207

MVS

December 2003

update

In this issue

MVS Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

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

Editor

Trevor Eddolls

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

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

Printed in England.

A simple ISP productivity aid

We are always looking for ways to work smarter and quicker. In this short article we offer a simple REXX program that does just that. Three of the most common activities that most of us perform under TSO are browsing a dataset, editing a dataset, and submitting a job. Our REXX program will help us perform all three of these functions. There are two other components besides the REXX program that have to be put into place to enable this: command table entries and the EDPANEL.

COMMAND TABLE ENTRIES

These are the entries that you need to make in the ISPF SITECMDS table. You must make the following entries in that table:

```
ED        2  SELECT CMD(%ED ED &ZPARM) NEWAPPL(ISR)
BR        2  SELECT CMD(%ED BR &ZPARM) NEWAPPL(ISR)
SJ        2  SELECT CMD(%ED SJ &ZPARM) NEWAPPL(ISR)
```

Note that all three entries are pointing at the same command – %ED.

EDPANEL

The second item that needs to be placed in your local ISPF panel library is the EDPANEL, which is shown below:

```
)ATTR
% TYPE(TEXT) INTENS(HIGH) ATTN(OFF) SKIP(ON)
+ TYPE(TEXT) INTENS(LOW)  ATTN(OFF) SKIP(ON)
@ TYPE(INPUT) INTENS(HIGH) CAPS(ON) PADC(_) ATTN(OFF)
! TYPE(INPUT) INTENS(LOW)  CAPS(ON) PADC(_) ATTN(OFF) COLOR(TURQ)
{ TYPE(TEXT) COLOR(WHITE) HILITE(REVERSE) INTENS(HIGH)
)BODY EXPAND($$)
%${$<< Alias Settings for ED/BR/SJ Commands >> $+$ +
%OPTION===>_ZCMD                                                +
+
+ALIAS  FULLY QUALIFIED DATASET NAME  ALIAS FULLY QUALIFIED DATASET NAME
+
@A1    !D1                          + @A18   !D18
```

```
@A2    !D2                          + @A19   !D19
@A3    !D3                          + @A2Ø   !D2Ø
@A4    !D4                          + @A21   !D21
@A5    !D5                          + @A22   !D22
@A6    !D6                          + @A23   !D23
@A7    !D7                          + @A24   !D24
@A8    !D8                          + @A25   !D25
@A9    !D9                          + @A26   !D26
@A1Ø   !D1Ø                         + @A27   !D27
@A11   !D11                         + @A28   !D28
@A12   !D12                         + @A29   !D29
@A13   !D13                         + @A3Ø   !D3Ø
@A14   !D14                         + @A31   !D31
@A15   !D15                         + @A32   !D32
@A16   !D16                         + @A33   !D33
@A17   !D17                         + @A34   !D34
+
)INIT
 .CURSOR = ZCMD
)PROC
 VPUT(A1 A2 A3 A4 A5 A6 A7 A8 A9 A1Ø) PROFILE
 VPUT(A11 A12 A13 A14 A15 A16 A17 A18 A19 A2Ø) PROFILE
 VPUT(A21 A22 A23 A24 A25 A26 A27 A28 A29 A3Ø) PROFILE
 VPUT(A31 A32 A33 A34) PROFILE
 VPUT(D1 D2 D3 D4 D5 D6 D7 D8 D9 D1Ø) PROFILE
 VPUT(D11 D12 D13 D14 D15 D16 D17 D18 D19 D2Ø) PROFILE
 VPUT(D21 D22 D23 D24 D25 D26 D27 D28 D29 D3Ø) PROFILE
 VPUT(D31 D32 D33 D34) PROFILE
)END
```

This is a very straightforward panel that is used to create and maintain aliases for datasets. It allows us to associate a short name with a full dataset. Typical entries might look like the following:

```
JCL  hlq.my.jcl.library
SAS  hlq.my.sas.library
```

As implemented, up to 34 entries can be created. Note that this information is saved in your ISPF profile so that it will be preserved across TSO sessions.

ED REXX PROGRAM

The last item that we need to consider is the REXX program itself. We have kept the code very simple and straightforward. If you invoke the program by entering BR, ED, or SJ without any

arguments or with ? being the only argument, the EDPANEL panel will be displayed, so that you can maintain the aliases and their associated datasets. If you invoke it with only a single passed parameter, it is treated as the alias. If you invoke the program with two arguments, the second of the two arguments is treated as the member name. Several examples are given below. Once you have all of the pieces in their respective locations, you can invoke the ED REXX program from any command line in TSO.

```
/* REXX EXEC  */
parse upper arg FUNC NAME MBR
if NAME = "?" | NAME = "" then
  "ISPEXEC DISPLAY PANEL(EDPANEL)"
else
  do
    X = 1
    do until NAME = ALIAS | X > 34
      AL = "A"||X ; DS = "D"||X ; CN = "C"||X
      "ISPEXEC VGET ("AL DS CN") PROFILE"
      ALIAS = value('AL') ; ALIAS = value(ALIAS)
      X = X + 1
    end
    if X > 34 then
      do
        ZEDSMSG = "Invalid alias - "||NAME
        ZEDLMSG = "Alias does not exist, or you have a spelling error"
        "ISPEXEC SETMSG MSG(ISRZØØ1)"
        exit
      end
    else
    DS1 = value('DS') ; DSN = value(DS1)
    CT1 = value('CN') ; CNT = value(CT1)
    if CNT = "" then
      CNT = Ø
    else
    CNT = CNT + 1
    CNT_DATA = CT1 ; interpret CNT_DATA ' = CNT'
    "ISPEXEC VPUT ("CN") PROFILE"
    code = LISTDSI(DSN)
    if code > Ø then
      do
        ZEDSMSG = "Invalid Dsn - "||NAME
        ZEDLMSG = "Dataset name associated with alias does not exist"
        "ISPEXEC SETMSG MSG(ISRZØØ1)"
        exit
      end
```

```
    if FUNC = "ED" then
      do
        if MBR = "" then
          "ISPEXEC EDIT DATASET("DSN")"
        else
          "ISPEXEC EDIT DATASET("DSN"("MBR")"
        if lastcc > Ø then "ISPEXEC SETMSG MSG(ISRZØØ2)"
      end
    if FUNC = "BR" then
      do
        "ISPEXEC BROWSE DATASET("DSN")"
        if lastcc > Ø then "ISPEXEC SETMSG MSG(ISRZØØ2)"
      end
    if FUNC = "SJ" then
      do
        ADDRESS "TSO"
        "SUBMIT "DSN"("MBR")"
        if lastcc > Ø then "ISPEXEC SETMSG MSG(ISRZØØ2)"
      end
end
exit
```

Here are some examples:

- ED – display EDPANEL
- BR ? – display EDPANEL
- ED JCL – edit the dataset associated with JCL
- BR SAS – browse the dataset associated with SAS
- SJ JCL BR14 – submit the BR14 member of the JCL dataset.

DFHSM automatic TAPECOPY

It is good practice to hold a copy of all the cartridges managed by the HSM, particularly those containing the data migrated to level 2.

This operation is carried out by using the command **TAPECOPY ML2** (see *DFSMShsm Storage Administrator Reference*, SH21-

1075). However, this command will cause *all* of the cartridges without an alternative volume to be copied, and each will be assigned a PRIVAT output volser.

Instead, it would be better to limit the operation, in the input, to only certain cartridges, and to specify in the output the VOLSER for the cartridges. This can be done using the simple HSMNOAL CLIST. With this CLIST, it is not necessary to know which cartridges have an alternative copy and which do not. It is also possible to:

- Select in the input the prefix or the volume (1-6 characters) of the cartridges (original volumes) to copy.
- Select in the output the prefix or the volume (with the same number of characters) of the cartridges (alternate volumes).
- Maintain the same alphanumeric suffix for OVOL and AVOL.

To recapitulate:

- If you want to copy all HSM ML2 tapes that do not have an alternate volume and you don't want to have a specific VOLSER in the output, just use **HSEND TAPECOPY ML2**.
- If you want to copy only the cartridge HSMnnn and label the alternate XSMnnn, enter the command **TSO %HSMNOAL HSM XSM**.
- If you want to copy the cartridge DFHSM1 on an alternate volume ALTERN, enter the command **TSO %HSMNOAL DFHSM1 ALTERN**.

HSMNOAL CLIST

```
PROC 2 OVOLPREF AVOLPREF DEBUG
/*- SET UP FOR DEBUG IF REQUESTED ----------------------------------*/
  CONTROL NOMSG NOLIST NOFLUSH END(ENDO) NOCONLIST NOPROMPT
  IF &DEBUG  =  DEBUG THEN +
   CONTROL MSG LIST NOFLUSH END(ENDO) PROMPT SYMLIST CONLIST
/*- END OF SET UP --------------------------------------------------*/
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
/*                                                                 */
/* HSMNOAL :  LIST ML2 TAPE WITHOUT ALTERNATE VOLUME               */
```

```
/*            AND SUBMIT TAPECOPY FOR A SUBSET OF THEM.              */
/*            DFSMSHSM MUST BE ACTIVE AND YOUR USERID MUST BE        */
/*            AUTHORIZED TO ISSUE 'HSEND' COMMANDS.                  */
/*                                                                   */
/* PARM:   OVOLPREF ==> REQUIRED. PREFIX OF TAPE VOLUME IN INPUT     */
/*                      (ORIGINAL VOLUME)                            */
/*         AVOLPREF ==> REQUIRED. PREFIX OF TAPE VOLUME IN OUTPUT    */
/*                      (ALTERNATE VOLUME)                           */
/*         DEBUG    ==> IF SPECIFIED, LET YOU SEE ALL CLIST MSGS     */
/*                                                                   */
/* HOW TO USE: %HSMNOAL HSM XSM  ==> COPY ALL 'HSM' PREFIXED TAPE    */
/*                                   TO 'XSM' ALTERNATE VOLUMES      */
/*                                                                   */
/*             %HSMNOAL H X      ==> COPY ALL 'H' PREFIXED TAPE VOL  */
/*                                   TO 'X' PREFIXED AVOLS           */
/*                                                                   */
/*             YOU MAY ALSO SUPPLY A FULLY QUALIFIED VOLUME (6 CHAR):*/
/*             %HSMNOAL DFH555 ALT797   ==>  COPY ONLY HSM555 OVOL   */
/*                                      TO ALT797 ALTERNATE VOLUME   */
/*                                                                   */
/* EXIT CODES: Ø  OK, TAPECOPY SUCCESSFULLY SUBMITTED                */
/*             4  NO TAPECOPY SUBMITTED BECAUSE OVOL(S) NOT MATCHING */
/*             8  OVOLPREF AND AVOLPREF ARE EQUAL (NOT PERMITTED)    */
/*            12  OVOLPREF AND AVOLPREF LENGTHS ARE DIFFERENT        */
/*                                                                   */
/* . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . */
 IF &OVOLPREF = &AVOLPREF THEN DO
 WRITE OVOLPREF AND AVOLPREF CANNOT BE THE SAME!
       EXIT CODE(8)
       ENDO
 SET &SUFF=SUFFIX
 SET &AVOLL=&LENGTH(&AVOLPREF)
 SET &OVOLL=&LENGTH(&OVOLPREF)
 IF &OVOLL NE &AVOLL THEN DO
 WRITE OVOLPREF AND AVOLPREF LENGTH MUST BE THE SAME!
       EXIT CODE(12)
       ENDO
 DEL '&SYSUID..NOAL'
 HSEND WAIT LIST TTOC +
  SELECT(ML2 NOALTERNATEVOL) OUTD('&SYSUID..NOAL')
ALLOC F(LD) DA('&SYSUID..NOAL') SHR REU
OPENFILE LD
 ERROR GOTO FINE
  IF &OVOLL=6 THEN +
            SET &SUFF=
 SET &R=5-&OVOLL
INCR:+
GETFILE LD
   SET &L = &SYSINDEX(&OVOLPREF,&SUBSTR(1:1Ø,&LD))
   IF &L EQ Ø THEN GOTO INCR
```

```
   IF &SUFF= THEN GOTO TAPC
   SET &SUFF=&SUBSTR(&L+&OVOLL:&L+&OVOLL+&R,&LD)
   SET &L = &SYSINDEX(*NONE*,&LD)
   IF &L EQ Ø THEN GOTO INCR
TAPC:+
   SET ALM1=OK
WRITE TAPECOPY OVOLS(&OVOLPREF&SUFF) AVOLS(&AVOLPREF&SUFF) SUBMITTED...
HSEND TAPECOPY OVOLS(&OVOLPREF&SUFF) AVOLS(&AVOLPREF&SUFF)
  IF &OVOLL=6 THEN GOTO FINE
GOTO INCR
FINE:+
 IF &ALM1 NE OK THEN DO
WRITE OVOLPREF=&OVOLPREF NOT FOUND IN HSM LIST, NO TAPECOPY SUBMITTED
       EXIT CODE(4)
                  ENDO
CLOSFILE LD
FREE F(LD)
       EXIT CODE(Ø)
```

Alberto Mungai
Senior Systems Programmer (Italy)

Checking the validity of mounts needed by a job

Network File System (NFS) is becoming a popular replacement for internal use of NDM and FTP. The reasoning is, why waste time doing a file transfer between 'trusted' systems when you can read and write the data from the original source?

