$IRXMODU      R4->TABLE OF MODULES ADDRESSES
         BCTR  R3,Ø             -1, SO RELATIVE TO Ø
         SLL   R3,2             NUM FCT * 4
         L     R15,Ø(R3,R4)     R15->ENTRY FOR THIS MODULE
         LTR   R15,R15          ADDR=Ø ?
         BNE   L5Ø              NO, B CALL THIS MODULE
         LA    R15,TABMODU      YES, MUST LOAD. R15->MODULES NAMES
         LR    R14,R3           R14:=R3...
         SLL   R14,1            R3*2 BECAUSE 1 ENTRY IS 8 BYTES LONG
         LA    R2,Ø(R15,R14)    R2->NAME OF THE MODULE TO LOAD
         LOAD  EPLOC=(R2)       LOAD THE MODULE
         ST    RØ,Ø(R3,R4)      STORE MODULE ADDRESS
         LR    R15,RØ           R15->MODULE
L5Ø      EQU   *
         L     R1,4(,R13)       R1->PREVIOUS SAVE AREA
         L     RØ,2Ø(,R1)       RØ AT ENTRY OF IRXFLOC
         L     R1,24(,R1)       R1 AT ENTRY OF IRXFLOC
         LR    R2,R7            R2:=NUM OF FCT IN MODULE
         STC   R2,$IRFONC       STORE CODE OF FUNCTION
* WE CALL MODULE WITH       RØ = RØ AT ENTRY OF IRXFLOC,
*                           R1 = R1 AT ENTRY OF IRXFLOC,
*                           R2 = NUM OF FUNCTION IN MODULE,
*                           (ALSO STORED IN $IRFONC DS C)
*                           R8 -> TABLE $IRX
         BASR  R14,R15          CALL THE MODULE
* AT EXIT   OF MODULE,        R15=Ø OR R15=NUM OF ERROR MESSAGE
*                             R1=Ø IF ALPHA,
*                                4 IF NUMERIC, TO BE NORMALIZED
         LTR   R2,R15           RETURN CODE Ø ?
         BNZ   L6Ø              NO, B
         B     *+4(R1)          B ACCORDING TO NORMALIZ. TO BE DONE
         B     FIN              B IF NOTHING TO DO
         B     NORM             B IF NORMALIZATION REQUIRED
* NORMALIZATION OF A NUMERIC VALUE.
* NUMERIC VALUES MUST BE PROVIDED SIGNLESS EXTENDED, OR WITH
* OVERPUNCHED SIGN ON LAST BYTE, BUT W/O A LEADING SIGN
NORM     EQU   *                NORMALIZATION OF A NUMERIC VALUE
```

```
         L     R1,4(,R13)      R1:=...
         L     R1,24(,R1)      R1 AT ENTRY OF IRXFLOC
         USING EFPL,R1
         L     R4,EFPLEVAL
         L     R4,Ø(,R4)       R4->EVALBLOCK
         USING EVALBLOCK,R4
         L     R1,EVALBLOCK_EVLEN R1:=LNG
         LTR   R1,R1           LNG<=Ø?
         BNP   NORM99          YES, NOTHING TO DO
         CH    R1,=H'32'       LNG > 32 ?
         BH    NORM99          YES, NOTHING TO DO
         BCTR  R1,Ø            LNG -1 FOR EX
         EX    R1,NORMMVC1     STORE NUMBER IN WORK AREA
         LA    R14,$IRMFE(R1)  R14->LAST BYTE OF THE NUMBER
         XC    $IRXØ,$IRXØ     CLEAR WORK AREA FOR SIGN
         MVZ   $IRXØ,Ø(R14)    STORE SIGN IN WORK AREA
         NI    $IRDRAP1,X'FF'-$IRDR8Ø SET "WE HAVE A +"
         CLI   $IRXØ,X'DØ'     IS NUMBER < Ø ?
         BE    NORM1Ø          YES, B
         CLI   $IRXØ,X'BØ'     IS NUMBER < Ø ?
         BNE   NORM2Ø          NO, B
NORM1Ø   EQU   *               NUMBER IS < Ø
         OI    $IRDRAP1,$IRDR8Ø SET "WE HAVE A -"
NORM2Ø   EQU   *
         OI    Ø(R14),X'FØ'    FORCE LAST BYTE READABLE
*              HERE, $IRDRAP1  CONTAINS X'8Ø' IF <Ø
*              $IRMFE CONTAINS THE EXTENDED NUMBER TO BE NORMALIZED
         LA    R2,$IRMFE       R2->1ST BYTE TO BE PROCESSED
         L     R1,EVALBLOCK_EVLEN R1:=LNG TO BE PROCESSED (>Ø)
NORM3Ø   EQU   *               LOOP FOR SEARCHING OF 1ST NON ZERO
         CLI   Ø(R2),C'Ø'      IS IT A Ø ?
         BNE   NORM5Ø          NO, 1ST ZERO FOUND
         LA    R2,1(,R2)       YES, NEXT BYTE
         BCT   R1,NORM3Ø       LOOP
         BCTR  R2,Ø            WE HAD ONLY C'Ø'. POINT TO LAST C'Ø'
         LA    R1,1            AND FORCE LNG:=1
NORM5Ø   EQU   *
* HERE, R2->1ST NON Ø IN $IRMFE, R1=REMAINING LNG TO BE PROCESSED (>Ø)
         LR    R15,R1          R15:=LNG OF RESULT
         LA    R14,EVALBLOCK_EVDATA R14->AREA TO BE FILLED
         TM    $IRDRAP1,$IRDR8Ø IS NUMBER <Ø ?
         BNO   NORM6Ø          NO, B
         MVI   EVALBLOCK_EVDATA,C'-' YES, STORE "-" SIGN
         LA    R14,1(,R14)     WE MUST FILL 1 BYTE LATER
         LA    R15,1(,R15)     THE RESULT WILL HAVE 1 MORE BYTE (-
                                SIGN)
NORM6Ø   EQU   *
         ST    R15,EVALBLOCK_EVLEN STORE LNG OF RESULT
         BCTR  R1,Ø            LNG - 1 FOR EX
```

```
         EX    R1,NORMMVC2     STORE RESULT
NORM99   EQU   *
         XR    R15,R15         CLEAR RETURN CODE FOR REXX
         B     FIN
NORMMVC2 MVC   Ø(R1-R1,R14),Ø(R2) STORE $IRMFE IN EVDATA
NORMMVC1 MVC   $IRMFE(R1-R1),EVALBLOCK_EVDATA
         DROP  R4
L6Ø      EQU   *               SEND THE ERROR MESSAGE (R2)
         LINK  EP=$IRXMSG      SEND THE MESSAGE (R2)
         LA    R15,4           WRONG RETURN CODE FOR REXX
FIN      EQU   *
         LR    R3,R15          SAVE R15
         L     R2,4(,R13)
         FREEMAIN R,LV=SAUVL,A=(R13)
         LR    R13,R2
         LR    R15,R3          RESTORE R15
         L     R14,12(,R13)    RESTORE R14
         LM    RØ,R12,2Ø(R13)  RESTORE REGS
         BR    R14             RETURN
ABEND1   ABEND 4ØØ2            ABEND: MAIN TABLE IS NOT $IRX
         LTORG
*DATA*************************************************************
TABMODU  DS    ØF              TABLE OF MODULES NAMES
         DC    CL8'$IRXDAT '   MODULE 1: DATES PROCESSING
         DC    CL8'$IRXFI1 '   MODULE 2: DATASET PROCESSING
         DC    CL8'$IRXCNV '   MODULE 3: CONVERSIONS
         DC    CL8'$IRXTDC '   MODULE 4: TOD CLOCK
         DC    CL8'$IRXOPT '   MODULE 5: $OPTION
ID       DC    CL4'$IRX'       ID OF TABLE $IRX
FINARG   DC    X'FFFFFFFF'     FOR THE END OF ARGUMENTS
*DSECTS***********************************************************
         COPY  $IRXDSEC
SAUV     DS    18F,18F,18F,18F SAVE AREA LEVEL Ø, 1, 2, 3
SAUVL    EQU   *-SAUV
MODU     DSECT                 MODULES ADDRESSES. PTD TO BY $IRXMODU
         DS   F                MODULE  1: $IRXDAT
         DS   F                MODULE  2: $IRXFI1
         DS   F                MODULE  3: $IRXCNV
         DS   F                MODULE  4: $IRXTDC
         DS   F                MODULE  5: $IRXOPT
MODUL    EQU  *-MODU           LNG OF TABLE
         IRXEFPL
         IRXEVALB
         IRXENVB
         IRXWORKB
         END
```

MACF

MACF is a macro for IRXFLOC

```
         MACRO
&NOM     MACF  &MODULE=,&FONCTION=
.* MACRO FOR EACH FUNCTION IN IRXFLOC
&NOM     DS    OH
         STM   R14,R12,12(R13)
         LA    R3,&MODULE
         LA    R7,&FUNCTION
         USING &NOM,R15
         B     COMMUN
         DROP  R15
         MEXIT
         MEND
```

MACFD

MACFD is a macro for IRXFLOC.

```
MACRO
&NOM     MACFD &FCT=
.* MACRO FOR EACH FUNCTION IN IRXFLOC: HEADER
.* EXAMPLES:
.* $FCT1  MACFD
.*        IN THIS CASE, NAME OF REXX FUNCTION IS $FCT1
.* $FCT1  MACFD FCT=FUNCT1
.*        IN THIS CASE, NAME OF REXX FUNCTION IS FONCT1
&A       SETC  '&FCT'
         AIF   ('&A' NE '').L1
&A       SETC  '&NOM'
.L1      ANOP
         DC    CL8'&A'
         DC    AL4(&NOM)
         DC    FL4'Ø'
         DC    CL8' '
         DC    CL8' '
         MEXIT
         MEND
```

A24

The A24 macro sets 24-bit addressing mode.

```
MACRO
         A24   &REG
```

```
&R       SETC  '&REG'
         AIF   ('&REG' NE '').L1
&R       SETC  '14'
.L1      ANOP
         LA    &R,*+6          ADDRESS WITH BIT Ø = Ø
         BSM   Ø,&R            SET 24 BIT ADDRESSING MODE
         MEXIT
         MEND
```

A31

The A31 macro sets 31-bit addressing mode.

```
MACRO
         A31   &REG
&R       SETC  '&REG'
         AIF   ('&REG' NE '').L1
&R       SETC  '14'
.L1      ANOP
&ETIQ    SETC  '&SYSNDX'
         L     &R,A&ETIQ       ADDRESS WITH BIT Ø = 1
         BSM   Ø,&R            SET 31 BIT ADDRESSING MODE
A&ETIQ   DC    A(*+4+X'80000000')
         MEXIT
         MEND
```

Editor's note: This article will be continued in the next and subsequent issues.

Patrick Leloup
Systems Engineer
Credit Agricole de Loire Atlantique (France)

Managing VSAM from TSO ISPF

Managing VSAM from TSO ISPF can present a number of problems. The programs presented here are designed to make life a little easier. They include:

- A simple tool for browsing a VSAM file.
- A simple tool for doing a listcat for a VSAM file.
- A REXX program to create a DASD report from a file created by DCollect.
- A back-up utility.

BROWSING A VSAM FILE

Although the dataset listing (DSLIST) displayed by TSO ISPF Option 3.4 provides a generous set of line commands, it also allows CLIST or REXX to execute against a listed dataset.

The tool presented below is easy to use, and prevents you having to use a non-IBM product simply to browse a VSAM file. REXX enables you to invoke the BRSE command either under TSO or in a 3.4 list under ISPF.

Browse tool

```
/* REXX */
X = MSG("OFF")
IF ARG()  = Ø   THEN CALL MAP
                ELSE PARSE ARG NUMBER NOMFICH .
/* SAY NAME OF FILE: ' NOMFICH NUMBER*/
IF DATATYPE(NUMBER,'N') = Ø THEN DO
                                     NOMFICH = NUMBER
                                     NUMBER ='*'
                                END
"FREE F(SORTIN)"
"FREE F(SORTOUT)"
"FREE F(DDSORT) DELETE"
"FREE F(SYSIN)"
"ALLOC FILE(SYSIN)  DA('PROG69.OLIVIER.CNTL(PARAM)')"
"ALLOC FILE(SORTIN) DA("NOMFICH") SHR"
```

```
"ALLOC FILE(SORTOUT) DA('"USERID()"."TEMP"')",
             "UNIT(SYSDA) SPACE(5,5) TRACKS NEW CATALOG"
"ALLOC FILE(SYSOUT)  DA(*)"
"CALL 'SYS1.LPALIB(SORT)'"
"ALLOC DS('"USERID()"."TEMP"') DD(DDSORT) OLD REUSE"
"EXECIO" NUMBER "DISKR DDSORT (STEM DATA. FINIS"
DO I = 1 TO DATA.Ø
    SAY DATA.I
END
"FREE F(SORTIN)"
"FREE F(SORTOUT)"
"FREE F(DDSORT) DELETE"
"FREE F(SYSIN)"
X = MSG("ON")
EXIT
MAP:PROCEDURE
  SAY ' ENTER THE FILE NAME BETWEEN THE QUOTES '
  PARSE EXTERNAL NUMBER
  PARSE EXTERNAL NOMFICH
RETURN
```

DOING A LISTCAT FOR A VSAM FILE

The tool presented here allows you to do a LISTCAT for a VSAM file. It is a simple ISPF windowed dialog, and the output is directed to a file which is browsed. The dialog allows you to invoke the LISTCAT command either under TSO or in a 3.4 list under ISPF.

LISTCAT tool

```
/* REXX */
/* ****************************************************************/
/* * REXX procedure . LISTCAT  of files               .           */
/* *                . Listing in Userid.LISTCAT     .             */
/* ****************************************************************/
If Arg()  = Ø   Then Call MAP
                Else Parse Arg file
If file = '' then Exit 4
FCode=SYSDSN("'"Userid()".LISTCAT'")
If FCode = 'OK' Then
  Do
    Address TSO "DELETE '"userid()".LISTCAT' NONVSAM"
  End
Address TSO "ALLOCATE DATASET('"userid()".LISTCAT') FILE(Exit)",
                    "DSORG(PS) SPACE(1,Ø) TRACKS RELEASE",
                    "LRECL(125) BLKSIZE(629) RECFM(V,B)",
                    "MGMTCLAS(INTERIM) STORCLAS(BASE)",
                    "NEW CATALOG"
```

```
Address TSO     "LISTCAT ENTRY("file") ALL OUTFILE(Exit)"
Address ISPEXEC "VIEW DATASET('"userid()".LISTCAT') MACRO(MACLCAT)"
Address TSO     "FREE FILE(Exit) DELETE"
Exit rc
MAP:
  Say ' Enter the file NAME beween QUOTES '
  Parse External file
Return
```

CREATING A DASD REPORT FROM A FILE CREATED BY DCOLLECT

A frequent problem in performance reporting and monitoring is that of manipulating and managing the vast amounts of data produced by SMF, RMF, and third-party reporters.

Various data reduction and reporting tools have evolved over the years to tackle this problem, perhaps one of the most widely installed being Barry Merrill's SAS/MXG. The book(s) and software provide a basic set of SAS routines that reformat raw SMF data into SAS files (databases). Sets of reports and trending macros are also provided.

But the REXX program presented below removes the need for either SAS or SMF files, thus saving money. The code was developed in an MVS520 environment. It reads a file created by the IBM DCollect utility and creates a DASD report.

REXX DCollect program

```
REXX
        /*************************************************************/
        /* DFSMS13 : 16/Ø6/97 MODIF PHILG                            */
        /*************************************************************/
        /* HERE IS AN EXAMPLE TSO/E REXX EXEC THAT READS RECORDS     */
        /* PRODUCED BY DCOLLECT AND CREATES A SIMPLE REPORT.         */
        /*                                                           */
        /* THERE ARE MANY WAYS OF PROCESSING THIS DATA, REXX IS      */
        /* JUST ONE OF THEM.                                         */
        /*                                                           */
        /* FOR MORE EXAMPLES OF REPORTS THAT CAN BE CREATED FROM     */
        /* THESE RECORDS, SEE THE FOLLOWING PUBLICATION:             */
        /*                                                           */
        /* DFSMS PLANNING REPORTING WITH DCOLLECT (GG24-3540-ØØ).    */
        /*                                                           */
        /* NOTE: THE REXX LANGUAGE IS A GENERAL-USE PROGRAMMING      */
        /*       INTERFACE.                                          */
        /*************************************************************/
```

```
/*************************************************************/
/* OPEN THE INPUT FILE (CONTAINING DCOLLECT RECORDS)         */
/*************************************************************/
 INNAME = "'EXPL69.HSM.DCOLLECT'"
 "ALLOC F(INFILE) DA("INNAME") SHR"
 IF RC¬= Ø THEN DO
    SAY  'ALLOCATION OF ('INNAME') FAILED'
    EXIT 8
    END

/*************************************************************/
/* OPEN THE OUTPUT FILE (CONTAINING YOUR REPORT)             */
/*************************************************************/
 OUTNAME = "'EXPL69.HSM.REPORT'"
 "ALLOC F(OUTFILE) DA("OUTNAME")"
 IF RC¬= Ø THEN DO
    SAY  'ALLOCATION OF ('OUTNAME') FAILED'
    EXIT 12
    END

/*************************************************************/
/* PROCESS EACH RECORD UNTIL END-OF-FILE REACHED             */
/*************************************************************/
 EOF = 'NO'
 DO WHILE EOF='NO'
   "EXECIO 1 DISKR INFILE"
   IF RC¬= Ø THEN
      EOF = 'YES'
   ELSE DO
      PARSE PULL RECORD
/*************************************************************/
/* DETERMINE THE RECORD TYPE (M,B,C,T)                       */
/*************************************************************/
      DCURCTYP=SUBSTR(RECORD,5,2)
         SELECT
/*************************************************************/
/* PROCESS MIGRATED DATASET INFORMATION RECORD               */
/*************************************************************/
         WHEN(DCURCTYP='M ') THEN DO
                             CALL UMRECORD
                             CALL DISPLAYUM
                             END
/*************************************************************/
/* PROCESS BACKUP VERSION INFORMATION RECORD                 */
/*************************************************************/
         WHEN(DCURCTYP='B ') THEN DO
                             CALL UBRECORD
                             CALL DISPLAYUB
                             END
```

```
/*************************************************************/
/* PROCESS DASD CAPACITY PLANNING RECORD                     */
/*************************************************************/
         WHEN(DCURCTYP='C ') THEN DO
                             CALL UCRECORD
                             CALL DISPLAYUC
                             END
/*************************************************************/
/* PROCESS TAPE CAPACITY PLANNING RECORD                     */
/*************************************************************/
         WHEN(DCURCTYP='T ') THEN DO
                             CALL UTRECORD
                             CALL DISPLAYUT
                             END
/*************************************************************/
/* DO NOT PROCESS OTHER RECORD TYPES                         */
/*************************************************************/
         OTHERWISE SAY "UNKNOWN RECORD TYPE="DCURCTYP
         END
      END
 END
 "FREE ALL"
 EXIT
/*************************************************************/
/* PROCESS MIGRATED DATASET INFORMATION RECORD -             */
/*   CONVERT THE RECORD INTO INDIVIDUAL FIELD VARIABLES.     */
/*************************************************************/
 UMRECORD:
       UMDSNAM = SUBSTR(RECORD,25,44)
       UMFLAG1 = BITSTR(SUBSTR(RECORD,69,1))
       UMLEVEL = SUBSTR(UMFLAG1,1,2)
       UMCHIND = SUBSTR(UMFLAG1,3,1)
       UMDEVCL = SUBSTR(RECORD,70,1)
       UMDSORG = C2X(SUBSTR(RECORD,71,2))
       UMDSIZE = C2D(SUBSTR(RECORD,73,4))
       UMTIME  = C2X(SUBSTR(RECORD,77,4))
       UMDATE = C2X(SUBSTR(RECORD,81,4))
       UMDATCL = SUBSTR(RECORD,87,30)
       UMSTGCL = SUBSTR(RECORD,119,30)
       UMMGTCL = SUBSTR(RECORD,151,30)
       UMRECRD = C2X(SUBSTR(RECORD,181,1))
       UMRECOR = C2X(SUBSTR(RECORD,182,1))
       UMBKLNG = C2D(SUBSTR(RECORD,183,2))
       UMFLAG2 = BITSTR(SUBSTR(RECORD,185,1))
       UMRACFD = SUBSTR(UMFLAG2,1,1)
       UMGDS   = SUBSTR(UMFLAG2,2,1)
       UMREBLK = SUBSTR(UMFLAG2,3,1)
       UMPDSE  = SUBSTR(UMFLAG2,4,1)
       UMSMSM  = SUBSTR(UMFLAG2,5,1)
       UMNMIG  = C2D(SUBSTR(RECORD,187,2))
```