In a nutshell, NFS is a set of services and a network protocol for accessing remote file systems over TCP/IP. There are two components to NFS: the NFS Server and the NFS Client. The NFS Server can accept MOUNT requests from external systems for access to its local disk. The NFS Client can request access to another system's disk. MVS supports both NFS Server and NFS Client. As a replacement for internal NDM and FTP staging requests on behalf of MVS batch jobs, MVS will be exploiting the NFS Client.

MVS has supported NFS for several years, but its popularity has grown recently since it now works almost identically to the

traditional Unix NFS. Basically, disks on any system can be mounted on another system and used as if they were local. In a traditional sysplex, shared DASD and GRS takes care of this for us. If a non-MVS system is involved in the mix, sysplex and shared DASD are of limited value, since most other operating systems are ASCII and maintain file systems that use a directory structure. This required a few things in MVS, ie:

- A mount command to identify what to mount and where to mount it.
- ASCII to EBCDIC translation services.
- JCL techniques to access the files in the remote file system.

My early experiments with NFS involved mounting a Sun Solaris system and an HP/UX system to MVS and running a batch IEBGENER to copy data between the two systems. This kind of thinking allows the more robust job scheduling packages on MVS to deal with critical batch scheduling issues that are far beyond what **cron** or Unix-based scheduling packages can handle.

The NFS mount command is a valid TSO command that identifies the mount name, the mountpoint, the location of the data, and some additional parameters. NFS piggybacks on the built-in ASCII to EBCDIC translation tables in TCP/IP. Remote file systems are mounted on the Unix Systems Services (USS) 'side' of MVS. USS is the Unix 'personality' of MVS and supports the Hierarchical File System and NFS. Batch JCL has been enhanced to provide new DD statement options for navigating file systems. The PATH, PATHMODE, PATHOPTS, PATHDISP, and FILEDATA parameters allow JCL to work directly with data under USS or on a remote file system.

There are a few techniques for managing NFS mounts. A mount is a transient thing. It will exist from the time someone explicitly issues a MOUNT command until someone issues an UNMOUNT command (or the implicit UNMOUNT that occurs at MVS shutdown). In some shops the MOUNT commands are placed in the SYS1.PARMLIB(BPXPRM*xx*) member and re-issued every

time an IPL occurs (very common for HFS files). Like the Unix counterpart, MVS NFS also provides an optional AUTOMOUNT facility through the use of a 'map' file. Additionally, in a Parallel Sysplex, mounts can be 'shared' and the AUTOMOVE feature can be used to allow a designated system to take over a MOUNT if the original owner is shut down for maintenance. Even though several mechanisms are available on both sides of a MOUNT to support re-establishing a mount after maintenance windows, it is still possible for an MVS batch job to run at a time when a mount is 'stale' or not yet available. Finally, if NFS becomes a wholesale replacement for internal NDMs and FTPs, the maintenance issues involved with the inventory of possible mounts can become unwieldy.

This led to the creation of the NFSCHECK utility. NFSCHECK is a REXX EXEC that runs in batch and will check the validity of the mounts needed by a job. If the mount is found to be missing, NFSCHECK will attempt to MOUNT it. When the MOUNT is found to be valid or the MOUNT occurs without incident, the EXEC simply returns a zero return code and the job continues. If the MOUNT fails, the EXEC will terminate with a non-zero return code that the job can use in standard condition code processing to minimize re-run issues. NFSCHECK will also write some diagnostic messages and issue a WTO.

NFSCHECK also performs a test to see whether the data is actually accessible. It will issue a USS **readdir** command against the mountpoint to ensure that the data is accessible from MVS. Unfortunately, I have this line commented out in the code because of a bug in z/OS 1.3 that has not been resolved yet. I did confirm that this worked under OS/390 2.10. Simply uncomment the line with the **readdir** command to re-activate this test. If you try this and get an S0C4 during the run, you have the same bug on your system. You can also confirm this by using ISHELL to list the directory for the same mountpoint (ISHELL uses the same readdir command from REXX).

The JCL for NFSCHECK is very simple:

```
//NFSCHECK EXEC PGM=IKJEFTØ1,PARM=NFSCHECK
```

```
//SYSEXEC  DD   DSN=rexx.pds,DISP=SHR
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//MOUNTS   DD   DSN=your.parm.pds(mountmem),DISP=SHR
```

DD statements required for NFSCHECK are:

- SYSEXEC – the PDS containing the NFSCHECK REXX EXEC.
- SYSTSPRT – default SYSOUT for the TSO TMP.
- SYSTSIN – default input source for TSO TMP (DUMMY).
- MOUNTS – syntactically valid MOUNT commands.

Return codes for NFSCHECK are:

- 00 – everything worked, the mount already existed or was mounted successfully.
- 12 – NFS path not found, could be a misspelling or a permission problem.
- 98 – MOUNTS DD is not readable.
- 99 – MOUNTS DD is missing.
- *XX* – all other RCs are valid USS return codes found in the *USS Messages and Codes* manual, related to mount problems.

The MOUNTS DD is used to point to the syntactically valid NFS MOUNT command(s) for the mounts required by the job. Multiple MOUNT commands can be included in a single step. The syntax for the input dataset is identical to the BPXPRM*xx* member from PARMLIB. TSO command syntax is also tolerated (trailing '-' for continuation lines).

Here is a single mount example:

```
MOUNT FILESYSTEM('NFSTEST') TYPE(NFS) MOUNTPOINT('/NFS/TEST1') -
PARM('SRVØØ1:/vol/volØ/TEST,PROTO(UDP), -
XLAT(Y),HARD')
```

Here is an example with multiple mounts and comments:

```
* Sample mounts with asterisk in column style comments
* Blank lines are OK too

/* Another style of comments supported */

   MOUNT FILESYSTEM('fs1')
         MODE(RDWR)
         TYPE(NFS)
         MOUNTPOINT('/nfstest/fsØ1')
   PARM('srvØØ2:/vol/volØ/fsØ1,proto(udp),xlat(Y),HARD')

   MOUNT FILESYSTEM('fs2')
         MODE(RDWR)
         TYPE(NFS)
         MOUNTPOINT('/nfstest/fsØ2')
   PARM('srvØØ2:/vol/volØ/fsØ2,proto(udp),xlat(Y),HARD')

   MOUNT FILESYSTEM('fs3')
         MODE(RDWR)
         TYPE(NFS)
         MOUNTPOINT('/nfstest/fsØ3')
   PARM('srvØØ2:/vol/volØ/fsØ3,proto(udp),xlat(Y),HARD')
```

The output from the NFSCHECK EXEC will look like this after a successful run where no action was needed:

```
NFSCHECK found 3 mounts to confirm

NFS Mount: FS1     Path: /nfstest/fsØ1          is accessible from SYØ1
NFS Mount: FS2     Path: /nfstest/fsØ2          is accessible from SYØ1
NFS Mount: FS3     Path: /nfstest/fsØ3          is accessible from SYØ1

NFSCHECK confirmed 3 of 3 paths accessible

NFSCHECK output when a MOUNT was issued:

NFSCHECK found 1 mounts to confirm


NFS Mount: NFSTEST /NFS/TEST1 was successfully mounted with the
following MOUNT command:

MOUNT FILESYSTEM('NFSTEST') TYPE(NFS)  MOUNTPOINT('/NFS/TEST1')
PARM('SRVØØ1:/vol/volØ/TEST,PROTO(UDP),XLAT(Y),HARD')

NFSCHECK confirmed 1 of 1 paths accessible
```

Sample NFSCHECK output when a mount fails:

```
NFSCHECK found 1 mounts to confirm
```

```
BPXF135E RETURN CODE ØØØØØØ79, REASON CODE Ø55BØØ5C.  THE MOUNT FAILED
FOR FILE

NFS Mount: NFSTEST mount point /NFS/TEST1 has a bad status on SYØ3,
RC=12

MOUNT FILESYSTEM('NFSTEST') TYPE(NFS)  MOUNTPOINT('/NFS/TEST1')
PARM('SRVØØ1:/VOL/volØ/TEST,PROTO(UDP),XLAT(Y),HARD')

NFS Mount: NFSTEST mount point /NFS/TEST1 has a bad status on SYØ3,
RC=12
```

Remember that mounts can fail for many reasons on both sides of the request. Don't forget that the problem can be on the remote system because of permissions or other server-related problems. NFSCHECK will also issue a WTO when a MOUNT fails. Look in the SYSLOG for this message and it will occur adjacent to a BPX message with more diagnostic information:

```
+NFSCHECK/<jobname>: NFS Mount: NFSTEST mount point /NFS/TEST1 has a bad
status on SYØ3, RC=12 <userid>
```