```
      UMALLSP = C2D(SUBSTR(RECORD,189,4))
      UMUSESP = C2D(SUBSTR(RECORD,193,4))
      UMRECSP = C2D(SUBSTR(RECORD,197,4))
      UMCREDT = C2X(SUBSTR(RECORD,201,4))
      UMEXPDT = C2X(SUBSTR(RECORD,205,4))
      UMLBKDT = C2X(SUBSTR(RECORD,209,8))
      UMLRFDT = C2X(SUBSTR(RECORD,217,4))
 RETURN
/*************************************************************/
/* PROCESS MIGRATED DATASET INFORMATION RECORD -             */
/*   WRITE THE FORMATTED FIELDS TO YOUR REPORT               */
/*************************************************************/
 DISPLAYUM:
 PUSH "M:DSN="UMDSNAM " MIGLVL="UMLEVEL " CHANGED="UMCHIND ,
      " MVOL="UMDEVCL "  DATE MIG="UMDATE "  NB MIG="UMNMIG ,
      " MGMTCL="LEFT(UMMGTCL,10)
 "EXECIO 1  DISKW OUTFILE"
 RETURN
/*************************************************************/
/* PROCESS BACKUP VERSION INFORMATION RECORD                 */
/*   CONVERT THE RECORD INTO INDIVIDUAL FIELD VARIABLES.     */
/*************************************************************/
 UBRECORD:
      UBDSNAM = SUBSTR(RECORD,25,44)
      UBFLAG1 = BITSTR(SUBSTR(RECORD,69,1))
      UBDEVCL = SUBSTR(RECORD,70,1)
      UBDSORG = C2X(SUBSTR(RECORD,71,2))
      UBDSIZE = C2D(SUBSTR(RECORD,73,4))
      UBTIME  = C2X(SUBSTR(RECORD,77,4))
      UBDATE = C2X(SUBSTR(RECORD,81,4))
      UBDATCL = SUBSTR(RECORD,87,30)
      UBSTGCL = SUBSTR(RECORD,119,30)
      UBMGTCL = SUBSTR(RECORD,151,30)
      UBRECRD = C2X(SUBSTR(RECORD,181,1))
      UBRECOR = C2X(SUBSTR(RECORD,182,1))
      UBBKLNG = C2D(SUBSTR(RECORD,183,2))
      UBFLAG2 = BITSTR(SUBSTR(RECORD,185,1))
      UBRACFD = SUBSTR(UBFLAG2,1,1)
      UBGDS   = SUBSTR(UBFLAG2,2,1)
      UBREBLK = SUBSTR(UBFLAG2,3,1)
      UBPDSE  = SUBSTR(UBFLAG2,4,1)
      UBSMSM  = SUBSTR(UBFLAG2,5,1)
      UBALLSP = C2D(SUBSTR(RECORD,189,4))
      UBUSESP = C2D(SUBSTR(RECORD,193,4))
      UBRECSP = C2D(SUBSTR(RECORD,197,4))
 RETURN
/*************************************************************/
/* PROCESS BACKUP VERSION INFORMATION RECORD                 */
/*   WRITE THE FORMATTED FIELDS TO YOUR REPORT               */
/*************************************************************/
```

```
 DISPLAYUB:
 PUSH "B:DSN="UBDSNAM " DATACL="LEFT(UBDATCL,1Ø) ,
      " MGMTCL="LEFT(UBMGTCL,1Ø) ,
      " STORCL="LEFT(UBSTGCL,1Ø) " DATE="UBDATE ,
      " TIME="UBTIME
 "EXECIO  1 DISKW OUTFILE"
 RETURN
/*************************************************************/
/* PROCESS DASD CAPACITY PLANNING RECORD                     */
/*   CONVERT THE RECORD INTO INDIVIDUAL FIELD VARIABLES.     */
/*************************************************************/
 UCRECORD:
       UCVOLSR = SUBSTR(RECORD,25,6)
       UCCOLDT = C2X(SUBSTR(RECORD,31,4))
       UCFLAG1 = BITSTR(SUBSTR(RECORD,35,1))
       UCLEVEL = SUBSTR(UCFLAG1,1,2)
       UCTOTAL = C2D(SUBSTR(RECORD,37,4))
       UCTGOCC = C2D(SUBSTR(RECORD,41,1))
       UCTROCC = C2D(SUBSTR(RECORD,42,1))
       UCBFOCC = C2D(SUBSTR(RECORD,43,1))
       UCAFOCC = C2D(SUBSTR(RECORD,44,1))
       UCNOMIG = C2D(SUBSTR(RECORD,45,1))
       UCNINTV = C2D(SUBSTR(RECORD,46,1))
       UCINTVM = C2D(SUBSTR(RECORD,47,1))
RETURN
DISPLAYUC:
PUSH "C:VOLSER="UCVOLSR " MIGLEVEL="UCLEVEL " DATE="UCCOLDT ,
      " OBJ="UCTGOCC "  AV MIG="UCBFOCC "  AP MIG="UCAFOCC ,
      "  NB SPC MGNT="UCNINTV " %NOMIG="UCNOMIG
 "EXECIO  1 DISKW OUTFILE"
 RETURN
/*************************************************************/
/* PROCESS TAPE CAPACITY PLANNING RECORD                     */
/*   CONVERT THE RECORD INTO INDIVIDUAL FIELD VARIABLES.     */
/*************************************************************/
 UTRECORD:
       UTSTYPE = SUBSTR(RECORD,25,1)
       UTFULL  = C2D(SUBSTR(RECORD,29,4))
       UTPART  = C2D(SUBSTR(RECORD,33,4))
       UTEMPTY = C2D(SUBSTR(RECORD,37,4))
 RETURN
/*************************************************************/
/* PROCESS TAPE CAPACITY PLANNING RECORD                     */
/*   WRITE THE FORMATTED FIELDS TO YOUR REPORT               */
/*************************************************************/
 DISPLAYUT:
 PUSH "T:TAPE VOLUME TYPE="UTSTYPE "  NB FULL="UTFULL ,
      "  NB PART="UTPART "  NB EMPTY="UTEMPTY
 "EXECIO  1 DISKW OUTFILE"
 RETURN
```

```
/*************************************************************/
/* CONVERT THE HEX VALUE TO INDIVIDUAL 1'S AND Ø'S           */
/* EACH BYTE WILL TAKE UP 8 CHARACTER POSITIONS              */
/*************************************************************/
 BITSTR:
   IF BIT.F ¬= "1111"  THEN DO
      BIT.Ø="ØØØØ"; BIT.1="ØØØ1"; BIT.2="ØØ1Ø"; BIT.3="ØØ11"
      BIT.4="Ø1ØØ"; BIT.5="Ø1Ø1"; BIT.6="Ø11Ø"; BIT.7="Ø111"
      BIT.8="1ØØØ"; BIT.9="1ØØ1"; BIT.A="1Ø1Ø"; BIT.B="1Ø11"
      BIT.C="11ØØ"; BIT.D="11Ø1"; BIT.E="111Ø"; BIT.F="1111"
   END
   CH=C2X(ARG(1))
   BS=''
   DO I=1 TO LENGTH(CH)
      Q=SUBSTR(CH,I,1)
      BS=BS≥≥BIT.Q
   END
RETURN BS
```

JCL

```
//DCOLHSMØ JOB SYS,
//         'PHILG-PSE-LYON',
//         CLASS=Y,
//         COND=(Ø,LT),
//         MSGCLASS=9,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID,
//*        TYPRUN=SCAN,
//         REGION=4M,
//         RESTART=*
//*
//*********************************************************************
//**                                                                 **
//**  IDCAMS DCOLLECT OF DFSMSHSM 1.3 INFORMATION                    **
//**                                                                 **
//*********************************************************************
//**  USE OF THE DCOLREXX EXEC (ARCTOOLS IN SAMPLIB)                 **
//*********************************************************************
//*
//IDCAMS   EXEC  PGM=IDCAMS
//*========================
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.HSM.DCOLLECT    NONVSAM PURGE
 DELETE EXPL69.HSM.REPORT      NONVSAM PURGE
 DELETE EXPL69.HSM.DCMIG       NONVSAM PURGE
 DELETE EXPL69.HSM.DCBAC       NONVSAM PURGE
 DELETE EXPL69.HSM.DCVOL       NONVSAM PURGE
 DELETE EXPL69.HSM.DCTAP       NONVSAM PURGE
```

```
 IF MAXCC = 8 THEN SET MAXCC = Ø
/*
//ALLOCAT EXEC PGM=IEFBR14
//ALLOCAT DD DSN=EXPL69.HSM.REPORT,
//           DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(75,15)),
//           DCB=(DSORG=PS,RECFM=FB,LRECL=132),
//           STORCLAS=BASE
//*
//DCOLLECT    EXEC PGM=IDCAMS
//*-------------------------
//SYSPRINT    DD SYSOUT=*
//MCDS        DD DSN=HSM.MCDS,DISP=SHR
//BCDS        DD DSN=HSM.BCDS,DISP=SHR
//OUTDS       DD DSN=EXPL69.HSM.DCOLLECT,DISP=(NEW,CATLG,DELETE),
//            UNIT=338Ø,DCB=(RECFM=VB,LRECL=644,BLKSIZE=Ø),
//            SPACE=(TRK,(1Ø5,15),RLSE)
//SYSIN       DD *
    DCOLLECT OFILE(OUTDS)    -
           BACKUPDATA -
           CAPPLANDATA -
           MIGRATEDATA
/*
//TSOBATCH EXEC PGM=IKJEFTØ1
//*-------------------------
//SYSEXEC  DD DSN=RDVUSER.EXEC,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 %DCOLREXX
/*
//SORTØØØ1 EXEC  PGM=ICEMAN
//*========================
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//*
//SYSOUT   DD SYSOUT=*
//*
//SORTIN   DD DISP=SHR,DSN=EXPL69.HSM.REPORT
//SORTOUT  DD DSN=EXPL69.HSM.DCMIG,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(4Ø,5),RLSE),UNIT=SYSDA,RECFM=FB,LRECL=132
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 SORT     FIELDS=(1,1,CH,A)
 OPTION   DYNALLOC
 INCLUDE  COND=(1,2,CH,EQ,C'M:')
/*
//SORTØØØ2 EXEC  PGM=ICEMAN
//*========================
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//*
```

```
//SYSOUT   DD SYSOUT=*
//*
//SORTIN   DD DISP=SHR,DSN=EXPL69.HSM.REPORT
//SORTOUT  DD DSN=EXPL69.HSM.DCBAC,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(4Ø,5),RLSE),UNIT=SYSDA,RECFM=FB,LRECL=132
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 SORT     FIELDS=(1,1,CH,A)
 OPTION   DYNALLOC
 INCLUDE  COND=(1,2,CH,EQ,C'B:')
/*
//SORTØØØ3 EXEC  PGM=ICEMAN
//*=========================
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//*
//SYSOUT   DD SYSOUT=*
//*
//SORTIN   DD DISP=SHR,DSN=EXPL69.HSM.REPORT
//SORTOUT  DD DSN=EXPL69.HSM.DCVOL,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(2Ø,5),RLSE),UNIT=SYSDA,RECFM=FB,LRECL=132
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 SORT     FIELDS=(1,1,CH,A)
 OPTION   DYNALLOC
 INCLUDE  COND=(1,2,CH,EQ,C'C:')
/*
//SORTØØØ4 EXEC  PGM=ICEMAN
//*=========================
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//*
//SYSOUT   DD SYSOUT=*
//*
//SORTIN   DD DSN=EXPL69.HSM.REPORT,DISP=(OLD,DELETE,KEEP)
//SORTOUT  DD DSN=EXPL69.HSM.DCTAP,DISP=(NEW,CATLG,DELETE),
//            SPACE=(TRK,(1,Ø),RLSE),UNIT=SYSDA,RECFM=FB,LRECL=132
/*
//SORTDIAG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 SORT     FIELDS=(1,1,CH,A)
 OPTION   DYNALLOC
 INCLUDE  COND=(1,2,CH,EQ,C'T:')
/*
```

Backing up VSAM from TSO ISPF can present problems. The following utility should help. (See also the following article on page 44, entitled *Checking DFHSM volume dumps*.)

Utility

```
//HSMNODU JOB EXP,'SASHSM',CLASS=W,MSGCLASS=O,MSGLEVEL=(1,1),
//         NOTIFY=DUNAND,COND=(9,LT),USER=SYSOP8,PASSWORD=MANXX,
//         PERFORM=4
//JOBLIB   DD DSN=RDVLYO.BATCH.LOADLIB,DISP=SHR
//         DD DSN=SRS.LOADLIB,DISP=SHR
//       DD DSN=SYS1.VSCLLIB,DISP=SHR
//HSNODU EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD SYSOUT=*
//SYSIN   DD *

  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;

LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HSMNODUM);
  RUN;
//DELFAIL EXEC PGM=IDCAMS,REGION=2048K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'EXPL69.DUMPFAIL'
/*
//DUMPFAIL EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD DSN=EXPL69.DUMPFAIL,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SASLIST DD SYSOUT=A
//SYSIN   DD *

  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;
```

```
LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HSMNODUZ);
  RUN;
/*
//TESTVIDE EXEC PGM=SYFICUS,PARM='EXPL69.DUMPFAIL'
//SYSPRINT DD SYSOUT=*
/*
//*  IF THE DUMPFAIL IS NOT EMPTY NOT OK MESSAGE
//MESANO   EXEC PGM=IKJEFTØ1,COND=(Ø4,EQ,TESTVIDE)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(DUNAND)
     LOGON
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(PHILG) LOGON
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(CEXP) LOGON
/*
//*  * * NUMBER OF KB PER STORAGE GROUP * * * *//
//DCOLLEC  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.DCOLLECT
 IF MAXCC <= 8 THEN SET MAXCC=Ø
/*
//*
//* DCOLLECT
//*
//SGLIST      EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//MCDS        DD DSN=HSM.MCDS,DISP=SHR
//BCDS        DD DSN=HSM.BCDS,DISP=SHR
//OUTDS       DD DSN=EXPL69.DCOLLECT,DISP=(,CATLG,DELETE),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=264),
//            SPACE=(TRK,(295,85),RLSE)
//SYSIN       DD *
    DCOLLECT OFILE(OUTDS)    -
             VOLUMES(*)
//SMFRMM EXEC PGM=IDCAMS,REGION=2Ø48K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'SMS.DFRMM.AUDREPT'
   DELETE 'SMS.DFRMM.SECREPT'
/*
//AUDREPT EXEC PGM=EDGAUD,REGION=5M,
//             PARM='SMFAUD(133),SMFSEC(134),SEC(IUO),DATEFORM(E)'
//SYSPRINT DD SYSOUT=O
//SYSOUT   DD SYSOUT=O
//SMFIN DD DISP=SHR,DSN=EXPL69.CPESMF
//AUDREPT DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//           SPACE=(4Ø96,(399,99),RLSE),DSN=SMS.DFRMM.AUDREPT,
```

```
//           DCB=(DSORG=PS,RECFM=VBS,BLKSIZE=4096,LRECL=32000),
//           DATACLAS=SEQCOMPD
//SECREPT DD DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//           SPACE=(4096,(199,99),RLSE),DSN=SMS.DFRMM.SECREPT,
//           DCB=(DSORG=PS,RECFM=VBS,BLKSIZE=4096,LRECL=32000)
//SYSIN DD *
SELECT -
VOLUMES(40*,80*)
//DELOUT  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.DISQUE.LIST
 IF MAXCC <= 8 THEN SET MAXCC=0
/*
//VOLSPACE EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD DSN=EXPL69.DISQUE.LIST,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SYSIN   DD *

  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;

LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HIER);
   %INCLUDE SOURCLIB(VOLSPAZ);
  RUN;
/*
//DELLK7 EXEC PGM=IDCAMS,REGION=2048K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'EXPL69.CEXP.LST.K7'
/*
// EXEC SAS
//RMMDATA DD DSN=SMS.DFRMM.SASDATA,DISP=SHR
//SASLIST DD DSN=EXPL69.CEXP.LST.K7,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,9),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SYSIN DD *
TITLE1 "LIST OF CARTRIDGES / BY DSN AND OCCUPANCY       ";
PROC SORT DATA=RMMDATA.EDGRVEXT OUT=VOL1;
     BY  RVVOLSER;
PROC SORT DATA=RMMDATA.EDGRDEXT OUT=VOL2;
     BY  RDVOLSER;
```

```
DATA;
      MERGE VOL1 VOL2 ( RENAME = (RDVOLSER = RVVOLSER));
      BY RVVOLSER;
PROC SORT ;
     BY  RDDSNAME;
           WHERE   RDDSNAME LIKE 'RD_.___._____.ADUMP.%';
PROC PRINT SPLIT='*';
     VAR  RDDSNAME RVVOLSER RVTUSE RVSTATUS  RVCRDATE RVCRTIME ;
RUN;
/*
//DSNOLD1 EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD SYSOUT=*
//SYSIN   DD *

  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;

LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HIER);
   %INCLUDE SOURCLIB(DSNOLD1);
  RUN;
/*
//DELLST  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE EXPL69.CICS.TREP.LST
 IF MAXCC <= 8 THEN SET MAXCC=0
/*
//*
//*
//*
//*
//*
/*
//*******************************************************************
//* GENERATE A LISTCAT FOR A USUAL      CATALOG                     *
//*******************************************************************
//* DELETE FROM FILE THE DISK TO WHICH LIST RECEIVED
//*
//S1       EXEC PGM=IDCAMS,REGION=2M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE (EXPL69.LISTCAT.ALL.OK ) NONVSAM PURGE
 IF LASTCC <=8 THEN SET MAXCC=0
```

```
//*
//LISEXPL  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=(NEW,CATLG,DELETE),
//             UNIT=SYSDA,SPACE=(CYL,(49,99)),
//             DCB=(LRECL=125,BLKSIZE=625,RECFM=VBA)
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CAT.EXPL) OUTFILE(OUTDD)
/*
/*
//LISTEST  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=MOD
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CAT.TEST) OUTFILE(OUTDD)
/*
//LISTCA2  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=MOD
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CAT.EXPL2) OUTFILE(OUTDD)
/*
/*
//LISTTSO  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=MOD
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CAT.TSO) OUTFILE(OUTDD)
/*
//LISSYST  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=MOD
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CAT.SYST) OUTFILE(OUTDD)
/*
/*
//LISEXP2  EXEC PGM=IDCAMS,REGION=6M
//SYSPRINT DD  SYSOUT=*
//OUTDD    DD  DSN=EXPL69.LISTCAT.ALL.OK,DISP=MOD
//SYSIN     DD  *
 LISTCAT  ALL  CATALOG(CATALOG.OS390.VLYO996) OUTFILE(OUTDD)
/*
//*
//* DELETE FROM FILE THE DISK TO WHICH LIST RECEIVED
//*
//*
//DELFIC   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DELETE (EXPL69.LISTCAT.ALL.CEXP) PURGE
```

```
   DELETE (EXPL69.LISTCAT.ALL.STAT) PURGE
   IF MAXCC <=8 THEN SET MAXCC=Ø
/*
//*
//LSTJOB2 EXEC PGM=IDCAMS,REGION=4M
//E1       DD DSN=EXPL69.LISTCAT.ALL.OK,DISP=SHR
//*
//S1       DD DSN=EXPL69.LISTCAT.ALL.CEXP,
//         DISP=(,CATLG,DELETE),
//         LIKE=EXPL69.LISTCAT.ALL.OK
//*        SPACE=(TRK,(2ØØ,3ØØ),RLSE),
//*        RECFM=VBS,BLKSIZE=Ø,LRECL=125,DSORG=PS,
//*        STORCLAS=BASE,MGMTCLAS=AUDIT1
//*
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD *
 REPRO INFILE(E1) OUTFILE(S1)
/*
//*
//DELFICS  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DELETE (EXPL69.LISTCAT.ALL.STAT) PURGE
   IF MAXCC <=8 THEN SET MAXCC=Ø
/*
//*
//CEXDLSTC EXEC PGM=CEXDLSTC
//STEPLIB DD  DISP=SHR,DSN=EXPL69.CEXP.LOADLIB
//        DD  DISP=SHR,DSN=RDVLYO.BATCH.LOADLIB
//*-----------------------
//SYSOUT   DD SYSOUT=B
//SYSDBOUT DD SYSOUT=P
//SYSUDUMP DD SYSOUT=D
//ETAT     DD DSN=EXPL69.LISTCAT.ALL.CEXP,DISP=SHR
//VERIF   DD DSN=EXPL69.LISTCAT.ALL.STAT,
//           UNIT=SYSDA,SPACE=(TRK,(1Ø,5Ø),RLSE),
//         DISP=(,CATLG,DELETE),
//  DCB=(SYS1.PROCLIB,DSORG=PS,RECFM=FB,BLKSIZE=Ø,LRECL=12Ø),
//            STORCLAS=BASE,MGMTCLAS=INTERIM
/*
//XTRAND EXEC PGM=IDCAMS,REGION=2Ø48K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'EXPL69.VSAM.LST1'
   DELETE 'EXPL69.RACF.LST1'
/*
// EXEC SAS
//RMMDATA DD DSN=SMS.DFRMM.SASDATA,DISP=SHR
//SASLIST DD DSN=EXPL69.VSAM.LST1,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,9),RLSE),
```

```
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=Ø),MGMTCLAS=DEL32
//SYSIN DD *
TITLE1 "LIST OF VSAM FILES ACCORDING TO LSTCAT           ";
DATA VSAMLST;
      INFILE 'EXPL69.LISTCAT.ALL.STAT';
      INPUT DSN $ 1-44 DATE $ 46-53 RECTOT 55-63 RECDEL 65-73
            RECINS 75-83 RECUPD 85-93 CISPLIT 95-1Ø3
            SPLITCA 1Ø5-113 TYPE $ 115-123 SPAPRIM 125-133
            SPASEC 135-143 HIALLOC 145-153 HIUSED 155-163
            PERCENT 165-169 NBEXTEND 171-172 ;
PROC PRINT DATA=VSAMLST;
     VAR  DSN DATE HIALLOC HIUSED PERCENT ;
          WHERE   PERCENT < Ø5ØØØ;
          FOOTNOTE ' LIST FOR FILES WITH WASTE OF SPACE';
PROC PRINT DATA=VSAMLST;
     VAR  DSN DATE HIALLOC HIUSED PERCENT NBEXTEND;
          WHERE   NBEXTEND > 1Ø;
          FOOTNOTE ' LIST FOR FILES WITH EXTENT';
PROC PRINT DATA=VSAMLST;
     VAR  DSN DATE HIALLOC HIUSED RECTOT RECINS RECUPD RECDEL;
          WHERE   RECINS = Ø OR RECUPD = Ø OR RECDEL = Ø ;
          FOOTNOTE ' LIST FOR UNUSED FILES';
RUN;
/*
//* COPY OF BASE RACF ON A SEQUENTIAL FILE.
//*
//STEP1    EXEC PGM=IRRDBUØØ,PARM=NOLOCKINPUT
//SYSPRINT DD SYSOUT=*
//INDD1    DD DSN=SYS1.RACF19,DISP=SHR
//OUTDD    DD DSN=EXPL69.RACF.LST1,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(3ØØ,1ØØ)),RECFM=VB,LRECL=4Ø96
/*
/*
//*-------------------------------------------------------------------
//DEL EXEC PGM=IDCAMS,TIME=1
//*=========================
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEL (EXPL69.CTRL.CATS.ALIAS)
  DEL (EXPL69.CTRL.CATS.ALIASLST)
  DEL (EXPL69.CTRL.CATS.ALIASTRI)
/*
//LISTCALI EXEC PGM=IDCAMS,REGION=512K
//*===================================
//SYSPRINT DD  DSN=EXPL69.CTRL.CATS.ALIAS,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(15,15),RLSE),
//         DCB=(LRECL=125,RECFM=VBA,BLKSIZE=629),
//         UNIT=SYSDA,MGMTCLAS=INTERIM
//SYSIN    DD  *
 LISTCAT ALIAS UCAT ALL
```