NFSCHECK SOURCE

```
/*********************************************************************/
/*                              REXX                                 */
/*********************************************************************/
/* Purpose: NFSCHECK                                                 */
/*-------------------------------------------------------------------*/
/* Syntax:  NFSCHECK                                                 */
/*-------------------------------------------------------------------*/
/* Parms: DEBUG      - Turn on a REXX TRACE (optional)               */
/*-------------------------------------------------------------------*/
/* Notes:                                                            */
/* Return Codes:     Ø  NFS path is mounted, available and readable  */
/*                  12  NFS path was not found, possibly misspelled  */
/*                  98  MOUNTS DD is unreadable                      */
/*                  99  MOUNTS DD is missing                         */
/*                                                                   */
/*                  All other non-zero return codes are the actual   */
/*                  USS RETVALs from the USS API call.  See the USS  */
/*                  Using REXX and OS/39Ø UNIX Systems Service Guide */
/*                  SC28-19Ø5 - Appendix A for RC to literal map     */
/*                                                                   */
/* Sample JCL:                                                       */
/*    //NFSCHECK EXEC PGM=IKJEFTØ1,PARM=NFSCHECK                     */
```

```
/*    //SYSEXEC  DD   DSN=rexx.pds,DISP=SHR                          */
/*    //SYSTSPRT DD   SYSOUT=*                                       */
/*    //MOUNTS   DD   DSN=your.mount.list,DISP=SHR                   */
/*    //SYSTSIN  DD   DUMMY                                          */
/*                                                                   */
/* To avoid upper/lower case issues, set CAPS OFF in JCL             */
/*                                                                   */
/* Format of MOUNTS DD input:                                        */
/*                                                                   */
/* '*' in column 1 is a comment                                      */
/*                                                                   */
/* Syntactically valid MOUNT statement, like in BPXPRMxx members     */
/*                                                                   */
/* ---------------------- Top of Data ------------------------     */
/* * FS1                                                             */
/* MOUNT FILESYSTEM('fs1')                                           */
/*       MODE(RDWR)                                                  */
/*       TYPE(NFS)                                                   */
/*       MOUNTPOINT('/netapp/fsØ1')                                  */
/* PARM('fØØØ1cdc:/vol/volØ/fsØ1,proto(udp),xlat(Y),HARD')           */
/* * FS2                                                             */
/* MOUNT FILESYSTEM('fs2')                                           */
/*       MODE(RDWR)                                                  */
/*       TYPE(NFS)                                                   */
/*       MOUNTPOINT('/netapp/fsØ2')                                  */
/* PARM('fØØØ1cdc:/vol/volØ/fsØ2,proto(udp),xlat(Y),HARD')           */
/*                                                                   */
/* ---------------------- Bottom of Data ---------------------    */
/*                                                                   */
/*********************************************************************/
/*                       Change Log                                  */
/*                                                                   */
/*********************************************************************/
/* Accept any parms and ignore them                                  */
/*********************************************************************/
parse arg parms
parse upper source . . execname . execdsn . . execenv .
/*********************************************************************/
/* Determine if DEBUG was requested                                  */
/*********************************************************************/
if parms = 'DEBUG' then trace i
/*********************************************************************/
/* Make sure the MOUNTS DD exists                                    */
/*********************************************************************/
if listdsi('MOUNTS' 'FILE') <> Ø then
    do
     EXITRC = 99
     msg = 'The MOUNTS DD is missing, RC='EXITRC
     signal shutdown
    end
```

```
/*********************************************************************/
/* Read the contents of the MOUNTS DD                                */
/*********************************************************************/
"EXECIO * DISKR MOUNTS (STEM MOUNTS. FINIS"
EXITRC = RC
if EXITRC <> Ø then
    do
     EXITRC = 98
     msg = 'Error reading MOUNTS DD, RC='EXITRC
     signal shutdown
    end
/*********************************************************************/
/* Initialize counters and variables                                 */
/*********************************************************************/
pathcnt = Ø
confirmcnt = Ø
lpar = mvsvar('SYSNAME')
/*********************************************************************/
/* Extract the MOUNTPOINT and format the MOUNT command               */
/*********************************************************************/
do l=1 to mounts.Ø
    line = strip(strip(substr(mounts.l,1,72)),'T','-')
    select
/*********************************************************************/
/* Ignore comment lines and blank lines                              */
/*********************************************************************/
      when substr(line,1,1) = '*' then iterate
      when substr(line,1,2) = '/*' then iterate
      when line = ' ' then iterate
/*********************************************************************/
/* Find the MOUNT command                                            */
/*********************************************************************/
      when word(line,1) = 'MOUNT' then
           do
            pathcnt = pathcnt + 1
            cmd.pathcnt = line
/*********************************************************************/
/* Parse out the filesystem name and the mountpoint                  */
/*********************************************************************/
            parse var line . "FILESYSTEM('" fsname "')" .
            parse var line . "MOUNTPOINT('" mpname "')" .
            fsname.pathcnt = fsname
            mpname.pathcnt = mpname
           end
/*********************************************************************/
/* Find the FILESYSTEM name if on a separate line                    */
/*********************************************************************/
      when pos('FILESYSTEM(',line) > Ø then
           do
            parse var line . "FILESYSTEM('" fsname "')" .
```

```
            parse var line . "MOUNTPOINT('" mpname "')" .
            fsname.pathcnt = fsname
            mpname.pathcnt = mpname
            cmd.pathcnt = cmd.pathcnt line
           end
/*********************************************************************/
/* Find the MOUNTPOINT name if on a separate line                    */
/*********************************************************************/
      when pos('MOUNTPOINT(',line) > 0 then
           do
            parse var line . "MOUNTPOINT('" mpname "')" .
            mpname.pathcnt = mpname
            cmd.pathcnt = cmd.pathcnt line
           end
      otherwise cmd.pathcnt = cmd.pathcnt line
    end
end
/*********************************************************************/
/* Print the mount count                                             */
/*********************************************************************/
say
say execname 'found' pathcnt 'mounts to confirm'
say
/*********************************************************************/
/* Dub the address space                                             */
/*********************************************************************/
if syscalls('ON') <> 0 then
    do
     EXITRC = RC
     msg = 'Error dubbing' execname 'RC='EXITRC
     call wto msg
     signal shutdown
    end
/*********************************************************************/
/* Get the USS Mount Table                                           */
/*********************************************************************/
address SYSCALL 'getmntent m.'
EXITRC = RETVAL
if EXITRC <> 0 then
    do
     EXITRC = RC
     msg = 'Error getting the USS Mount Table information',
           'RETVAL='RETVAL 'ERRNO='ERRNO 'ERRNOJR='ERRNOJR
     call wto msg
     signal shutdown
    end
/*********************************************************************/
/* Loop through the Mount Table looking for NFS mounts               */
/*********************************************************************/
do p=1 to pathcnt
```

```
    do i=1 to m.Ø
/*********************************************************************/
/* If this NFS mount matches the user's NFSNAME then...              */
/*********************************************************************/
       if m.MNTE_FSTYPE.i = 'NFS' & m.MNTE_FSNAME.i = fsname.p then
          do
/*********************************************************************/
/* Clean up the variables for a readable message                     */
/*********************************************************************/
           nfsname = strip(m.MNTE_FSNAME.i)
           EXITRC  = strip(m.MNTE_STATUS.i)
           nfspath = strip(m.MNTE_PATH.i)
/*********************************************************************/
/* If the Mount Status is good, print a message and try to read dir  */
/*********************************************************************/
           if EXITRC = Ø then
              do
/*********************************************************************/
/* Ensure we can read the directory                                  */
/*********************************************************************/
              RETVAL = Ø
/*            address SYSCALL "readdir" nfspath 'file.' 'stat.' */
              EXITRC = RETVAL
/*********************************************************************/
/* If we can't read the directory, print error                       */
/*********************************************************************/
              if EXITRC <> Ø then
                 do
                  msg = 'Error reading NFS Mount' nfsname 'directory',
                         nfspath 'RETVAL='RETVAL 'ERRNO='ERRNO,
                        'ERRNOJR='ERRNOJR
                  call wto msg
                  EXITRC = RETVAL
                  signal shutdown
                 end
/*********************************************************************/
/* It's good                                                         */
/*********************************************************************/
              say 'NFS Mount:' left(nfsname,1Ø) 'Path:',
                   left(nfspath,25) 'is accessible from' lpar
              confirmcnt = confirmcnt + 1
              drop file. stat.
              iterate p
             end
/*********************************************************************/
/* If the initial Mount Table showed a bad Mount Status, remount it  */
/*********************************************************************/
           else
              do
               call nfsmount nfsname nfspath cmd.p
```

```
            end
        end
    end
/*********************************************************************/
/* Mount was not found, so try to mount it                           */
/*********************************************************************/
/* First confirm the Mountpoint directory exists                     */
/*********************************************************************/
    address SYSCALL 'stat' mpname.p 's.'
    EXITRC = RETVAL
    if EXITRC <> 0 | s.ST_TYPE <> S_ISDIR then
       do
        EXITRC = 12
        msg = 'Mountpoint directory' mpname.p 'does not exist on' lpar
        call wto msg
        signal shutdown
       end
/*********************************************************************/
/* Attempt the mount                                                 */
/*********************************************************************/
    EXITRC = nfsmount(fsname.p mpname.p cmd.p)
/*********************************************************************/
/* If the mount failed, bail with a message                          */
/*********************************************************************/
    if EXITRC <> 0 then
       do
        msg = 'Mount for' fsname.p mpname.p 'failed RC='EXITRC
        call wto msg
        signal shutdown
       end
    else
       confirmcnt = confirmcnt + 1
end
/*********************************************************************/
/* If we made it to the end without an error, print the confirm msg  */
/*********************************************************************/
msg = execname 'confirmed' confirmcnt 'of' pathcnt 'paths accessible'
/*********************************************************************/
/* Shutdown                                                          */
/*********************************************************************/
shutdown: say
          say msg
          say
          exit(EXITRC)
/*********************************************************************/
/* WTO                                                               */
/*********************************************************************/
wto: parse arg msg
      jobname = mvsvar('SYMDEF','JOBNAME')
      if length(msg) > 95 then msg = substr(msg,1,95)
```

```
      address TSO "SEND '"execname"/"jobname": "msg"'"
      return
/*********************************************************************/
/* NFS Mount                                                         */
/*********************************************************************/
nfsmount: parse arg mname mpoint mcmd
/*********************************************************************/
/* First attempt to quietly unmount the NFS mount                    */
/*********************************************************************/
          call outtrap 'toss.'
          address TSO "UNMOUNT FILESYSTEM('"mname"') IMMEDIATE"
          call outtrap 'off'
/*********************************************************************/
/* Re-issue the mount command from the MOUNTS DD                     */
/*********************************************************************/
          address TSO mcmd
          EXITRC = RC
          if EXITRC = Ø then
             do
              say
              say 'NFS Mount:' mname mpoint 'was successfully mounted',
                  'with the following MOUNT command:'
              say
              say mcmd
             end
          else
             do
              msg = 'NFS Mount:' mname 'Mountpoint:' mpoint,
                    'has a bad status on' lpar', RC='EXITRC
              say msg
              say
              say mcmd
              call wto msg
              signal shutdown
             end
          return EXITRC
```

Robert Zenuk
Systems Programmer (USA)

Splitting a file across several files

The following program reads an input file and distributes the lines read over several output files. This distribution can be done in two ways – each output file can have a constant number of lines or

a constant number of blocks, where a block is a logical group of lines. In the first case, supposing we have an input file with 20,000 lines, we could create, for example, 20 output files with 1,000 lines each.

The second case is a little more elaborate. Instead of specifying the number of lines each output file should have, we specify the number of blocks. A block is a set of contiguous lines, read from the input file, that end when a line contains a specific string in a fixed position. That line will be considered the last line of the current block. The number of lines within a block can be variable, and can range from 1 to the entire input file (for example, if the string is specified but not found).

Consider the following example: an input file containing movements of customers, with several records for each customer, ending up in a total record:

```
CUST00304 20020601 00786 007767
CUST00304 20020605 01876 000981
          TOTAL          008748
CUST00305 20020323 09813 063737
CUST00305 20020430 12231 000981
CUST00305 20020511 98001 004316
          TOTAL          069034
```

The number of records for each customer is not known beforehand; however, we know that each block (each customer) ends with the word TOTAL in position 11.

If we wish to split such a file into several smaller files, we may want to do it in such a way that each block remains undivided. That is precisely what the program does.

Why perform this division? The main reason is that in MVS we daily produce large files that are later transmitted to Unix and processed by SAP batch jobs. Since SAP jobs can be very time-consuming, a common technique is to have several jobs running the same program in parallel but working on different input files. This program was created precisely to produce those files.

The program needs two input files. One is the file to be split. By default, it is associated with DDname INFILE. The other is a file

containing several parameters for the program. Its default DDname is PARMS.

The output files are created dynamically as needed with a name equal to the input file suffixed by '.Snnn', where nnn ranges from 001 to 200 (the maximum currently allowed).

Using the above example, possible JCL for it would be:

```
//STEP1    EXEC PGM=SPLITFIL
//SYSPRINT DD SYSOUT=*
//STEPLIB  DD DISP=SHR,DSN=loadlib
//INFILE   DD DISP=SHR,DSN=input.file.name
//PARMS    DD *
MAX=00000100       maximum blocks per file
STRING=TOTAL       string to search
POSITION=0011      position of string
TRACKS=0030        primary track allocation for new files
FILES=0050         maximum number of files to create
```

Here, what we say is: the word TOTAL in position 11 of a record means that that record is the last one of the current block. I want to write 100 blocks in each file; I want no more than 50 files; each file has 30 tracks as the primary allocation space.

With these premises, what could happen? If the number of blocks existing in the input file happens to be exactly 5,000, then we have an even distribution of 100 blocks across 50 files. If the number of blocks is less than 5,000, then the last file will probably have fewer than 100 blocks, and probably also fewer than 50 files will be created. But if the number of blocks is more, and since we limited the number of files to 50, then the last file will have over 100 blocks on it. In any case, sysprint will state what files were created and also how many blocks the last file contains.

If I simply wish to split a file with a certain number of lines for each output file, without considering logical blocks, all I have to do is omit the STRING parameter. This way, the program just writes MAX records to each file.

This program uses two Assembler subroutines, DYNALOC1 and DYNALOC7, already published in *MVS Update* (issues 172 and 199, January 2001 and April 2003).

At the beginning of the program there are several variables where you can control the default values for several options. Note, however, that the numeric parameters must have an exact number of characters.

SPLITFIL SOURCE

```
*===============================================================*
* SPLITFIL - Program to split a file across several files.
*            The input file will be split in blocks, where a block
* is a group of lines where the last line contains a specific string
* in a given position. If no string is given, each line is a block.
*
* The number of files created can be less than or equal to the number
* of files specified. In most cases, the last file will contain an
* undefined amount of blocks, which can be less or more than the
* maximum specified, depending on what ends first: the input blocks
* or the output files.
*
* The output files are created dynamically as needed.
* Their names are the same as the input file suffixed by .Snnn where
* nnn ranges from 001 to 200. Other characteristics (recfm, lrecl)
* are also taken fron the input file.
*
* DDnames: INFILE- Input file (sequential with recfm F or V)
*          PARMS - File containing the following parameters in any
*                  order, one per line and left-justified:
*  MAX=nnnnnnnn  - Number of blocks to write per output file
*  STRING=XXXXX  - String whose presence in a record marks the end
*                  of a block (max 30 characteres). If no string is
*                  given, then each record is considered a block.
*  POSITION=nnnn - Position (NOT offset) of string within the record
*  FILES=nnnn    - Maximum number of files to create
*  TRACKS=nnnn   - Primary tracks for new files allocation
*
* External subrotines: DYNALOC1 - Create and alloc new files.
*                    : DYNALOC7 - Get datasetname from DDname.
* (These subroutines were published in MVS UPDATE No. 172 and 199)
*===============================================================*
&PROGRAM SETC  'SPLITFIL'
&MAX1    SETC  '2000'              default block limit per file
&FILES1  SETC  '20'                default number of files
&POSICA1 SETC  '1'                 default string position
&TRACKP1 SETC  '15'                default primary tracks value
&SUF     SETC  '.S'                default suffix for new files
&PARMDD  SETC  'PARMS'             ddname for parameter file
&INPUTDD SETC  'INFILE'            ddname for input file
```

```
&OUTDD   SETC  'DDOUT'              internal ddname used for outfiles
&BUFFER  SETC  '32768'               buffer for record reading
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         USING IHADCB,R11
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     CONTINUA
         DC    CL16' &PROGRAM 2.Ø'
         DC    CL8'&SYSDATE'
*
*=================================================================*
* Read parameters from parms file
*=================================================================*
*
CONTINUA EQU   *
         XR    R7,R7
         OPEN  (SYSPRINT,OUTPUT)
         OPEN  (PARMS1,INPUT)
         LTR   R15,R15
         BNZ   EXITØ
*
GETPARM  EQU   *
         GET   PARMS1,LINHA
         CLC   LINHA(4),=C'MAX='
         BNE   GETP1
         PACK  PACKTEMP,LINHA+4(8)
         CVB   R2,PACKTEMP
         ST    R2,MAX
         LR    R5,R2
         B     GETPARM
*
GETP1    EQU   *
         CLC   LINHA(9),=C'POSITION='
         BNE   GETP2
         PACK  PACKTEMP,LINHA+9(4)
         CVB   R2,PACKTEMP
         S     R2,=F'1'            Turn position into offset
         ST    R2,POSITION
         B     GETPARM
*
GETP2    EQU   *
         CLC   LINHA(7),=C'STRING='
         BNE   GETP3
```

```
         MVC   STRING,LINHA+7
         LA    R4,STRING           Load initial pointer to string
         LR    R3,R4               Keep it in R3
         BAL   R1Ø,FINDSPC
         SR    R4,R3               Calculate length
         ST    R4,STRINGL          Store length
         B     GETPARM
*
GETP3    EQU   *
         CLC   LINHA(7),=C'TRACKS='
         BNE   GETP4
         PACK  PACKTEMP,LINHA+7(4)
         CVB   R2,PACKTEMP
         STH   R2,TRACKPRI
         B     GETPARM
*
GETP4    EQU   *
         CLC   LINHA(6),=C'FILES='
         BNE   GETPARM
         PACK  PACKTEMP,LINHA+6(4)
         CVB   R2,PACKTEMP
         CH    R2,=H'2ØØ'
         BH    ERRO4
         ST    R2,FILES
         B     GETPARM
*
LOOPARMF EQU   *
         CLOSE PARMS1
         STORAGE OBTAIN,                                          X
               ADDR=(R2),                                         X
               LENGTH=&BUFFER
         ST    R2,READADDR
*
OPENENT  EQU   *
         OPEN  (ENTRADA,INPUT)     Open input file
         LTR   R15,R15
         BNZ   ERRO1
         LA    R11,ENTRADA         Address DCB of entrada
         MVC   CAB1DSOR,DCBDSORG   Keep file dcb characteristics
         MVC   CAB1RECF,DCBRECFM   before any reading to have
         MVC   CAB1BLKS,DCBBLKSI   the greatest lrecl and not
         MVC   CAB1LREC,DCBLRECL   the individual records.
*
GETFNAME EQU   *                   Get filename from ddname
         CALL  DYNALOC7,(DDNAME,DSNAME)
         MVC   NEWFNAME,DSNAME     Move DSN for new files
         LA    R4,NEWFNAME
         BAL   R1Ø,FINDSPC         Find first blank after fname
         ST    R4,NEWFNEND         Store its pointer
         L     R1Ø,CAB1BLKS
```

```
         BAL   R1Ø,NEWFILE         Create a new filename
         L     R6,STRINGL          Search string length
         S     R6,=F'1'            Leng -1 for compare
         L     R8,POSITION         string offset in record
         TM    DCBRECFM,DCBRECV    Is RECFM variable?
         BNO   LEITURA             No, jump. (Branch if not ones)
         LA    R8,4(Ø,R8)          Yes, variable, add 4 bytes
         ST    R8,POSITION         to offset and store it.
*
LEITURA  EQU   *                   Read loop
         LA    R11,ENTRADA         Address DCB of Entrada
         GET   ENTRADA,(R2)        Get record
         CLI   NEWFLAG,C'1'        Flag for new file?
         BNE   LEITURA1            No, jump
         BAL   R1Ø,NEWFILE         Yes, alloc new
*
LEITURA1 EQU   *                   Read loop
         LH    R4,DCBLRECL         Get lrecl of the record
         LA    R11,SAIDA           Address DCB of Saida
         STH   R4,DCBLRECL         Put lrecl for saida
         PUT   SAIDA,(R2)          Write output record
         CLI   STRINGL,X'ØØ'       String length is null?
         BE    COMPARA1            Yes, no compare performed.
*
COMPARA  EQU   *                   Compare string
         LR    R9,R2               Copy posicao (R2) to R9
         AR    R9,R8               Add to record location
         EX    R6,EXCOMPAR         Compare string
         BNE   LEITURA             No match, read another
*
COMPARA1 EQU   *                   Compare string
         LA    R7,1(Ø,R7)          Inc counter
         CR    R7,R5               Max attained
         BL    LEITURA             No, continue reading
         MVI   NEWFLAG,C'1'        Flag for new file after reading
         B     LEITURA             Not yet, continue reading
*
EXITØ    EQU   *
         L     R2,READADDR         Load addr of getmained storage
         STORAGE RELEASE,          and release it                    X
               ADDR=(R2),                                          X
               LENGTH=&BUFFER
*
MEESAGE4 EQU   *                   State how many blocks last file has
         BAL   R1Ø,REG7DISP        Call routine to display R7
         MVC   VALOR,ZUNP2         Move display field to message
         PUT   SYSPRINT,MSG1       print message
*
EXIT1    EQU   *
         CLOSE ENTRADA
```

```
         CLOSE SAIDA
         CLOSE SYSPRINT
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         SR    R15,R15
         BR    R14
*
*================================================================*
* Subroutines
*================================================================*
*
FINDSPC  EQU   *                    Find first space or x'ØØ' after
         CLI   Ø(R4),X'4Ø'           a string.
         BE    FINDSPCF              On entry, R4 points to the
         CLI   Ø(R4),X'ØØ'           beginning of string.
         BE    FINDSPCF              On exit, R4, points the first
         LA    R4,1(Ø,R4)            space or low-value.
         B     FINDSPC
FINDSPCF EQU   *
         BR    R1Ø
*
NEWFILE  EQU   *                    Allocate a new file
         MVI   NEWFLAG,C'Ø'          reset new file flag
         L     R1,CURFILE            Load current file number to R1
         LA    R1,1(Ø,R1)            Increment R1
         ST    R1,CURFILE            and store it back
         CLC   CURFILE,FILES         Max files attained?
         BH    NEWFILE1              Yes, jump ahead
         CLOSE SAIDA                 Close file (free dd auto)
         L     R3,TABINPOS           Load current pos table addr
         L     R4,NEWFNEND           Point to end of newfname
         MVC   Ø(2,R4),=C'&SUF'      Add suffix to fname
         MVC   2(3,R4),Ø(R3)         Move 3 chars ftom table to fname
         PUT   SYSPRINT,NEWFNAME
         LA    R3,3(Ø,R3)            Increment 3 bytes
         ST    R3,TABINPOS           Store addr for next
         CALL  DYNALOC1,(NEWDDNAM,NEWFNAME,TRACKPRI,DYBLOCK)
         LTR   R15,R15
         BNZ   ERRO2
         LA    R11,SAIDA             Address SAIDA DCB
         MVC   DCBDSORG,CAB1DSOR     Load DCB values before opening
         MVC   DCBRECFM,CAB1RECF     the new file.
         MVC   DCBBLKSI,CAB1BLKS
         MVC   DCBLRECL,CAB1LREC
         OPEN  (SAIDA,OUTPUT)        Open new file (after DCB loaded)
         LTR   R15,R15
         BNZ   ERRO3
         XR    R7,R7                 Reset block counter
         BR    R1Ø                   Return
*
```

```
NEWFILE1 EQU   *                   Last file
         L     R5,=F'9999999'      remove block limit (set it high)
         BR    R1Ø                 Return
*
REG7DISP EQU   *                   Convert register R7 to display.
         CVD   R7,ZUNP             Convert to decimal
         UNPK  ZUNP2,ZUNP          Unpack from zunp to zunp2
         OI    ZUNP2A,X'FØ'        Remove signal with or
         B     REG7DEND            Jump around storage
ZUNPØ    DS    ØD                  Double align for unpack
ZUNP     DS    CL8                 Decimal field
ZUNP2    DS    ØCL8                Unpacked field (8 bytes)
         DS    CL7
ZUNP2A   DS    C                   Sign byte will be ored with FØ
REG7DEND BR    R1Ø                 Return
*
*================================================================*
* Work areas
*================================================================*
*
ERRO1    EQU   *
         PUT   SYSPRINT,=CL8Ø'ERROR OPENING INPUT FILE'
         B     EXIT1
ERRO2    EQU   *
         PUT   SYSPRINT,=CL8Ø'ERROR ALLOCATING OUTPUT FILE'
         B     EXITØ
ERRO3    EQU   *
         PUT   SYSPRINT,=CL8Ø'ERROR OPENING OUTPUT FILE'
         B     EXITØ
ERRO4    EQU   *
         PUT   SYSPRINT,=CL8Ø'FILES parameter over max allowed (2ØØ)'
         B     EXIT1
MSG1     DC    C'Last file has '
VALOR    DS    CL8
         DC    C' blocks.                                        '
*
ENTRADA  DCB   DSORG=PS,MACRF=(GM),EODAD=EXITØ,                    X
               DDNAME=&INPUTDD
*
SAIDA    DCB   DSORG=PS,MACRF=(PM),                                X
               DDNAME=&OUTDD
*
PARMS1   DCB   DSORG=PS,RECFM=FB,MACRF=(GM),EODAD=LOOPARMF,        X
               LRECL=8Ø,                                           X
               DDNAME=&PARMDD
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),LRECL=8Ø,                       X
               DDNAME=SYSPRINT
*
         DS    ØF
```

```
EXCOMPAR CLC   Ø(Ø,R9),STRING compare string to R9 (record)
*
         LTORG
         DS    ØD            Double align for packed
PACKTEMP DS    CL8           Field for pack instruction
FILES    DC    F'&FILES1'    Max files
CURFILE  DC    F'Ø'          Number of current file
MAX      DC    F'&MAX1'      Max blocks
POSITION DC    F'&POSICA1'   String position (will become offset)
TRACKPRI DS    H'&TRACKP1'   Primary tracks of files to allocate
STRING   DS    CL3Ø          String to search
STRINGL  DC    F'Ø'          String length (zero=no string given)
LINHA    DS    CL8Ø          Line to read parms1
DDNAME   DC    CL8'&INPUTDD' Input DDname
DSNAME   DS    CL44          Input dsname
NEWDDNAM DC    CL8'&OUTDD'   Internal output DDname
NEWFNAME DS    CL44          Output dsname
         DC    CL36' '
NEWFLAG  DS    C
NEWFNEND DS    F             Point after last char in basic newfname
READADDR DS    F             Address of getmained area for all recs.
FIRSTADR DS    F             Address of getmained area for first rec.
FIRSTLEN DS    F             Length (lrecl) of first record
DYBLOCK  DC    H'Ø'          Sequential file (dir blocks zero)
TABINPOS DC    A(TABINCRE)   Table of sequential suffixes (3 chars)
TABINCRE DC    C'ØØ1ØØ2ØØ3ØØ4ØØ5ØØ6ØØ7ØØ8ØØ9Ø1Ø'
         DC    C'Ø11Ø12Ø13Ø14Ø15Ø16Ø17Ø18Ø19Ø2Ø'
         DC    C'Ø21Ø22Ø23Ø24Ø25Ø26Ø27Ø28Ø29Ø3Ø'
         DC    C'Ø31Ø32Ø33Ø34Ø35Ø36Ø37Ø38Ø39Ø4Ø'
         DC    C'Ø41Ø42Ø43Ø44Ø45Ø46Ø47Ø48Ø49Ø5Ø'
         DC    C'Ø51Ø52Ø53Ø54Ø55Ø56Ø57Ø58Ø59Ø6Ø'
         DC    C'Ø61Ø62Ø63Ø64Ø65Ø66Ø67Ø68Ø69Ø7Ø'
         DC    C'Ø71Ø72Ø73Ø74Ø75Ø76Ø77Ø78Ø79Ø8Ø'
         DC    C'Ø81Ø82Ø83Ø84Ø85Ø86Ø87Ø88Ø89Ø9Ø'
         DC    C'Ø91Ø92Ø93Ø94Ø95Ø96Ø97Ø98Ø99ØØ'
         DC    C'1ØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9Ø11Ø'
         DC    C'111Ø121Ø131Ø141Ø151Ø161Ø171Ø181Ø191Ø2Ø'
         DC    C'121Ø221Ø231Ø241Ø251Ø261Ø271Ø281Ø291Ø3Ø'
         DC    C'131Ø321Ø331Ø341Ø351Ø361Ø371Ø381Ø391Ø4Ø'
         DC    C'141Ø421Ø431Ø441Ø451Ø461Ø471Ø481Ø491Ø5Ø'
         DC    C'151Ø521Ø531Ø541Ø551Ø561Ø571Ø581Ø591Ø6Ø'
         DC    C'161Ø621Ø631Ø641Ø651Ø661Ø671Ø681Ø691Ø7Ø'
         DC    C'171Ø721Ø731Ø741Ø751Ø761Ø771Ø781Ø791Ø8Ø'
         DC    C'181Ø821Ø831Ø841Ø851Ø861Ø871Ø881Ø891Ø9Ø'
         DC    C'191Ø921Ø931Ø941Ø951Ø961Ø971Ø981Ø991ØØ'
*
SAVEA    DS    18F
CAB1DSOR DS    CL2           DCB DSORG
CAB1RECF DS    CL1           DCB RECFM
CAB1BLKS DS    H             DCB BLKSIZE
```

```
CAB1LREC DS    H             DCB LRECL
*
         DCBD  DSORG=PS
         YREGS
         END
```


HFS files at a glance

PROBLEM ADDRESSED

Each new release of z/OS seems to feature more components that rely on functions provided by Unix System Services (USS). The heart of the z/OS Unix file system is the HFS (Hierarchical File System), and many z/OS problems can be traced to poor decisions related to HFS datasets. The good news is that the latest level of DFSMS (DFSMS 1.5) vastly improves HFS performance and adds some new controls for managing these important datasets. The bad news is that most of these new controls have not been fully documented and explained. Thus, every now and then 'something' happens to 'some' HFS file belonging to someone from a site's growing USS user base, and a request for help fixing the problem is thus generated. During the course of a recent installation of a new release of an application package, which heavily relies on USS services and HFS files, the question occurred more frequently than usual and this prompted me to search for a quick, simple, and easy-to-use solution that would supply a straightforward HFS file system report and thus help me to solve the problem.

SOLUTION PROPOSED

In a search for a solution I asked myself whether there is any simple and easy-to-use way to get all the relevant information on

HFS files mounted. A quick and simple way to get the HFS file report with a summary of selected information is available from MXI (MVS eXtended Information).

MXI is an MVS tool available free from Scott Enterprise Consultancy Ltd (http://www.secltd.co.uk). It does not use any method of CPU serial number protection or encryption. No passwords or activation zaps are required.