```
/*
//*-------------------------------------------------------------
//LISTCATA EXEC PGM=SYSGCATA,TIME=1Ø
//*==================================
//*    SYSPRINT DD SYSOUT=Z
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD SYSOUT=W
//SYSDBOUT DD SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//*     SYSOUT   DD SYSOUT=*
//ALIAS    DD DSN=EXPL69.CTRL.CATS.ALIAS,DISP=OLD
//ALIASLST DD DSN=EXPL69.CTRL.CATS.ALIASLST,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(15,15),RLSE),UNIT=SYSDA,
//         MGMTCLAS=INTERIM
//*-------------------------------------------------------------
//TRIALIAS EXEC PGM=ICEMAN,PARM='SIZE(MAX)'
//*=======================================
//SORTLIB  DD DSN=SYS1.SORTLIB,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SORTIN   DD DSN=EXPL69.CTRL.CATS.ALIASLST,DISP=OLD
//SORTOUT  DD DSN=EXPL69.CTRL.CATS.ALIASTRI,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(15,15),RLSE),UNIT=SYSDA,
//         MGMTCLAS=INTERIM
//SORTWKØ1 DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWKØ2 DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWKØ3 DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWKØ4 DD UNIT=SYSDA,SPACE=(CYL,5)
//SORTWKØ5 DD UNIT=SYSDA,SPACE=(CYL,5)
//SYSIN    DD *
 SORT FIELDS=(1,3,A,32,8,A,6,15,A),FILSZ=E15ØØØ,FORMAT=BI
/*
//STEPLIB DD  DSN=RDVLYO.BATCH.LOADLIB,DISP=SHR
//*
//*-------------------------------------------------------------------
//*
//DELETES  EXEC PGM=IDCAMS
//*-----------------------------------
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE    (CEXP.CTRL.RACF.ANO  )  NONVSAM  PURGE
/*
//CEXRACF EXEC PGM=CEXDRACF,TIME=1,REGION=4Ø96K
//STEPLIB DD  DSN=EXPL69.CEXP.LOADLIB,DISP=SHR
//SYSOUT DD SYSOUT=P
//SYSABOUT DD SYSOUT=*
//SYSDBOUT DD SYSOUT=*
//LST1    DD DSN=EXPL69.RACF.LST1,DISP=SHR
//ALIAS    DD DSN=EXPL69.CTRL.CATS.ALIASTRI,DISP=SHR
//ANO      DD DSN=CEXP.CTRL.RACF.ANO,
//            DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(1,1),RLSE),
```

```
//            DCB=(RECFM=FB,LRECL=121)
//*
/*
//
//ANDRE EXEC PGM=IDCAMS,REGION=2Ø48K
//SYSPRINT DD SYSOUT=P
//SYSIN  DD *
   DELETE 'CEXP.LST.K7'
/*
// EXEC SAS
//RMMDATA DD DSN=SMS.DFRMM.SASDATA,DISP=SHR
//SASLIST DD DSN=CEXP.LST.K7,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,9),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=Ø),MGMTCLAS=DEL32
//SYSIN DD *
TITLE1 "LIST OF CARTRIDGES / BY DSN AND OCCUPANCY      ";
PROC SORT DATA=RMMDATA.EDGRVEXT OUT=VOL1;
     BY  RVVOLSER;
PROC SORT DATA=RMMDATA.EDGRDEXT OUT=VOL2;
     BY  RDVOLSER;
DATA;
      MERGE VOL1 VOL2 ( RENAME = (RDVOLSER = RVVOLSER));
      BY RVVOLSER;
PROC SORT ;
     BY  RDDSNAME;
PROC PRINT SPLIT='*';
     VAR  RDDSNAME RVVOLSER RVTUSE RVSTATUS  RVCRDATE RVCRTIME ;
RUN;
//*-------------------------------------------------------------
//ANDRE EXEC PGM=IDCAMS,REGION=2Ø48K
//SYSPRINT DD SYSOUT=P
//SYSIN  DD *
   DELETE 'CEXP.LST.NOADUMP'
/*
// EXEC SAS
//RMMDATA DD DSN=SMS.DFRMM.SASDATA,DISP=SHR
//SASLIST DD DSN=CEXP.LST.NOADUMP,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,9),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=Ø),MGMTCLAS=DEL32
//SYSIN DD *
TITLE1 "LIST OF CARTRIDGES / BY DSN AND OCCUPANCY       ";
PROC SORT DATA=RMMDATA.EDGRVEXT OUT=VOL1;
     BY  RVVOLSER;
PROC SORT DATA=RMMDATA.EDGRDEXT OUT=VOL2;
     BY  RDVOLSER;
DATA;
      MERGE VOL1 VOL2 ( RENAME = (RDVOLSER = RVVOLSER));
      BY RVVOLSER;
PROC SORT ;
     BY  RDDSNAME;
```

```
          WHERE   RDDSNAME NE   'RD_.___._____.ADUMP.%';
PROC PRINT SPLIT='*';
     VAR  RDDSNAME RVVOLSER RVTUSE RVSTATUS  RVCRDATE RVCRTIME ;
RUN;
//*-------------------------------------------------------------------
//DEL EXEC PGM=IDCAMS,TIME=1
//*=========================
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
  DEL (EXPL69.CTRL.VTOCALLD)
  DEL (EXPL69.CTRL.CATS.VOL)
  IF MAXCC<=8 THEN SET MAXCC=Ø
//*-----------------------------------------------------------------
//LISTCVOL EXEC PGM=IDCAMS,REGION=512K
//*===================================
//SYSPRINT DD  DSN=EXPL69.CTRL.CATS.VOL,DISP=(,CATLG,DELETE),
//         SPACE=(TRK,(1ØØ,99),RLSE),
//         DCB=(LRECL=125,RECFM=VBA,BLKSIZE=629),
//         UNIT=SYSDA,MGMTCLAS=INTERIM
//SYSIN    DD  *
 LISTCAT VOL CAT(CATALOG.OS39Ø.VLYO996)
 LISTCAT VOL CAT(CATALOG.HSM)
 LISTCAT VOL CAT(CAT.EXPL)
 LISTCAT VOL CAT(CAT.SYST)
 LISTCAT VOL CAT(CAT.EXPL2)
 LISTCAT VOL CAT(CAT.TSO)
 LISTCAT VOL CAT(CAT.TEST)
 LISTCAT VOL CAT(CATALOG.VDLB7Ø7)
/*
```

Claude Dunand
(France)

Checking DFHSM volume dumps

The HSMNODU program presented below can be used to check that DFHSM volume dumps have been successful.

HSMNODU calls HSMNODUM and HSMNODUZ in sysin. Note that SASJFIC, which collects information on the volumes, must be run first.

If there are any problems, a message is sent to the TSO user, who will be able to pass the following command to DFHSM:

```
dfhsm, backvol volume (vol001) dump (dumpclass(bando))
```


The IBM DCollect utility is used to create an output file. This same file is used on entry by an application written in SAS, which contains all the necessary information on the volumes, including information on the dumps. DICTFIC and SYSDCOL are used to create variables for the SAS dictionary

SASJFIC

```
//SASJFIC  JOB COM,'SASFIC',CLASS=W,MSGCLASS=O,
//       COND=(4,LT)
//DELETE   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE SAS.DCOLLECT
 IF MAXCC <= 8 THEN SET MAXCC=Ø
/*
//*
//* DCOLLECT
//*
//DCOLLECT    EXEC PGM=IDCAMS
//SYSPRINT    DD SYSOUT=*
//MCDS        DD DSN=HSM.MCDS,DISP=SHR
//BCDS        DD DSN=HSM.BCDS,DISP=SHR
//OUTDS       DD DSN=SAS.DCOLLECT,DISP=(,CATLG,DELETE),
//*           UNIT=SYSDA,DCB=(RECFM=VB,LRECL=264),
//            UNIT=SYSDA,DCB=(RECFM=VB,LRECL=644,BLKSIZE=Ø),
//            SPACE=(TRK,(1Ø5,15),RLSE)
//SYSIN       DD *
    DCOLLECT OFILE(OUTDS)    -
             VOLUMES(*)
/*
//DCOL   EXEC SAS,REGION=8M,
//       WORK='15Ø,2Ø',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//DCOLLECT DD DSN=SAS.DCOLLECT,DISP=SHR
//REPORT   DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//* TLMS     DD DSN=EXPL69.TLMS.VMF,DISP=SHR
//SASLIST  DD SYSOUT=O
//SYSIN    DD *

  OPTIONS PAGESIZE=6Ø LINESIZE=132 ;

  %CPSTART(MODE=BATCH,
           SYSTEM=MVS,
             ROOT=SAS.SAS6Ø9.TS45Ø.CPE.,
             PDB=SAS.BERCY.FICPDB.,
             DISP=OLD,
```

```
             ROOTSERV=,
             SHARE=N/A,
             MXGSRC=('SAS.BERCY.SOURCLIB' 'SAS.MXG.V1313.SOURCLIB'),
              ),
             MXGLIB=SAS.MXG.V1313.FORMATS
            ) ;


  %INCLUDE SOURCLIB(TYPEDCOL);
  RUN;

  %CMPROCES(,
            COLLECTR=GENERIC,
            TOOLNM=SASDS,
            UNIT=DISK,
            GENLIB=WORK
           );

  %CPREDUCE();

                  /****  DAILY REPORTS *****/

   %INCLUDE REPORT(OPTIONS);
   %INCLUDE REPORT(YESTERDAY);
   %INCLUDE REPORT(RJVOL);
/*
//* DELETE FROM FILE AFTER EXECUTION
//*
//DELETE   EXEC PGM=IDCAMS,COND=(Ø,NE,DCOL.SAS)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE SAS.DCOLLECT
/*
```

HSMNODU

```
//HSMNODU JOB EXP,'SASHSM',CLASS=W,MSGCLASS=O,MSGLEVEL=(1,1),
//         NOTIFY=DUNAND,COND=(9,LT),USER=SYSOP8,PASSWORD=MANXX,
//         PERFORM=4
//JOBLIB   DD DSN=RDVLYO.BATCH.LOADLIB,DISP=SHR
//         DD DSN=SRS.LOADLIB,DISP=SHR
//       DD DSN=SYS1.VSCLLIB,DISP=SHR
//HSNODU EXEC SAS,REGION=8M,
//       WORK='2ØØ,5Ø',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD SYSOUT=*
//SYSIN   DD *
```

```
  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;

LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HSMNODUM);
  RUN;
//DELFAIL EXEC PGM=IDCAMS,REGION=2048K
//SYSPRINT DD SYSOUT=O
//SYSIN  DD *
   DELETE 'EXPL69.DUMPFAIL'
/*
//DUMPFAIL EXEC SAS,REGION=8M,
//       WORK='200,50',
//         OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.BERCY.REPORTS,DISP=SHR
//LIBRARY  DD DSN=SAS.MXG.FORMATS,DISP=SHR
//SASLIST DD DSN=EXPL69.DUMPFAIL,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,SPACE=(TRK,(1,1),RLSE),
//        DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SASLIST DD SYSOUT=A
//SYSIN   DD *

  OPTIONS PAGESIZE=60 LINESIZE=132 ;