The neat REXX interface that MXI provides was utilized to get the information needed. However, the information it returns was deemed to be only a part of what we needed and therefore it was extended to supply even more information on all mounted HFS files. To accomplish the task three additional REXX procedures were constructed, each being designed to return only specific information on an HFS file.

The report produced consists of three parts – the first part is a view on an HFS file from an ISPF perspective, the second part is a space report from a Unix point of view, while the last part lists the attributes of each HFS mounted. A brief description of each procedure might be helpful to get an idea of what was done, as well as how it might be changed to suit one's demands and needs.

The first EXEC (MVSINFO) looks at a given file from the ISPF space allocation perspective. It uses an ISPF service called DSINFO, which was made available in OS/390 V2R10. The DSINFO service returns information about a particular dataset in dialog variables in the function pool. It is similar to LISTDSI (which doesn't work for HFS file), but it lets you see everything you can see in 3.2 or 3.4 under ISPF. It should be noted that the *number of members* includes the name and subname directories, so this value reflects more than just the number of files in an HFS dataset. On the other hand, the amount of *Current Allocation* and *Used* pages matches the information provided by the Unix **df** command. Additionally, DSINFO returns the unformatted DSCB format1. DSINFO does not require an LMINIT to be performed first. Out of 29 dialog variables saved in the function pool, only a few were actually used – one may choose to pick up and display

any additional variables if they're found to be useful.

The second EXEC (HFSTAT) invokes the statfs callable service to obtain status information about a specified file system from the Unix perspective. On return, the stem used contains the number of variables returned. One can use the predefined variables, beginning with STFS_, to access the status values they represent. Among information returned, those pertaining to space allocation should be carefully examined to avoid out-of-space failures.

In order to update an HFS file, DFSMS uses a technique that keeps duplicate pages in the dataset until the update is complete. This requires there always to be a certain amount of free space within a dataset. This amount actually is based on FS size and activity since the last sync. It is a good idea to monitor the utilized space regularly and take preventive action when one finds that a dataset is close to exhausting its available space.

The last EXEC (HFSATTR) was constructed to get attributes for a mounted HFS file. It invokes the USS stat callable service to obtain the attributes of a specific file.

DISPHFS EXEC

```
/* REXX */
/* Monitoring and reporting information on HFS files             */
 Address TSO
 userid=SYSVAR(SYSUID)
 outds  =userid||'.hfs.out'                /*  change dataset name*/
 x = MSG('OFF')                            /* to fit your standards*/
 if SYSDSN(outds) = 'OK'
 Then "DELETE "outds" PURGE"
 "ALLOC FILE(HFF) DA("outds")",
    " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
    " REUSE LRECL(175) RECFM(F B) BLKSIZE(27825)"
/*-------------------------------------------------------------*/
/* Print headers and labels                                     */
/*-------------------------------------------------------------*/
 mvsout.1 = left('HFS at glance',3Ø,)
 mvsout.2 = '  '
 mvsout.3 = left('PART 1: OS/39Ø HFS UNIX File Information',7Ø)
 mvsout.4 = '  '
 mvsout.5 =left(' ',66,' ') left('------ Space allocation ------',3Ø),
        ||left(' ',13,' ') left('-------- Date ------',21)
 mvsout.6 =left('MVS Data Set',26) left('Path',33),
```

```
       ||left('Volume',7)        left('Members',8),
       ||left('Alloc.',6)        left('Prim.',6),
       ||left('Sec.',4)          left('Ext.',5),
       ||left('Pages',6)         left('%used',6),
       ||left('Created',9)       left('Last ref.',11),
       ||left('Stor. Cl.',10)    left('Mgt. Cl.',11),
       ||left('Data Cl.',11)
 mvsout.7 =left('-',163,'-')
 stat.1 = '  '
 stat.2 =left('PART 2: Display statistics for a mounted HFS',60)
 stat.3 = '  '
 stat.4 = left(' ',60,' ') left('Device',6) left('Parent',25),
        ||left('Block',6)  left('------ Space in blocks ------',31)
 stat.5 =left('File',33)    left('MVS Data Set',27),
       ||left('number',6)  left('device',7),
       ||left('Type',4)    left('Status',7) left('Mode',6),
       ||left('size',6)    left('Total',8)  left('Used',7),
       ||left('Avail.',8)  left('Free',7)
 stat.6 =LEFT('-',130,'-')

 attr.1 = '  '
 attr.2 =left('PART 3: Display mounted HFS attributes',60)
 attr.3 = '  '
 attr.4 =left(' ',32,' ') left('File',5) left('# of',6),
      ||left('General',9) left('Security',9),
      ||left('SetUID',34) left('User',4) left('Group',12),
      ||left('Sticky',7),
      ||left('------------ Date & time stamps ----------',44)
 attr.5 =left('File',32)       left('owner',5) left('links',6),
      ||left('attributes',11) left('used',6)  left('issued',7),
      ||left('RACF file id for auditing',29)  left('id',3),
      ||left('id',3)          left('Auditing',10),
      ||left('bit',4)         left('Created',16),
      ||left('Changed',16)    left('Last access',11)
 attr.6 =left('-',172,'-')
/*-----------------------------------------------------------------*/
/* Issue MXI HFS command to get a list of files                    */
/*-----------------------------------------------------------------*/
 x = mxirexx('list.','NOTITLES','hfs * view(dsn)')
 t = 8
 k = 7
 r = 7
/*-----------------------------------------------------------------*/
/*  Get description of each dataset listed                         */
/*-----------------------------------------------------------------*/
 do i = 3 to list.0
   dss.i=word(list.i,1)
   y = mxirexx('line.','NOTITLES','hfs 'dss.i' view(dsn)')
       DO j = 2 TO LINE.0
          dsn = word(LINE.j,2)  /*                MVS dataset name*/
```

```
          j = j+1
          vol = word(line.j,3)  /*                    Volume Serial*/
          sys = word(line.j,5)  /*                    Owning system*/
          j = j+1
          type = word(line.j,2) /*              Type of file system*/
          mode = word(line.j,4) /*            Write protection mode*/
          j = j+1
          fst  = word(line.j,2) /*     File system type from parmlib*/
          stat = word(line.j,4) /*                           Status*/
          j = j+1
          dev = word(line.j,3)  /*  Device number of the file system*/
          pdev = word(line.j,6) /*              Parent device number*/
          j = j+1
          sec  = word(line.j,2) /*Security to be used to access file*/
          suid = word(line.j,4) /*SetUID can be issued for this file*/
          j = j+1
          path = word(line.j,2) /*                       Path name*/
          j = j+1
          mount = word(line.j,3) /*                 Mount Parameter*/
        SELECT
          when type = 'MVS'     then typ='Local file          '
          when type = 'Remote'  then typ='Remote file         '
          when type = 'Pipe'    then typ='Pipe file           '
          when type = 'Socket'  then typ='Socket file         '
          when type = 'XPFS'    then typ='Cross System PFS    '
          when type = 'CSPS'    then typ='Char special streams'
        END
/*-----------------------------------------------------------------*/
/*  Look at dataset MVS description                                */
/*-----------------------------------------------------------------*/
  call mvsinfo dss.i             /*   MVS HFS Dataset Information */
  mvsout.t= left(dss.i,25)    left(path,33),
            left(vol,6)       right(ZDS#MEM,6),
            right(ZDSTOTA,6)  right(ZDS1EX,6),
            right(ZDS2EX,4)   right(ZDSEXTA,4),
            right(ZDSPAGU,7)  right(ZDSPERU,4),
            left(ZDSCDATE,1Ø) left(ZDSRDATE,1Ø),
            right(ZDSSC,8)    right(ZDSMC,1Ø) right(ZDSDC,1Ø)
  PUSH mvsout.t
  t= t + 1
/*-----------------------------------------------------------------*/
/*  Look at dataset USS description                                */
/*-----------------------------------------------------------------*/
  call hfstat dss.i        /* Display statistics for a mounted HFS */
  stat.k= left(path,33)   left(dss.i,25),
            right(dev,5)  right(pdev,5),
            right(type,7) left(stat,8),
            left(mode,4)  right(blksize,4),
            right(tot,7)  right(inuse,7),
            right(ava,7)  right(free,7)
```

```
  PUSH stat.k
  k= k + 1
/*-----------------------------------------------------------------*/
/*  Look at dataset USS attributes                                 */
/*-----------------------------------------------------------------*/
  call hfsattr path                    /* Display mounted HFS data*/
  attr.r= left(path,33)     right(own,3) right(nlink,6),
          right(genval,9)  right(sec,6) right(suid,7),
          right(aud,3Ø)    right(setuid,2),
          right(setgid,2)  right(aaud,2),
          right(uaud,8)    right(Sticky,2),
          left(rtime,16)   left(ctime,16),
          left(atime,16)
  PUSH attr.r
  r= r + 1
       END
    END
 "EXECIO * DISKW HFF (STEM mvsout.)"
 "EXECIO * DISKW HFF (STEM stat.)"
 "EXECIO * DISKW HFF (STEM attr. FINIS)"
 "free FILE(HFF)"
 Address ISPEXEC
"ISPEXEC BROWSE DATASET('"outds"')"
exit
mvsinfo:
/* rexx                                                          */
/* MVS HFS Dataset Information using DSINFO service              */
 arg dsn
 address ispexec "dsinfo dataset("dsn")"
return
hfstat:
/* rexx :  Get size/space stats for an HFS file                   */
arg fsname
fsname = strip(fsname,,"'")
call syscalls 'ON'
address syscall
"statfs (fsname) st."
  blksize= st.STFS_BLOCKSIZE                        /*Block size*/
  tot    = st.STFS_TOTAL                 /*Total space in blocks*/
  inuse  = st.STFS_INUSE              /*Allocated space in blocks*/
  ava    = st.STFS_AVAIL  /*Space available to unprivileged users*/
  free   = st.STFS_BFREE            /*Total number of free blocks*/
call syscalls 'OFF'
return
hfsattr:
/* rexx : Display attributes for a mounted HFS                    */
ADDRESS SYSCALL
PARSE ARG fname
call syscalls 'ON'
address syscall
```

```
'stat (fname) st.'
call syscalls 'OFF'
  atime=potime(st.ST_ATIME)        /*               Last access*/
  mtime=potime(st.ST_MTIME)        /*             Last modified*/
  ctime=potime(st.ST_CTIME)        /*         File status change*/
  rtime=potime(st.ST_CRTIME)       /*         File creation time*/
  btime=potime(st.ST_RTIME)        /*           File backup time*/
  ccsid = c2x(st.ST_CCSID)         /*     Coded character set ID*/
  genval= c2x(st.ST_GENVALUE)      /*   General attribute values*/
  aud = c2x(LEFT(st.ST_AUDITID,1)) , /*RACF File ID for auditing*/
  SUBSTR(st.ST_AUDITID,2,6)' 'c2x(SUBSTR(st.ST_AUDITID,8))' '
  mdd   =st.ST_MODE                /*                 File mode*/
  nlink =st.ST_NLINK               /*           Number of links*/
  own   =st.ST_UID                 /*          Owner of the file*/
  gid   =st.ST_GID                 /*                  Group ID*/
  size  =st.ST_SIZE                /*                 File size*/
  setuid=st.ST_SETUID              /*  User ID on execution flag*/
  setgid=st.ST_SETGID              /* Group ID on execution flag*/
  aaud  =st.ST_AAUDIT              /*  Auditor audit information*/
  uaud  =st.ST_UAUDIT              /*     User audit information*/
/*blocksa=st.ST_BLOCKS                      Blocks allocated*/
  Sticky =st.ST_STICKY             /*           Sticky bit flag*/
  Extlnk =st.ST_EXTLINK            /*External symbolic link flag*/
/*format= st.ST_FILEFMT                    Format of the file*/
/*ttp   = st.ST_TYPE                           The file type*/
return
potime: procedure expose svalue tm_hour tm_min tm_mon tm_mday tm_year
/* Format posix time values                                  */
  call syscalls 'ON'
arg gt
'gmtime 'gt' gm.'
day = right(gm.tm_mday,2,Ø)              /* derive day   */
min = right(gm.tm_min,2,Ø)               /* derive min   */
mo  = right(gm.tm_mon,2,Ø)               /* derive month */
hr  = right(gm.tm_hour,2,Ø)              /* derive hour  */
value  = day'.'mo'.'gm.tm_year' 'hr':'min
call syscalls 'OFF'
return valueMile Pekic
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

z/OS commands installation exit

The first clue that something was amiss was innocuous enough – a production job had entered device allocation for an offline DASD volume that should have been online. Then another job had an identical problem in allocation for a volume that had been used in a previous step. At that point a multitude of problems occurred, including more of the ones just mentioned. CICS exceeded its number of maximum tasks and experienced a short-on-storage situation, causing it to lock up. IMS experienced device allocation for a volume that was needed for an unopened dataset that it attempted to open. IMS was unable to process its transactions, which began to rapidly pile up in its long and short transaction datasets. Once these datasets were full, IMS abended.

The reason behind all of these problems was that an operator had mistakenly varied 1,500 devices off-line! It was an honest mistake. In order to prevent a recurrence of such a mistake, I developed PCGLCMDX. It is a z/OS commands installation exit. It receives control whenever any command is issued. If the command issued is not a VARY command, it lets it pass without any further checks. If the command *is* a VARY command, it checks whether or not the command is to vary devices OFFLINE. If not, the command is allowed to pass; otherwise the number of devices that are to be varied offline is computed and if that number exceeds 32, the command is disallowed by setting a return code of 2. Thirty-two is an arbitrary number and can easily be changed by any user of this routine.

PCGLCMDX must be link-edited into an authorized library that is in the LNKLST. I used the following attributes for my link-edit: AC=1, AMODE(31), RMODE(ANY), and RENT. PCGLCMDX must be the name specified on the USEREXIT parameter of the .CMD statement in SYS1.PARMLIB(MPFLSTCM), assuming that my names are used. My entry appears as .CMD USEREXIT(PCGLCMDX). In SYS1.PARMLIB(COMMND00), I also included the command **COM='SET MPF=CM'**. I put my

assembled code in SYS1.LINKLIB, refreshed that member, and then implemented it by issuing the **SET MPF=CM** command.

PCGLCMDX has prevented the recurrence of similar problems with device allocations many times since its implementation. Developing it has proved to have been a worthwhile endeavour.

SOURCE

```
         TITLE 'PCGLCMDX - MVS COMMANDS INSTALLATION EXIT'
*********************************************************************
*        PCGLCMDX IS AN MVS COMMANDS EXIT THAT PROCESSES 'VARY'     *
*        COMMANDS ENTERED BY AN OPERATOR WHENEVER HE INTENDS TO     *
*        REMOVE I/O DEVICES FROM USE BY AN OPERATING SYSTEM.  IT    *
*        ENSURES THAT WHENEVER A RANGE OF DEVICES IS ENTERED, IE    *
*        (XXXX-YYYY), WHERE X'S & Y'S ARE DEVICE ADDRESSES, THE RANGE *
*        DOES NOT EXCEED THIRTY-TWO.  IF THE RANGE OF DEVICES WERE  *
*        TO EXCEED THAT AMOUNT, THEN PCGLCMDX SETS A RETURN CODE OF *
*        TWO THEREBY PREVENTING THE OPERATOR FROM ISSUING SUCH A    *
*        COMMAND.                                                   *
*                                                                   *
*        ADDITIONAL REQUIREMENTS:                                   *
*        THE PRESENCE IN SYS1.PARMLIB OF THE FOLLOWING ENTRIES:     *
*            MPFLSTCM - .CMD  USEREXIT(PCGLCMDX)                    *
*            COMMNDØØ - COM='SET MPF=CM'                            *
*                                                                   *
*        REGISTERS AT ENTRY TO PCGLCMDX:                            *
*        1  - POINTER TO CMDX POINTER                               *
*        15 - ADDRESS OF PCGLCMDX                                   *
*        ALL OTHERS - IRRELEVANT TO PCGLCMDX                        *
*                                                                   *
*        REGISTERS AT EXIT FROM PCGLCMDX:                           *
*        15 - RETURN CODE                                           *
*             Ø - PERMIT SYSTEM TO PROCESS COMMAND                  *
*             2 - NO ONE IS AUTHORIZED TO ISSUE SUCH A COMMAND      *
*        ALL OTHERS - IDENTICAL TO VALUES AT ENTRY TO PCGLCMDX      *
*********************************************************************
         SPACE 3
         MACRO
&TAPNAME TAPINFO
         DS    ØF
         PUSH PRINT
         PRINT GEN
         SPACE
&TAPNAME DC    CL8'&SYSECT'
         DC    A(&SYSECT)
         DC    CL6'&SYSTIME'
         DC    CL8'&SYSDATE'
```

```
         SPACE
         POP  PRINT
         MEND
         EJECT
PCGLCMDX CSECT
PCGLCMDX AMODE 31
PCGLCMDX RMODE ANY
         PRINT NOGEN
         SPACE
         USING *,R12               ESTABLISH PCGLCMDX ADDRESSABILITY
         BAKR  R14,Ø               PRESERVE ENVIRONMENT AT ENTRY
         LR    R12,R15             PRIME BASE REGISTER
         L     R1,Ø(R1)            RETRIEVE ADDRESS OF CMDX
         SPACE
         USING CMDX,R1             ESTABLISH CMDX ADDRESSABILITY
         SPACE
         TM    CMDXSRF1,CMDXCCDA+CMDXCIDA TEST IF TERMINATION CALL
         BNZ   PCGLTERM            BRANCH IF SO
         SPACE
*********************************************************************
*        ENSURE THAT THE ENVIRONMENT EXPECTED BY PCGLCMDX IS PRESENT  *
*********************************************************************
         SPACE
         ICM   R2,15,CMDXCLIP      POINT TO COMMAND BUFFER
         BZ    PCGLTERM            EXIT IF UNAVAILABLE
         SPACE
         USING CMDXCLIB,R2         EXTABLISH CMDXCLIB ADDRESSABILITY
         SR    R8,R8               CLEAR A VOLATILE REGISTER
         ICM   R8,3,CMDXCMDL       OBTAIN LENGTH OF COMMAND TEXT
         BZ    PCGLTERM            EXIT IF UNAVAILABLE
         SPACE
         LH    R6,CMDXCMDL         FETCH LENGHT OF COMMAND TEXT
         CLC   CMDXCMDL,PCGLH7     ENSURE THAT COMMAND IS MINIMAL LENG
         BL    PCGLTERM            BRANCH IF NOT
         SPACE
         SH    R6,PCGLH2           REDUCE LENGTH OF TEXT BY TWO
         LA    R5,CMDXCMDI+2       ASSUME SHORT FORM OF VARY COMMAND
         CLC   CMDXCMDI(2),PCGLV   TEST IF SHORT FORM OF VARY COMMAND
         BE    PCGLNONB            BRANCH IF IT IS
         SPACE
         SH    R6,PCGLH3           REDUCE LENGTH OF TEXT BY THREE MORE
         LA    R5,CMDXCMDI+5       ASSUME LONG FORM OF VARY COMMAND
         CLC   CMDXCMDI(2),PCGLVARY TEST IF LONG FORM OF V COMMAND
         BNE   PCGLTERM            BRANCH IF IT IS NOT
         EJECT
*********************************************************************
*        LOCATE FIRST NON-BLANK FOLLOWING THE VARY COMMAND AND        *
*        ATTEMPT TO ENSURE THAT IT IS TO VARY DEVICES OFFLINE.        *
*********************************************************************
         SPACE
```

```
PCGLNONB CLI   Ø(R5),C' '          TEST FOR A BLANK
         BNE   PCGLEND            BRANCH IF NOT
         LA    R5,1(R5)           POINT TO NEXT CHARACTER
         BCT   R6,PCGLNONB        CONTINUE SEARCH FOR NON-BLANK CHAR
         B     PCGLTERM           EXIT IF NONE FOUND
         SPACE
PCGLEND  CLI   Ø(R5),C'('         TEST FOR AN OPEN PARENTHESIS
         BNE   PCGLEMON           BRANCH IF NOT
         CLC   1(L'PCGLOD,R5),PCGLOD TEST FOR O-
         BE    PCGLTERM           EXIT IF IT IS
         B     PCGLGO             ONWARD!
         SPACE
PCGLEMON CLC   Ø(L'PCGLPATH,R5),PCGLPATH TEST FOR PATH
         BE    PCGLTERM           EXIT IF IT IS
         SPACE
         CLC   Ø(L'PCGLXCF,R5),PCGLXCF TEST FOR XCF
         BE    PCGLTERM           EXIT IF IT IS
         SPACE
         CLC   Ø(L'PCGLSMS,R5),PCGLSMS TEST FOR SMS
         BE    PCGLTERM           EXIT IF IT IS
         SPACE
         CLC   Ø(L'PCGLCN,R5),PCGLCN TEST FOR CN
         BE    PCGLTERM           EXIT IF IT IS
         EJECT
*********************************************************************
*        ATTEMPT TO LOCATE THE CHARACTERS ',OFF' EMBEDDED WITHIN THIS *
*        COMMAND.  IF THEY ARE NOT ENCOUNTERED, THEN THIS CANNOT      *
*        BE A COMMAND ISSUED TO REMOVE DEVICES FROM THE SYSTEM.       *
*********************************************************************
         SPACE
PCGLGO   LH    R14,CMDXCMDL       FETCH LENGHT OF COMMAND TEST
         LA    R15,CMDXCMDI(R14)  POINT TO END OF COMMAND TEXT
         SPACE
PCGLCSCN CLI   Ø(R15),C','         SEARCH FOR A COMMA
         BE    PCGLCOMA           BRANCH WHEN ONE IS DISCOVERED
         SPACE
         CLI   Ø(R15),C')'         SEARCH FOR AN OPEN PARENTHESIS
         BE    PCGLTERM           DEPART WHEN ONE IS DISCOVERED
         SPACE
PCGLOOP  BCTR  R15,RØ              POINT TO PREVIOUS CHARACTER
         BCT   R14,PCGLCSCN        LOOP POWER!
         B     PCGLTERM           EXIT IF NONE
         EJECT
*********************************************************************
*        EXTRACT BEGINNING OF I/O DEVICE RANGE                        *
*********************************************************************
         SPACE
PCGLCOMA CLC   Ø(L'PCGLOFF,R15),PCGLOFF TEST IF UNITS ARE TO BE REMOVED
         BNE   PCGLOOP            BRANCH IF NOT
         SPACE
```

```
         SR    R1Ø,R1Ø             CLEAR A BEVY OF VOLATILE
         LR    R11,R1Ø              GENERAL
         LR    R7,R11                PURPOSE
         LR    R9,R7                  REGISTERS
         SPACE
         CLI   Ø(R5),C'('          TEST FOR AN OPEN PARENTHESIS
         BNE   PCGLEAN             BRANCH IF NOT
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER
         BCT   R6,PCGLEAN          REDUCE COUNT BY ONE
         B     PCGLTERM            BRANCH IF AT END OF TEXT
         SPACE
PCGLEAN  CLI   Ø(R5),C'/'          TEST FOR A SLASH
         BNE   PCGLFROM            BRANCH IF NOT
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER
         BCT   R6,PCGLFROM         REDUCE COUNT BY ONE
         B     PCGLTERM            BRANCH IF AT END OF TEXT
         SPACE
PCGLFROM ICM   R7,1,Ø(R5)          RETRIEVE A SINGLE CHARACTER
         LA    R15,PCGTRTBL(R7)    POINT TO ITS HEXADECIMAL VALUE
         CLI   Ø(R15),X'FF'        TEST IF INVALID DEVICE CHARACTER
         BE    PCGLTERM            EXIT IF SO
         SPACE
         ICM   R11,8,Ø(R15)        TRANSLATE EBCDIC TO HEXADECIMAL
         SLL   R11,4               REMOVE HIGH-ORDER ZEROES
         SLDL  R1Ø,4               ALIGN DEVICE ADDRESS IN REGISTER
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER IN CMD TEXT
         BCT   R6,PCGLGADR         REDUCE LENGTH REMAINING BY ONE
         B     PCGLTERM            QUIT WHEN DONE
         EJECT
*********************************************************************
*        ASCERTAIN POINT OF PROCESSING AND PROCEED ACCORDINGLY      *
*        IF AT END OF COMMAND, TERMINATE;                           *
*        IF AT END SET, PROCESS THE NEXT ONE;                       *
*        IF AT RANGE DELIMITER, GO PROCESS ITS UPPER LIMIT          *
*********************************************************************
         SPACE
PCGLGADR CLI   Ø(R5),C')'          TEST IF AT END OF COMMAND
         BE    PCGLTERM            BRANCH IF SO; PROCESSING IS COMPLETE
         SPACE
         CLC   Ø(L'PCGLOFF,R15),PCGLOFF TEST IF AT END OF COMMAND
         BE    PCGLTERM            BRANCH IF SO; PROCESSING IS COMPLETE
         SPACE
         CLI   Ø(R5),C'-'          TEST IF AT A RANGE-DELIMITER
         BE    PCGLTO              BRANCH IF SO TO PROCESS UPPER LIMIT
         SPACE
         CLI   Ø(R5),C','          TEST IF AT A SET-DELIMITER
         BNE   PCGLFROM            BRANCH IF NOT TO PROCESS CHARACTER
```

```
         SPACE
PCGLNSET LA    R5,1(R5)            POINT TO NEXT CHARACTER IN CMD TEXT
         SR    R1Ø,R1Ø             REMOVE DE DETRITUS
         BCT   R6,PCGLEAN          PROCESS NEXT CHARACTER
         B     PCGLTERM            EXIT - STAGE RIGHT
         EJECT
*********************************************************************
*        PROCESS THE ULTIMATE END OF THE RANGE OF DEVICES THAT      *
*        ARE TO BE VARIED OFFLINE.                                  *
*********************************************************************
         SPACE
PCGLTO   LR    RØ,R1Ø              STOW START OF RANGE OF ADDRESSES
         LA    R5,1(R5)            POINT TO NEXT CHARACTER IN CMD TEXT
         SR    R1Ø,R1Ø             REMOVE DE DETRITUS
         BCT   R6,PCGNEAN          PROCESS END OF RANGE OF DEVICES
         B     PCGLTERM            EXIT IF AT END OF COMMAND
         SPACE
PCGNEAN  CLI   Ø(R5),C'/'          TEST FOR A SLASH
         BNE   PCGNTO              BRANCH IF NOT
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER
         BCT   R6,PCGNTO           REDUCE COUNT BY ONE
         B     PCGLTERM            EXIT IF AT END OF COMMAND
         SPACE
PCGNTO   ICM   R7,1,Ø(R5)          RETRIEVE A SINGLE CHARACTER
         LA    R15,PCGTRTBL(R7)    POINT TO ITS HEXADECIMAL VALUE
         CLI   Ø(R15),X'FF'        TEST IF INVALID DEVICE CHARACTER
         BE    PCGLTERM            EXIT IF SO
         SPACE
         ICM   R11,8,Ø(R15)        TRANSLATE EBCDIC TO HEXADECIMAL
         SLL   R11,4               REMOVE HIGH-ORDER ZEROES
         SLDL  R1Ø,4               ALIGN DEVICE ADDRESS IN REGISTER
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER IN CMD TEXT
         BCT   R6,PCNLGADR         REDUCE LENGTH REMAINING BY ONE
         B     PCGLTERM            EXIT WHEN OUT OF TEXT
         SPACE
PCNLGADR CLI   Ø(R5),C')'          TEST IF AT END OF COMMAND
         BE    PCGNEND             BRANCH IF SO
         SPACE
         CLC   Ø(L'PCGLOFF,R15),PCGLOFF TEST IF AT END OF COMMAND
         BE    PCGNEND             BRANCH IF SO; PROCESSING IS COMPLETE
         SPACE
         CLI   Ø(R5),C','          TEST IF AT A SET-DELIMITER
         BNE   PCGNTO              BRANCH IF NOT TO PROCESS CHARACTER
         LA    R9,1                SET SWITCH FOR CONTINUATION
         EJECT
*********************************************************************
*        CHECK HERE WHETHER THE NUMBER OF DEVICES TO BE TAKEN OFFLINE *
*        EXCEEDS THIRTY-TWO.  IF SO, THEN TERMINATE PROCESSING      *
```