  %LET RETCODE=.;

LIBNAME MONTH 'SAS.BERCY.FICPDB.MONTH' DISP=SHR;
LIBNAME DETAIL'SAS.BERCY.FICPDB.DETAIL' DISP=SHR;
   %INCLUDE SOURCLIB(HSMNODUZ);
  RUN;
/*
//TESTVIDE EXEC PGM=SYFICUS,PARM='EXPL69.DUMPFAIL'
//SYSPRINT DD SYSOUT=*
/*
//*  IF DUMPFAIL IS NOT EMPTY DUMP NOT OK MESSAGE
//MESANO   EXEC PGM=IKJEFT01,COND=(04,EQ,TESTVIDE)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(DUNAND)
   LOGON
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(PHILG) LOGON
SE  'ATTENTION HSM DUMP NOT OK SEE EXPL69.DUMPFAIL' USER(CEXP) LOGON
/*
```

HSMNODUM

```
title "list of hsm dumps with errors";
proc print data=detail.xhsvsrs   (where=(vsrvtyp = "P"));
           label  width=UBY ;
  label vsrvol  = 'VOLSER'
        vsrvtyp = 'Type'
        datetime= 'Time'
        vsfdmpf = 'dump hsm failed'
  format datetime tod7.;
  by vsrvol notsorted ;
  id vsrvol ;
  var vsrvtyp datetime
       vsfdmpf vsrunit vsrhwm;
run;
```

HSMNODUZ

```
title "list of hsm dumps with errors";
proc print data=detail.xhsvsrs   (where=(vsrvtyp ne "P"
             and  vsfdmpf = "Y" ));
           label  width=UBY ;
  label vsrvol  = 'VOLSER'
        vsrvtyp = 'Type'
        datetime= 'Time'
        vsfdmpf = 'dump hsm failed'
  format datetime tod7.;
  by vsrvol notsorted ;
  id vsrvol ;
  var vsrvtyp datetime
       vsfdmpf vsrunit vsrhwm;
run;
```

DICTFIC

```
//SASDSMF  JOB SYS1Ø,SYSTEM,CLASS=W,MSGCLASS=Ø
//DICTFIC EXEC SAS,REGION=3M,OPTIONS='DMSBATCH TERMINAL MEMSIZE=24M'
//SYSIN   DD  *

    %CPSTART(MODE=BATCH,SYSTEM=MVS,ROOT=SAS.SAS6Ø8.CPE.,
         PDB=SAS.BERCY.FICPDB.,DISP=OLD,
         MXGSRC=('SAS.BERCY.SOURCLIB' 'SAS.MXG.SOURCLIB'),
         MXGLIB=SAS.MXG.FORMATS);

    %CPDDUTL(FILENAME='SAS.BERCY.PROGRAM(SYSDCO)');
```

SYSDCOL

```
set defaults
  collector=GENERIC
  toolname=SASDS
  extname=.
  label=' '
  description=' '
  kept=yes
  detail=(agelimit=7
         )
  day=(agelimit=35
       nocount nosum noaverage nominimum nomaximum
       nouss nocv norange novariance nostd nonmiss
      )
  week=(agelimit=Ø
       nocount nosum noaverage nominimum nomaximum
       nouss nocv norange novariance nostd nonmiss
      )
  month=(agelimit=14
       nocount nosum noaverage nominimum nomaximum
       nouss nocv norange novariance nostd nonmiss
      )
  year=(agelimit=Ø
       nocount nosum noaverage nominimum nomaximum
       nouss nocv norange novariance nostd nonmiss
      )
 ;


update TABLE NAME=DCOLDSE ;
  delete VARIABLE NAME=DCMGTCL noerror;
  create VARIABLE NAME=DCMGTCL
    EXTNAME=DCDMGTCL
    LABEL='Management Class'
    DESCRIPTION='Management Class'
    KEPT=YES VALIDITY=. INTERPRET=STRING TYPE=CHARACTER
    LENGTH=3Ø FORMAT=. INFORMAT=. IDNUM=Ø
    OID=.
    SUBJECT=''
  ;
  delete VARIABLE NAME=DCALLSP;
  CREATE VARIABLE NAME=DCALLSP
    EXTNAME=DCDALLSP
    LABEL='ALLOCATED_SPACE_<MGBYTES>'
    DESCRIPTION='ALLOCATED_SPACE_(MGBYTES)'
    KEPT=YES VALIDITY=. INTERPRET=COUNT TYPE=NUMERIC
    LENGTH=8 FORMAT=MGBYTES. INFORMAT=. IDNUM=Ø
    OID=.
    SUBJECT=''
```

```
    DAY =(
      NOCOUNT   SUM   AVERAGE   MAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    MONTH =(
      NOCOUNT   SUM   AVERAGE   MAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    WEEK =(
      NOCOUNT NOSUM NOAVERAGE NOMAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    YEAR =(
      NOCOUNT NOSUM NOAVERAGE NOMAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
  ;
  delete VARIABLE NAME=DCUSESP;
  CREATE VARIABLE NAME=DCUSESP
    EXTNAME=DCDUSESP
    LABEL='USED_SPACE_<MGBYTES>'
    DESCRIPTION='USED_SPACE_(MGBYTES)'
    KEPT=YES VALIDITY=. INTERPRET=COUNT TYPE=NUMERIC
    LENGTH=8 FORMAT=MGBYTES. INFORMAT=. IDNUM=Ø
    OID=.
    SUBJECT=''
    DAY =(
      NOCOUNT   SUM   AVERAGE   MAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    MONTH =(
      NOCOUNT   SUM   AVERAGE   MAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    WEEK =(
      NOCOUNT NOSUM NOAVERAGE NOMAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
    YEAR =(
      NOCOUNT NOSUM NOAVERAGE NOMAXIMUM NOMINIMUM NOUSS NOCV NORANGE
      NOVARIANCE NOSTD NONMISS
      )
  ;

UPDATE TABLE NAME=DCOLDSE
  DETAIL =( AGELIMIT=7
     BYVARS='MACHINE APPLI DCDSNAM DCVOLSR DATETIME'
     )
  DAY =( AGELIMIT=35
    CLASSVARS=
```

```
   'MACHINE APPLI DCDSNAM DCVOLSR ANNEE MOIS JOUR DATETIME DCMGTCL'
     )
  WEEK =( AGELIMIT=Ø
     CLASSVARS='MACHINE DATETIME'
     )
  MONTH =( AGELIMIT=13
     CLASSVARS=
'MACHINE APPLI YEAR MONTH DATETIME DCVOLSR ARTICLE CODEDI DCMGTCL'
     )
  YEAR =( AGELIMIT=Ø
     CLASSVARS='MACHINE DATETIME'
     )
;
```

Claude Dunand
(France)

Statistics display for a DB2/VSAM dataset

We have often found ourselves in ISPF option 3.4, looking at the space allocation and extents of VSAM datasets (especially DB2/VSAM datasets). Although there is no real problem in looking at the VSAM statistics using the LISTC command, the output is not easily readable. We therefore decided to redirect the output to a dataset and view it using the ISPF browse facility. This means that we can use the scroll left and right keys and also the page up and page down keys.

Fellow DBAs make up separate SQL queries to DB2 catalog tables in order to find out statistics from them for corresponding VSAM datasets. We wrote a REXX procedure to display both the LISTC output and the statistics from the DB2 catalog. This procedure displays the relevant statistics from the DB2 catalog, depending on whether the dataset is a tablespace or an indexspace.

The procedures described here use two programs which were previously published in *DB2 Update*:

- SQLISPF – an SQL interface to ISPF Dialog Manager. This is an Assembler program to pass the query as a REXX variable and return the result in an ISPF table (*DB2 Update*, May 1992).

- DB2CAF – this program uses two modules: DB2CAFI, initiate the Call Attachment Facility, and DB2CAFT, terminate the Call Attachment Facility (*DB2 Update*, August 1991).

You should have SQLISPF compiled and link edited as SQLPROD and SQLTEST in your production and test DB2 environments respectively, with the EXECUTE option either public or granted to you.

REXX EXEC SOURCE CODE

```
/***************************** REXX exec*********************************/
/* DB2LC: Display statistics for a DB2/VSAM dataset.                    */
/************************************************************************/
ARG dsn1 .
  debug = 1
  debug = Ø
  dsnsav = dsn1
  dsn1 = STRIP(dsn1,,"'")
  workdsn = "'"||dsn1||"'"
  j = 1
  x  = listdsi(workdsn)
  IF sysdsorg ¬= 'VS' THEN
  DO
    zedsmsg = " "
    zedlmsg = dsn1 " is not a VSAM dataset "
    ADDRESS ISPEXEC "SETMSG MSG(ISRZØØ1)"
    EXIT
  END
  PARSE VAR dsn1 v1'.'v2'.'dbn'.'tsn'.'icon'.'acon
  IF debug = 1 THEN
  DO
    SAY v1
    SAY v2
    SAY dbn
    SAY tsn
    SAY icon
    SAY acon
  END
/************************************************************************/
/* At our shop we name the PROD DB2 VSAM datasets with first 3 chars */
/* of the first qualifier as PRD and for TEST DB2 the first 3 chars  */
/* are TST. So either PRD/TST version of the program is invoked.     */
/************************************************************************/
  hlq3 = ' '
  hlq3 = SUBSTR(v1,1,3)
```

```
/*********************************************************************/
/* Assume it is a DB2 tablespace and look for the info in catalog    */
/*********************************************************************/
  CALL CHK_TABLE
  IF trc = Ø THEN
  DO
    ADDRESS ISPEXEC "TBTOP  " seltab
    ADDRESS ISPEXEC "TBSKIP " seltab
    skiprc = rc
    CALL TS_HEAD
    DO WHILE skiprc = Ø
       CALL TS_DETAIL
       ADDRESS ISPEXEC "TBSKIP " seltab
       skiprc = rc
    END
    CALL CHK_PART_TS
    IF trc = Ø THEN
    DO
      ADDRESS ISPEXEC "TBTOP  " seltab
      ADDRESS ISPEXEC "TBSKIP " seltab
      skiprc = rc
      CALL TS_HEAD_PART
      DO WHILE skiprc = Ø
         CALL TS_DETAIL_PART
         ADDRESS ISPEXEC "TBSKIP " seltab
         skiprc = rc
      END
    END
  END
/*********************************************************************/
/* If not tablespace look for indexspace in the catalog.             */
/*********************************************************************/
  ELSE
  DO
    CALL CHK_INDEX
    IF irc = Ø THEN
    DO
      ADDRESS ISPEXEC "TBTOP  " seltab
      ADDRESS ISPEXEC "TBSKIP " seltab
      skiprc = rc
      CALL IX_HEAD
      DO WHILE skiprc = Ø
         CALL IX_DETAIL
         ADDRESS ISPEXEC "TBSKIP " seltab
         skiprc = rc
      END
      CALL CHK_PART_IX
      IF irc = Ø THEN
      DO
        ADDRESS ISPEXEC "TBTOP  " seltab
```