```
*        WITH A RETURN CODE OF TWO, INVALID AUTHORITY, ELSE SEARCH    *
*        FOR AND PROCESS ADDITIONAL DEVICE RANGES THAT ARE TO BE      *
*        VARIED OFFLINE.                                              *
*********************************************************************
         SPACE
PCGNEND  SR    R1Ø,RØ              DETERMINE # OF DEVICES TO BE REMOVED
         BM    PCGLTERM            EXIT IF NEGATIVE RANGE
         C     R1Ø,PCGLF32         TERM IF WITHIN ALLOWABLE RANGE
         BL    PCGNNSET            CONTINUE IF SO
         LA    R15,2               SHOW OPERATOR COMMAND NOT AUTHORIZED
         PR    R14                 BACK TO DUST
         SPACE
PCGNNSET LTR   R9,R9               TEST IF END OF PROCESSING
         BE    PCGLTERM            BRANCH IF SO
         SR    R9,R9               CLEAR END-OF-PROCESSING SWITCH
         SPACE
         LA    R5,1(R5)            POINT TO NEXT CHARACTER IN CMD TEXT
         SR    R1Ø,R1Ø             REMOVE DE
         LR    RØ,R1Ø                DETRITUS
         BCT   R6,PCGLEAN          PROCESS NEXT SET OF DEVICES
         SPACE
PCGLTERM DS    ØH
         SR    R15,R15             SET ZERO RETURN CODE
         PR    R14                 BACK TO DUST
         EJECT
*********************************************************************
*        CONSTANTS AND OTHER SUCH NONSENSE                            *
*********************************************************************
         SPACE
PCGLF32  DC    F'33'
PCGLH2   DC    H'2'
PCGLH3   DC    H'3'
PCGLH7   DC    H'7'
         SPACE
PCGLV    DC    CL2'V '
PCGLVARY DC    CL5'VARY '
         SPACE
PCGLCN   DC    CL2'CN'
PCGLOD   DC    CL2'O-'
PCGLSMS  DC    CL3'SMS'
PCGLXCF  DC    CL3'XCF'
PCGLOFF  DC    CL4',OFF'
PCGLPATH DC    CL4'PATH'
         SPACE
PCGLTRAN DC    C'Ø123456789ABCDEF'
         SPACE
PCGTRTBL DC    256X'FF'
         SPACE
         ORG   PCGTRTBL+C'A'
         DC    X'ØAØBØCØDØEØF'
```

```
SPACE
ORG   PCGTRTBL+C'Ø'
DC    X'ØØØ1Ø2Ø3Ø4Ø5Ø6Ø7Ø8Ø9'
SPACE
ORG
SPACE
TAPINFO
SPACE
YREGS
SPACE
IEZVX1Ø1
SPACE
END
```


System-wide member search utility – part 2

This month we conclude the code for a PDS member search facility.

REXX1

```
/* REXX */
/*--------------------------------------------------------------------*
* REXX NAME :                                                         *
* Rexx1 (called from the job JCL1.)                                   *
*                                                                     *
* FUNCTION :                                                          *
* This REXX reads the Dcollect output and extracts all PO datasets    *
* within the Dcollect scope. Then members of each dataset are fetched *
* and put into the sequential Repository dataset with their related   *
* fields such as dataset name, volser etc.                            *
*                                                                     *
* DATASETS USED :                                                     *
* 1 - File name = DCOLIN                                              *
* Exp.Memsrch.Dcollect     (In) -> Dcollect dataset.                  *
*                                                                     *
* 2 - File name = MEMDB                                               *
* Exp.Memsrch.Seq         (Out) -> Member inventory temp dset (SEQL). *
*--------------------------------------------------------------------*/
"Prof Nopref"
Status = Msg('Off')
Trace Off
```



```
ADDRESS "TSO"
/*-------------------------------------------------------------------*/
/* Set up constants.                                                 */
/*-------------------------------------------------------------------*/
Eof = 'no'
Dsorg_po = c2x('Ø2ØØ'x)
k = Ø  /* member count */
m = Ø  /* Counter      */
n = Ø  /* Dset  count  */
Unique_volsers = ""
/*-------------------------------------------------------------------*/
/* Allocate the Inventory dataset (SEQL).                            */
/*-------------------------------------------------------------------*/
"Alloc Fi(Memdb) Da('Exp.Memsrch.Seq') Shr Reuse"
/*-------------------------------------------------------------------*/
/* Begin processing the DCOLLECT dataset.                            */
/*-------------------------------------------------------------------*/
Drop rec.
"Execio 1 Diskr Dcolin (Stem rec.)"
IF rc <> Ø THEN
Eof = 'yes'
DO WHILE(Eof = 'no')                        /* Loop  ANKARA   */
   Dcolrec = rec.1
   /* Check to see if this a D record.      */
  IF ((SUBSTR(dcolrec,5,1) = 'D') & (SUBSTR(dcolrec,5,2) <> 'DC')) THEN
   DO                                      /* SEVGI */
      /*---------------------------------------------------------------*/
      /* Parse D record into variables.                                */
      /*---------------------------------------------------------------*/
      PARSE VAR dcolrec 25 dset 69 . 79 volser 85 .
      Dsorg = c2x(substr(dcolrec,75,2))
      Dsn = Strip(Dset)
      If (Dsorg = dsorg_po) THEN Out_dsorg = 'PO'
                            ELSE Out_dsorg = 'XX'
   /*---------------------------------------------------------------*/
   /* To reduce the scope of the utility, you can put here a filter.  */
   /* For example, in order the Inventory to have just members of     */
   /* "EXP.**" datasets, you can code the following statement here:  */
   /* If (Out_dsorg='PO') & (Substr(Dsn,1,4)='EXP.') Then             */
   /*---------------------------------------------------------------*/
   If Out_dsorg='PO' Then
   Do
     m = m + 1
     /*-------------------------------------------------------------*/
     /* Fetch all members of current PO dataset by the Getmem proc.  */
     /*-------------------------------------------------------------*/
     Call Getmem
   End
  END  /* SEVGI */
 "Execio 1 Diskr Dcolin (Stem Rec.)"
```

```
 IF rc <> Ø THEN Eof = 'yes'
END                        /* End Loop ANKARA */
/*-------------------------------------------------------------------*/
/* At this point, we have read all Dcollect records.                 */
/*-------------------------------------------------------------------*/
"Execio Ø Diskr Dcolin (Finis"
"Free Fi(Dcolin)"
/*-------------------------------------------------------------------*/
/* Write a specific record which indicates the number of members and  */
/* the number of unique PDSs and number of disk volsers from which    */
/* PDSs are taken into account in building the Inventory. This        */
/* record will be presented to the user on the Query Input / Output   */
/* panels (PanelIN / PanelOUT).                                       */
/*-------------------------------------------------------------------*/
k = k + 1
n = k - 1
o = Length(Unique_volsers)/6  /* Num. of unique volsers in the INV.   */
Compound.k= "       $" || " " || "DSET NUM = " || m || ,
            " - MEMBER NUM = " || n || " - BUILD DATE = "Date('E'),
            " - VOL.NUM= " || o
/*-------------------------------------------------------------------*/
/* Write all volume information from which DCOLLECT extracted PO data */
/* sets in the system. These information will be kept in the Inventory*/
/* dataset for convenience and they will be presented to the user     */
/* on the Query Output panel (PanelOUT) on demand basis.              */
/*-------------------------------------------------------------------*/
k = k + 1
Do i = 1 to length(Unique_Volsers)/72
  Compound.k     = "      $ " || " " || Substr(Unique_Volsers,1,72)
  Unique_Volsers = Substr(Unique_Volsers,73)
  k = k + 1
End
Compound.k       = "      $ " || " " || Unique_Volsers
Compound.Ø= k-1   /* Put the number of entries in the compound var.*/
/*-------------------------------------------------------------------*/
/* Build Sequential Inventory dataset.                               */
/*-------------------------------------------------------------------*/
"Execio * Diskw Memdb (Stem Compound. Finis"
"Free File(Memdb)"
/*-------------------------------------------------------------------*/
/* Sort Sequential Inventory dataset.                                */
/*-------------------------------------------------------------------*/
Call Sortfile
EXIT Ø    /* End of MAIN REXX */
/*===================================================================*/
/*===================================================================*/
GETMEM:
/*-------------------------------------------------------------------*/
/* This procedure fetches all members of a given partitioned dataset. */
/*-------------------------------------------------------------------*/
```

```
Address "ISPEXEC"
/*-------------------------------------------------------------------*/
/* Set the error processing mode to allow the dialog function to     */
/* process Return codes of 12 or higher.                             */
/*-------------------------------------------------------------------*/
"Control Errors Return"
Member = ' '
Lmrc = Ø
"Lminit Dataid(PIR) Dataset('"Dsn"')"
If (Rc <> Ø & Index(ZERRSM,'Dataset not cataloged'))
     Then
       /*-------------------------------------------------------------*/
       /* If it's not catalogued, then we issue Lminit command again  */
       /* with VOLUME parameter, making the utility more versatile.   */
       /*-------------------------------------------------------------*/
       Do
           "Lminit Dataid(PIR) Dataset('"Dsn"') Volume("volser")"
           Lmrc = RC
       End
If Lmrc = Ø Then Do; "Lmopen Dataid("PIR") Option(Input)";Lmrc=Rc;End
            Else Say "LMINIT error for " Dsn "Error msg = "ZERRSM LMRC
If Lmrc <> Ø Then Say "LMOPEN error for" Dsn "Error msg = "ZERRSM LMRC
/*-------------------------------------------------------------------*/
/* Loop through all members in the current PO dataset.               */
/*-------------------------------------------------------------------*/
Do While Lmrc = Ø /* MOP */
  "Lmmlist Dataid("PIR") Option(List)  Member(member) Stats(Yes)"
  Lmrc = Rc
  If Lmrc = Ø Then
  DO    /* POP */
     DT = ZLMDATE       /* Last change date.                         */
     DC = ZLCDATE       /* Creation date.                            */
     DU = ZLUSER        /* User-id of last user to change the member. */
     If DU =  "" Then DU = "N/A"
     If DT <> "" Then DT2 = DT
                 Else DT2 = "N/A"
     If DC <> "" Then DC2 = DC
                 Else DC2 = "N/A"
     k = k + 1 /* Record count in Inventory. */
     Compound.k= Member || " " || Dset || " " || Volser,
              || " " || DC2 ||" " || DT2 || " " || DU
     Call Unique
  END   /* POP */
END               /* MOP */
"Lmclose Dataid("PIR")"
"Lmfree  Dataid("PIR")"
ADDRESS "TSO"
RETURN /* End_of_the_procedure GETMEM   */
/*===================================================================*/
/*===================================================================*/
```

```
SORTFILE:
/*-------------------------------------------------------------------*/
/* Sort Inventory dataset (SEQL) in the ascending order. Sort field  */
/* will be entire record, which contains  the following fields:      */
/* (Member+Dset+Volser+Creation date+Modification date+Modified user) */
/*-------------------------------------------------------------------*/
"Alloc Fi(Sortin)   Da('Exp.Memsrch.Seq') Shr Reuse"
"Alloc Fi(Sortout)  Da('Exp.Memsrch.Seq') Shr Reuse"
"Alloc Fi(Sortlib)  Da('Sys1.Sortlib')    Shr Reuse"
"Alloc Fi(sortwkØ1) Cylinders Space(5Ø,5) Unit(Sysda) Reuse"
"Alloc Fi(sortwkØ2) Cylinders Space(5Ø,5) Unit(Sysda) Reuse"
"Alloc Fi(Sysout) Space(9,3) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Alloc Fi(Sysin)  Space(1,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Newstack"         /*  Create a new data stack for Sort Sysin file.   */
Sort_sysin = ' SORT     FIELDS=(1,86,CH,A)'
Queue Sort_sysin
/*-------------------------------------------------------------------*/
/*  Add a null line to indicate the end of the information.          */
/*-------------------------------------------------------------------*/
 Queue ""
"Execio 1 Diskw Sysin (Finis"
"Delstack"                                  /* Delete the data stack. */
"Call 'Sys1.Sortlpa(Sort)'"                 /* Call the Sort program. */
If Rc <> Ø Then  Say 'Sort is NOT successfull. Return Code= ' RC
"Free File(Sortin,Sortout,Sysin,SortwkØ1,Sortlib)"
RETURN /* End_of_the_Procedure SORTFILE   */
/*===================================================================*/
/*===================================================================*/
UNIQUE:
/*-------------------------------------------------------------------*/
/* This procedure is used to merge all unique disk volumes within the */
/* Dcollect scope.                                                   */
/*-------------------------------------------------------------------*/
If Index(Unique_volsers,Volser) = Ø Then
                              Unique_volsers = Unique_volsers || Volser
RETURN /* End_of_the_Procedure UNIQUE   */
/*===================================================================*/
```

SIL_REXX2