```
        ADDRESS ISPEXEC "TBSKIP " seltab
        skiprc = rc
        CALL IX_HEAD_PART
        DO WHILE skiprc = Ø
           CALL IX_DETAIL_PART
           ADDRESS ISPEXEC "TBSKIP " seltab
           skiprc = rc
        END
      END
    END
  END
/*********************************************************************/
/* Execute the LISTCAT command and save the output in a dataset.    */
/*********************************************************************/
  CALL LST_INFO
/*********************************************************************/
/* Write the resultant lines from the DB2 catatog                    */
/*********************************************************************/
  CALL WRT_LINES
EXIT
/*********************************************************************/
/* Check for the tablespace                                          */
/*********************************************************************/
CHK_TABLE:
  SELPARM = 'SELECT NAME, CREATOR, DBNAME, ',
   'PARTITIONS, SPACE ,STATUS, NTABLES, SEGSIZE, NACTIVE ',
   'FROM SYSIBM.SYSTABLESPACE ',
   "WHERE DBNAME='"dbn"'" " AND " ,
   "      NAME  ='"tsn"'"  ;
  IF debug = 1 THEN
     SAY SELPARM
  SELTAB = "TEMP";
  ADDRESS ISPEXEC "VPUT (SELPARM SELTAB) SHARED"
  IF hlq3 = 'PRD' THEN
     ADDRESS ISPEXEC "SELECT PGM(SQLPROD)"
  ELSE
     ADDRESS ISPEXEC "SELECT PGM(SQLTEST)"
  trc = rc
RETURN
/*********************************************************************/
/* Check for the tablespace - partition info                         */
/*********************************************************************/
CHK_PART_TS:
  SELPARM = 'SELECT DBNAME, TSNAME, ',
   'PQTY  ,SQTY, FREEPAGE, PCTFREE, PARTITION  ',
   'FROM SYSIBM.SYSTABLEPART  ',
   "WHERE DBNAME='"dbn"'" " AND " ,
   "      TSNAME='"tsn"'"  ,
   "ORDER BY DBNAME,TSNAME,PARTITION "  ;
  IF debug = 1 THEN
```

```
     SAY SELPARM
  SELTAB = "TEMP";
  ADDRESS ISPEXEC "VPUT (SELPARM SELTAB) SHARED"
  IF hlq3 = 'PRD' THEN
     ADDRESS ISPEXEC "SELECT PGM(SQLPROD)"
  ELSE
     ADDRESS ISPEXEC "SELECT PGM(SQLTEST)"
  trc = rc
RETURN
/*********************************************************************/
/* Check for the indexspace                                          */
/*********************************************************************/
CHK_INDEX:
  SELPARM = 'SELECT NAME, CREATOR, TBNAME, TBCREATOR"TBCREA", ',
   'UNIQUERULE"URULE", COLCOUNT,  PGSIZE, ERASERULE"ERULE", ',
   'SPACE ',
   'FROM SYSIBM.SYSINDEXES    ',
   "WHERE DBNAME='"dbn"'" " AND " ,
   "      INDEXSPACE ='"tsn"'"  ;
  IF debug = 1 THEN
     SAY SELPARM
  SELTAB = "TEMPI";
  ADDRESS ISPEXEC "VPUT (SELPARM SELTAB) SHARED"
  IF hlq3 = 'PRD' THEN
     ADDRESS ISPEXEC "SELECT PGM(SQLPROD)"
  ELSE
     ADDRESS ISPEXEC "SELECT PGM(SQLTEST)"
  irc = rc
RETURN
/*********************************************************************/
/* Check for the indexspace - partition info                         */
/*********************************************************************/
CHK_PART_IX:
  SELPARM = 'SELECT IXCREATOR, IXNAME, ',
   'PQTY  ,SQTY, FREEPAGE, PCTFREE , PARTITION ',
   'FROM SYSIBM.SYSINDEXPART  ',
   "WHERE IXNAME='"inam"'" " AND " ,
   "   IXCREATOR='"icre"'" ,
   "ORDER BY IXCREATOR,IXNAME,PARTITION "  ;
  IF debug = 1 THEN
     SAY SELPARM
  SELTAB = "TEMPI";
  ADDRESS ISPEXEC "VPUT (SELPARM SELTAB) SHARED"
  IF hlq3 = 'PRD' THEN
     ADDRESS ISPEXEC "SELECT PGM(SQLPROD)"
  ELSE
     ADDRESS ISPEXEC "SELECT PGM(SQLTEST)"
  irc = rc
RETURN
```

```
/*********************************************************************/
/* Tablespace heading                                                */
/*********************************************************************/
TS_HEAD:
     C1 = "NAME"
     C2 = "CREATOR"
     C3 = "DBNAME"
     C4 = "PART"
     C5 = "SPACE"
     C6 = "STATUS"
     C7 = "NTABLES"
     C8 = "SEGSIZE"
     C9 = "NACTIVE"
     lin.j = ' Info from SYSIBM.SYSTABLESPACE ' || ,
             ' (One row for each tablespace) '
     j = j + 1
     lin.j = ' '||c1||COPIES(' ',(8-LENGTH(c1)))||' ',
             ||c2||COPIES(' ',(8-LENGTH(c2)))||' ',
             ||c3||COPIES(' ',(8-LENGTH(c3)))||' ',
             ||c4||COPIES(' ',(6-LENGTH(c4)))||' ',
             ||c5||COPIES(' ',(8-LENGTH(c5)))||' ',
             ||c6||COPIES(' ',(6-LENGTH(c6)))||' ',
             ||c7||COPIES(' ',(8-LENGTH(c7)))||' ',
             ||c8||COPIES(' ',(8-LENGTH(c8)))||' ',
             ||c9||COPIES(' ',(8-LENGTH(c9)))
     IF debug = 1 THEN
        SAY lin.j
     j = j + 1
RETURN
/*********************************************************************/
/* Tablespace detail                                                 */
/*********************************************************************/
TS_DETAIL:
 lin.j =  ' '||name||COPIES(' ',(8-LENGTH(name)))||' '||,
          creator||COPIES(' ',(8-LENGTH(creator)))||' '||,
          dbname||COPIES(' ',(8-LENGTH(dbname)))||' '||,
          partitio||COPIES(' ',(6-LENGTH(partitio)))||' '||,
          space||COPIES(' ',(8-LENGTH(space)))||' '||,
          status||COPIES(' ',(6-LENGTH(status)))||' '||,
          ntables||COPIES(' ',(8-LENGTH(ntables)))||' '||,
          segsize||COPIES(' ',(8-LENGTH(segsize)))||' '||,
          nactive||COPIES(' ',(8-LENGTH(nactive)))
  tharba = space * 1Ø24
  thurba = nactive * 4 * 1Ø24
  IF debug = 1 THEN
     SAY lin.j
  j = j + 1
  lin.j =  ' '
  j = j + 1
  lin.j =  ' Space    in KB and equivalent of HI-ALLOC-RBA  'tharba
```

```
  j = j + 1
  lin.j =  ' Nactive  in PAGE and equivalent of HI-USED-RBA 'thurba
  j = j + 1
  lin.j =  ' (Assuming a 4K Page ) '
  j = j + 1
  lin.j =  '  '
  j = j + 1
RETURN
/*********************************************************************/
/* Tablespace heading - partition                                    */
/*********************************************************************/
TS_HEAD_PART:
     C1 = "DBNAME"
     C2 = "TSNAME "
     C3 = "PQTY"
     C4 = "SQTY"
     C5 = "FREEPAGE"
     C6 = "PCTFREE"
     C7 = "PART"
     lin.j = ' Info from SYSIBM.SYSTABLEPART  ' ||,
             ' (One row for each unpartitioned Tablespace '
     j = j + 1
     lin.j = ' and one row for each partition of a partitioned TS)'
     j = j + 1
     lin.j = ' '||c1||COPIES(' ',(8-LENGTH(c1)))||' ',
             ||c2||COPIES(' ',(8-LENGTH(c2)))||' ',
             ||c3||COPIES(' ',(8-LENGTH(c3)))||' ',
             ||c4||COPIES(' ',(8-LENGTH(c4)))||' ',
             ||c5||COPIES(' ',(8-LENGTH(c5)))||' ',
             ||c6||COPIES(' ',(8-LENGTH(c6)))||' ',
             ||c7||COPIES(' ',(8-LENGTH(c7)))
     IF debug = 1 THEN
        SAY lin.j
     j = j + 1
RETURN
/*********************************************************************/
/* Table space detail - partition                                    */
/*********************************************************************/
TS_DETAIL_PART:
 lin.j =  ' '||dbname||COPIES(' ',(8-LENGTH(dbname)))||' '||,
          tsname||COPIES(' ',(8-LENGTH(tsname)))||' '||,
          pqty||COPIES(' ',(8-LENGTH(pqty)))||' '||,
          sqty||COPIES(' ',(8-LENGTH(sqty)))||' '||,
          freepage||COPIES(' ',(8-LENGTH(freepage)))||' '||,
          pctfree||COPIES(' ',(8-LENGTH(pctfree)))||' '||,
          partitio||COPIES(' ',(8-LENGTH(partitio)))
  IF debug = 1 THEN
     SAY lin.j
  j = j + 1
  /*
```

```
  lin.j =  ' '
  j = j + 1
  */
RETURN
/*********************************************************************/
/* Index space heading                                               */
/*********************************************************************/
IX_HEAD:
     C1 = "NAME"
     C2 = "CREATOR"
     C3 = "TBNAME"
     C4 = "TBCREA"
     C5 = "URULE"
     C6 = "COLCOUNT"
     C7 = "PGSIZE"
     C8 = "ERULE"
     C9 = "SPACE"
     lin.j = ' Info from SYSIBM.SYSINDEXES ' || ,
             ' one row for every index '
     j = j + 1
     lin.j = ' '||c1||COPIES(' ',(8-LENGTH(c1)))||' ',
             ||c2||COPIES(' ',(8-LENGTH(c2)))||' ',
             ||c3||COPIES(' ',(8-LENGTH(c3)))||' ',
             ||c4||COPIES(' ',(8-LENGTH(c4)))||' ',
             ||c5||COPIES(' ',(8-LENGTH(c5)))||' ',
             ||c6||COPIES(' ',(8-LENGTH(c6)))||' ',
             ||c7||COPIES(' ',(8-LENGTH(c7)))||' ',
             ||c8||COPIES(' ',(8-LENGTH(c8)))||' ',
             ||c9||COPIES(' ',(8-LENGTH(c9)))
     IF debug = 1 THEN
        SAY lin.j
     j = j + 1
RETURN
/*********************************************************************/
/* Index space detail                                                */
/*********************************************************************/
IX_DETAIL:
 lin.j =  ' '||name||COPIES(' ',(8-LENGTH(name)))||' '||,
          creator||COPIES(' ',(8-LENGTH(creator)))||' '||,
          tbname||COPIES(' ',(8-LENGTH(tbname)))||' '||,
          tbcrea||COPIES(' ',(8-LENGTH(tbcrea)))||' '||,
          urule||COPIES(' ',(8-LENGTH(urule)))||' '||,
          colcount||COPIES(' ',(8-LENGTH(colcount)))||' '||,
          pgsize||COPIES(' ',(8-LENGTH(pgsize)))||' '||,
          erule||COPIES(' ',(8-LENGTH(erule)))||' '||,
          space||COPIES(' ',(8-LENGTH(space)))
  tharba = space * 1Ø24
  inam = name
  icre = creator
  IF debug = 1 THEN
```

```
      SAY lin.j
   j = j + 1
   lin.j =  ' '
   j = j + 1
   lin.j =  ' Space    in KB and equivalent of HI-ALLOC-RBA  'tharba
   j = j + 1
   lin.j =  '  '
   j = j + 1
RETURN
/*********************************************************************/
/* Index space heading - partition                                   */
/*********************************************************************/
IX_HEAD_PART:
     C1 = "IXCREAT"
     C2 = "IXNAME "
     C3 = "PQTY"
     C4 = "SQTY"
     C5 = "FREEPAGE"
     C6 = "PCTFREE"
     C7 = "PART"
     lin.j = ' Info from SYSIBM.SYSINDEXPART  ' ||,
             ' (One row for each unpartitioned index '
     j = j + 1
     lin.j = ' and one row for each partition of a partitioned IX)'
     j = j + 1
     lin.j = ' '||c1||COPIES(' ',(8-LENGTH(c1)))||' ',
             ||c2||COPIES(' ',(8-LENGTH(c2)))||' ',
             ||c3||COPIES(' ',(8-LENGTH(c3)))||' ',
             ||c4||COPIES(' ',(8-LENGTH(c4)))||' ',
             ||c5||COPIES(' ',(8-LENGTH(c5)))||' ',
             ||c6||COPIES(' ',(8-LENGTH(c6)))||' ',
             ||c7||COPIES(' ',(8-LENGTH(c7)))
     IF debug = 1 THEN
        SAY lin.j
     j = j + 1
RETURN
/*********************************************************************/
/* Index space detail - partition                                    */
/*********************************************************************/
IX_DETAIL_PART:
 lin.j =  ' '||ixcreato||COPIES(' ',(8-LENGTH(ixcreato)))||' '||,
          ixname||COPIES(' ',(8-LENGTH(ixname)))||' '||,
          pqty||COPIES(' ',(8-LENGTH(pqty)))||' '||,
          sqty||COPIES(' ',(8-LENGTH(sqty)))||' '||,
          freepage||COPIES(' ',(8-LENGTH(freepage)))||' '||,
          pctfree||COPIES(' ',(8-LENGTH(pctfree)))||' '||,
          partitio||COPIES(' ',(8-LENGTH(partitio)))
  IF debug = 1 THEN
     SAY lin.j
  j = j + 1
```

```
  /*
  lin.j =  ' '
  j = j + 1
  */
RETURN
/*********************************************************************/
/* Write out LISTCAT information                                     */
/*********************************************************************/
LST_INFO:
    out = 'DBAPRT'
    'ALLOCATE FI('out') SPACE(10,10) UNIT(VIO) TRACK LRECL(133) ,
              RECFM(V B A) BLKSIZE(3059) REUSE '
    'LISTCAT ENTRY('workdsn') OUTFILE('out') ALL '
    lrc = rc
    'EXECIO * DISKR 'out' (STEM rec. FINIS '
    ADDRESS TSO 'FREE FI('out') '
RETURN
/*********************************************************************/
/* Write out information from DB2 catalog tables                     */
/*********************************************************************/
WRT_LINES:
  ADDRESS "ISPEXEC " " VGET (ZSCREEN) ASIS "
  tdat = ".D"||SUBSTR(DATE(E),1,2)||SUBSTR(DATE(E),4,2)||,
         SUBSTR(DATE(E),7,2)
  ttim = ".T"||SUBSTR(TIME(),1,2)||SUBSTR(TIME(),4,2)||,
         SUBSTR(TIME(),7,2)
  wdsn = USERID()||".LST"||ZSCREEN||tdat||ttim
  out1 = 'DBAPRT1'
  'ALLOCATE DA('''wdsn''') FI('out1') SPACE(10,10) TRACK LRECL(133) ,
            RECFM(V B A) BLKSIZE(3059) MOD '
  'EXECIO * DISKW 'out1' (STEM lin. FINIS '
  'EXECIO * DISKW 'out1' (STEM rec. FINIS '
  IF rc = 0 THEN
  DO
    ADDRESS ISPEXEC
    'CONTROL ERRORS RETURN '
    bdd = out1
    'LMINIT DATAID(ddvar)  DDNAME('bdd') '
    'BROWSE DATAID('ddvar') '
    'LMFREE DATAID('ddvar') '
  END
  ADDRESS TSO 'FREE FI('out1') DELETE '
RETURN
```


VSAM I/O operations – a review

The two charts presented here are designed as a memory jogger and checklist of useful VSAM I/O operations. The first provides a summary of programming verbs and macros; the second shows all the things you can do in a program with VSAM files, and indicates which macro/verb/statement you would use in Assembler, COBOL, and CICS Command Level to do each function, and the types of VSAM file to which the verbs apply.

Concept	Assembler	COBOL verb	CICS command
Open a dataset	OPEN	OPEN	not in user program
Close a dataset	CLOSE	CLOSE	not in user program
Direct read	GET	READ	READ
Output a new record	PUT	WRITE	WRITE
Update a record	PUT	REWRITE	REWRITE
Delete a record	ERASE	DELETE	DELETE
Release exclusive control	ENDREQ	any I/O on file	UNLOCK
Start browse	POINT	START	STARTBR
Browse forward	GET	READ NEXT	READNEXT
Browse backward	GET	not available	READPREV
Reset start of browse	POINT	START	RESETBR
End browse	not required	not required	ENDBR
Retrieve an index record	GETIX	not available	not available
Output an index record	PUTIX	not available	not available
Verify end of file	VERIFY	not available	not available

Figure 1: I/O operations programming verb/macro summary

The following points should be made about Figure 2:

- For Assembler, the words following the macro name indicate the RPL OPTCD settings.
- COBOL refers to programs written in COBOL for batch processing and running under TSO. The words in parentheses indicate phrases that must be coded in other statements describing the same dataset.
- CICS refers to command-level programs written in COBOL, PL/I, Assembler, or RPG II. All command-level statements begin EXEC CICS and end END-EXEC.

Concept	Assembler	COBOL Verb	CICS Command	KSDS	RRDS	ESDS
Browsing						
initial positioning by key	POINT	START	STARTBR EQUAL	X		
initial positioning by RRN	POINT	START	STARTBR RRN		X	
initial positioning by RBA	POINT	START	STARTBR RBA	X		X
posit. for backward browsing	POINT BWD	not available	same as above	X	X	X
browsing forward	GET SEQ FWD	READ NEXT	READNEXT	X	X	X
browsing backward	GET KEY SEQ BWD	not available	READPREV	X	X	X
repositioning while browsing	POINT	START	RESETBR EQUAL	X	X	X
finish browsing	not necessary	not necessary	ENDBR	X	X	X
position 'closest' key	POINT KGE	START KEY GREATER	STARTBR GTEQ	X		
position partial key	POINT GEN	START USING KEY	STARTBR GENERIC	X		
Input						
key sequence retrieval	GET KEY SEQ	READ NEXT	must Browse	X		
RRN sequence retrieval	GET KEY SEQ	READ NEXT	must Browse		X	
RBA sequence retrieval	GET ADR SEQ	READ NEXT	must Browse	X		X
sequential backward	GET SEQ BWD	not available	must Browse	X	X	X
skip sequential	GET KEY SKP	not available	READNEXT RIDFLD	X	X	
direct retrieval by full key	GET KEY DIR	READ KEY	READ RIDFLD	X		
direct retrieval by RRN	GET KEY DIR	READ KEY	READ RIDFLD RRN		X	
direct retrieval by RBA	GET ADR DIR	READ (RELATIVE KEY)	READ RIDFLD RBA	X		X
control interval retrieval	GET CNV DIR	not available	not available	X	X	X
retrieval by partial key	GET KEY DIR GEN	must Browse	READ GENERIC	X		
retrieving for update	GET UPD	READ (OPEN I-0)	READ UPDATE	X	X	X
Output						
insert a new record	PUT KEY DIR NUP	WRITE	WRITE	X	X	
mass insertion of records	PUT KEY SEQ NUP	not available	WRITE MASSINSERT	X		
terminate mass insert	not required	not applicable	UNLOCK	X		
skip sequential insertion	PUT KEY SKP NUP	not available	not available	X	X	
add a new record to the end	PUT SEQ	WRITE	WRITE	X		X
updating record just read	PUT UPD	REWRITE	REWRITE	X	X	X
cancel update	ENDREQ	any Input operation	UNLOCK	X	X	X
Deleting						
after seq. read for update	ERASE UPD	DELETE	DELETE	X	X	
after keyed read for update	ERASE UPD	not available	DELETE	X	X	
directly by key	not available	DELETE	DELETE RIDFLD	X	X	
all with same partial key	not available	not available	DELETE GENERIC	X		

Figure 2: I/O operations – concept and implementation

Jon E Pearkins
(Canada)

Contributing to *VSAM Update*

In addition to *VSAM Update*, the Xephon family of Update publications now includes *CICS Update*, *VM Update*, *MVS Update*, *TCP/SNA Update*, *VSE Update*, *DB2 Update*, *RACF Update*, *AIX Update*, *Domino Update*, *NT Update*, *SQL Server Update*, and *Web Update*. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with VSAM or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it. For a copy of our *Notes for Contributors*, which explains the terms and conditions under which we publish articles, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her on 100336.1412@compuserve.com.

VSAM news

Macro 4 has announced Version 3.3 of its VSAMTUNE VSAM data management package, which supports the latest version of OS/390 and is year 2000-compliant.

Improved facilities let users write tailored reports and run them as either a batch job or on an *ad hoc* basis. The Alternate Index Builder is claimed to reduce file search times and improve file access dramatically. It also claims better dynamic buffering and performance tuning plus a file tuning early-warning system.

The software is part of the company's Data Management Series, which also includes DATABACK for data back-up and recovery, VSAMLITE for compressing VSE data by up to 80%, and FOREMAN for forward recovery of datasets.

For further information contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, W Sussex, RH10 4SS, UK.
Tel: (01293) 886060.
Macro 4, 35 Waterview Blvd, PO Box 202, Parsippany, NJ 07054-0292, USA.
Tel: (973) 402 8000.
URL: http://www.macro4.com.

* * *

IBM has announced a performance boost for VSE/ESA TCP/IP through the addition of NFS Version 2 protocol support. Available on OS/390, AIX, and OS/2, it enables a VSE/ESA system to act as a file server. Systems that have the NFS Version 2 client function installed can access VSE/VSAM files, VSE/ESA Library files, and VSE/POWER queues as if those files were residing locally.

For further information, contact your local IBM representative.

* * *

IBM has previewed Version 1.5 of DFSMS/MVS, which will improve the performance of shared catalogs in a parallel sysplex environment, and provide the ability to maintain multiple HSMplexes within a single GRSplex. Each HSMsubplex will have its own set of control datasets and manage its own set of user catalogs. Addressing previous growth limitations, the DFSMShsm control datasets will take advantage of VSAM extended addressing introduced in DFSMS/MVS 1.3.

For further information, contact your local IBM representative.

* * *

Neon Systems has announced Version 4.1 of ShadowWebserver, the MVS Web server component of ShadowDirect, providing Web access directly to mainframe data and transactions without intermediate application tiers.

Providing a production Web-enablement infrastructure for OS/390 and MVS, the software incorporates DB2 for OS/390, IMS/DB, IMS/TM, CICS, and Adabas, as well as VSAM and other mainframe sources into Web-based applications.

For further information contact:
Neon Systems, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://www.neonsys.com.