```
/* REXX */
/*-------------------------------------------------------------------*
* REXX NAME :                                                        *
* Rexx2     ( executed in the foreground.)                           *
* FUNCTION :                                                         *
* This REXX is the interactive part of the "Member Search Utility."  *
* Once the "Member Inventory VSAM " is built by the Job JCL1, by     *
* executing this REXX, a user can do member search and take actions  *
* such as (B)rowse / (E)dit / (D)elete / (P)rint on any member       *
```

```
* against a user query.                                              *
* DATASETS USED :                                                    *
* Exp.Memsrch.Vsam     (In) -> Member Inventory dataset        (VSAM) *
* Exp.Memsrch.Output  (Out) -> Query  Result    dataset        (SEQL) *
* Exp.Memsrch.Journal (Out) -> Journal dataset to keep track         *
*                              of member deletions.             (SEQL) *
*--------------------------------------------------------------------*/
"Prof Nopref"
Status = Msg('Off')
Trace Off
DsetVS  = Exp.Memsrch.Vsam
DsetOU  = Exp.Memsrch.Output
Logdset = Exp.Memsrch.Journal
/*--------------------------------------------------------------------*/
/* Set the error processing mode to allow the dialog function to      */
/* process Return codes of 12 or higher.                              */
/*--------------------------------------------------------------------*/
"Ispexec Control Errors Return"
Call AllocISPF                 /* Allocate necessary ISPF libraries.  */
Envr = Mvsvar('SYMDEF','SYSNAME') /* Get Lpar name and put it in pool.*/
"Ispexec Vput Envr Profile"
/*--------------------------------------------------------------------*/
/* Fetch the num. of members, num. of datasets and the creation date */
/* of Inventory which are all recorded in the 1st record of the       */
/* Inventory. Note that this information will be presented to the user*/
/* on both Query Input and Query Output panels (PanelIN & PanelOUT).  */
/*--------------------------------------------------------------------*/
Flag    = 1
Type    = 'S'                        /* It will be (S)pecific search. */
Membrec = "       $"   /* This special record starts with '       $'. */
Call Read_By_Repro
Membrec = " "                   /* Initialize the Membrec variable.   */
Type    = " "                   /* Initialize the Type    variable.   */
Numdset = Word(Record.1,5)
Nummemb = Word(Record.1,10)
Blddate = Word(Record.1,15)
Numvol  = Word(Record.1,18)
"Ispexec Vput (Numdset,Nummemb,Blddate,Numvol) Profile"
LAB1:
/*--------------------------------------------------------------------*/
/* Display the Query Input Panel where user enters the member name he */
/* wants to search throughout the Inventory and which kind of search  */
/* he'll realize, Generic / Specific / Substring or Mask. These two   */
/* input variables are put into Membrec & Type variables respectively.*/
/*--------------------------------------------------------------------*/
"Ispexec Display Panel(PanelIN)"
/*--------------------------------------------------------------------*/
/* If user presses the PF03 or PF04 key, return to the main menu.     */
/*--------------------------------------------------------------------*/
If (Spfkey='PF03' | Spfkey='PF04') Then Return
```

```
/*-------------------------------------------------------------------*/
/* Get the member name and search type from "Query Input" panel.      */
/*-------------------------------------------------------------------*/
"Ispexec Vget (Membrec,Mask,Type) Profile"
Flag = Ø
/*-------------------------------------------------------------------*/
/* If Search Type is 'S' or 'G', member search is done by using the  */
/* IDCAMS program. However, for other types of search, the DFSORT    */
/* program will be called for fetching related records against user  */
/* query.                                                            */
/*-------------------------------------------------------------------*/
If Index('SG',Type) <> Ø Then Call Read_By_Repro   /* Read by IDCAMS. */
                         Else Call Read_By_Dfsort  /* Read by DFSORT. */
Call Deallocispf                        /* De-allocate ISPF libraries. */
EXIT Ø
/*===================================================================*/
/*===================================================================*/
READ_BY_REPRO:
/*-------------------------------------------------------------------*/
/* The VSAM Inventory dataset is read to get all occurrences of      */
/* specific or generic user-specified member patterns.               */
/*                                                                   */
/* DATASETS USED BY REPRO IN THIS PROCEDURE:                         */
/* File name Variable   Dataset name           Description           */
/* ========= =========  ====================   ============          */
/* In        DsetVS     Exp.Memsrch.Vsam        Repro input          */
/* Out       DsetOU     Exp.Memsrch.Output      Repro output         */
/*-------------------------------------------------------------------*/
"Alloc Fi(In) Da("DsetVS") Shr Reuse"
If RC <> Ø Then
             Do
               /*-------------------------------------------------*/
               /* Warn the user that VSAM Inventory is not defined.  */
               /*-------------------------------------------------*/
               Call Showmsg 8          /* "Inventory not exist" msg. */
               Exit
             End
/*-------------------------------------------------------------------*/
/* Delete and define "Repro output" dataset.                        */
/*-------------------------------------------------------------------*/
IF Sysdsn(DsetOU) = 'OK' Then Delete DsetOU
"Alloc Fi(Out) Blksize(Ø) Reuse Lrecl(86) New Catalog Recfm(F B),
Da("DsetOU") Dsorg(PS) Tracks Space(5 1)"
/*-------------------------------------------------------------------*/
/* Build the proper Repro statements depending on the member search  */
/* type; which can be '(S)pecific' or '(G)eneric'.                   */
/*-------------------------------------------------------------------*/
If Type = S Then
          Do
           Line1 = "Infile(In) Outfile(Out) Fromkey('"Membrec" ') "
```

```
            Line2 = "Tokey('"Membrec" ')"
           End
          ELSE
           Do
            Line1 = "Infile(In) Outfile(Out) Fromkey('"Membrec"') "
            Line2 = "Tokey('"Membrec"')"
           End
"Repro " Line1 Line2
RC_Repro = Rc
"Free Fi(In Out)"
/*-------------------------------------------------------------------*/
/*  Read Repro Output file.                                          */
/*-------------------------------------------------------------------*/
If Rc_Repro = Ø Then Do
                       "Alloc Fi(XX) Da("DsetOU") Shr Reuse"
                       "Execio * Diskr "XX" (Stem Record. Finis"
                       "Free Fi(XX)"
                       If Flag = Ø /* Present query results to user. */
                                Then  Call Present_Results
                    End
                /*-----------------------------------------------------*/
                /* If Repro ends with a non-zero Rc, no-records found.*/
                /*-----------------------------------------------------*/
                Else Do
                       Call Showmsg 7  /* "No record found" message. */
                       Signal LAB1  /* Ask user to do another query. */
                    End
RETURN    /* End_of_the-procedure READ_BY_REPRO */
/*===================================================================*/
/*===================================================================*/
READ_BY_DFSORT:
/*-------------------------------------------------------------------*/
/* DFSORT reads the Inventory dataset (VSAM) to fetch all members    */
/* which meet the character string entered by user for Sub-string and*/
/* Masking type of member-search.                                    */
/*                                                                   */
/* DATASETS USED BY DFSORT IN THIS PROCEDURE:                        */
/* File name Variable   Dataset name                                 */
/* ========= =========  ====================                         */
/* Sortin    DsetVS     Exp.Memsrch.Vsam                           */
/* Sortout   DsetOU     Exp.Memsrch.Output                         */
/*-------------------------------------------------------------------*/
"Alloc Fi(SortIN) Da("DsetVS") Shr Reuse"
If RC <> Ø Then
              Do
                /*-----------------------------------------------------*/
                /* Warn the user that Vsam Inventory is not defined.  */
                /*-----------------------------------------------------*/
                Call Showmsg 8          /* "Inventory not exist" msg. */
                Exit
```

```
            End
/*-------------------------------------------------------------------*/
/* Delete and define DFSORT Sortout dataset.                         */
/*-------------------------------------------------------------------*/
IF Sysdsn(DsetOU) = 'OK' Then Delete DsetOU
"Alloc Fi(SortOUT) Blksize(Ø) Reuse Lrecl(86) New Catalog Recfm(F B),
Da("DsetOU") Dsorg(PS) Tracks Space(5 1)"
"Alloc Fi(Sysout) Space(2,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Alloc Fi(Sysin)  Space(1,1) Track Lrecl(8Ø) Recfm(f) Reuse"
/*-------------------------------------------------------------------*/
/* Build the proper DFSORT statements depending on the member search  */
/* type; which can be 'Sub-string' or "Mask'.                        */
/*-------------------------------------------------------------------*/
If Type = T Then   /* Member-search type : Sub-string   */
   DO      /* 999 */
     Copy_sysin1= " SORT FIELDS=(1,86,CH,A)"
     Copy_sysin2= " INCLUDE FORMAT=SS,COND=(1,8,EQ,C'"Membrec"')"
     "Newstack"
     Queue Copy_sysin1
     Queue Copy_sysin2
     Queue ""
     "Execio 3 Diskw Sysin (Finis"
     "Delstack"
   END     /* 999 */
If Type = M Then   /* Member-search type : Mask        */
   DO      /* 888 */
/*-------------------------------------------------------------------*/
/* Build the Array. to hold non-% characters in the Mask variable.   */
/*-------------------------------------------------------------------*/
"Newstack"
j = Ø
Do i = 1 to 8
  If Substr(Mask,i,1) <> '%' Then
          Do
           j = j + 1
           Array.j = i
          End
End
Array.Ø = j
Queue " SORT FIELDS=(1,86,CH,A)"
p = Array.1
s = Substr(Mask,p,1)
If Array.Ø = 1 Then Queue " INCLUDE FORMAT=SS,COND=("p",1,EQ,C'"s"')"
               Else Queue " INCLUDE
FORMAT=SS,COND=("p",1,EQ,C'"s"',AND,"
Do i = 2 to Array.Ø
  p = Array.i
  s = Substr(Mask,p,1)
  If i <> Array.Ø Then Queue "         "p",1,EQ,C'"s"',AND,"
                  Else Queue "         "p",1,EQ,C'"s"')"
```

```
 End
Queue ""
"Execio * Diskw Sysin (Finis"
"Delstack"
  END      /* 888 */
/*--------------------------------------------------------------------*/
/*  Call DFSORT program.                                              */
/*--------------------------------------------------------------------*/
"Call 'Sys1.Sortlpa(Sort)'"
Rc_Sort = Rc
/*--------------------------------------------------------------------*/
/*  Free allocated datasets used by DFSORT.                           */
/*--------------------------------------------------------------------*/
"Free Fi(Sortin,Sortout,Sysout)"
/*--------------------------------------------------------------------*/
/*  Read the Sortout file.                                            */
/*--------------------------------------------------------------------*/
If Rc_Sort = Ø Then Do
                       "Alloc Fi(XX) Da("DsetOU") Shr Reuse"
                       "Execio * Diskr "XX" (Stem Record. Finis"
                       "Free Fi(XX)"
                       /*-----------------------------------------*/
                       /* If Record.Ø is Ø, no records found.     */
                       /*-----------------------------------------*/
                       If  Record.Ø = Ø Then
                        Do
                         Call Showmsg 25 /*"No record found" message.*/
                         Signal LAB1 /* Ask user to do another query.*/
                        End
                        /*-----------------------------------------*/
                        /* Present query results to user.         */
                        /*-----------------------------------------*/
                        Call Present_Results
                    End
               Else Do
                       Call Showmsg 24 /* "DFSORT error"  message.   */
                       Signal LAB1  /* Ask user to do another query. */
                    End
RETURN    /* End_of_the-procedure READ_BY_DFSORT */
/*====================================================================*/
/*====================================================================*/
ALLOCISPF:
/*--------------------------------------------------------------------*/
/* Allocate utility's ISPF panel and Rexx libraries dynamically.      */
/*--------------------------------------------------------------------*/
"Ispexec Libdef Ispplib"
"Altlib Deactivate Application(Exec)"
"Ispexec Libdef Ispplib Dataset    Id(Exp.Memsrch.Cntl)"
"Altlib Activate Application(Exec) Da(Exp.Memsrch.Cntl)"
RETURN    /* End_of_the_procedure ALLOCISPF   */
```

```
/*====================================================================*/
/*====================================================================*/
DEALLOCISPF:
/*--------------------------------------------------------------------*/
/* De-allocate utility's ISPF panel and Rexx libraries.               */
/*--------------------------------------------------------------------*/
"Ispexec Libdef Ispplib"
"Altlib Deactivate Application(Exec)"
RETURN    /* End_of_the_procedure DEALLOCISPF     */
/*====================================================================*/
/*====================================================================*/
PRESENT_RESULTS:
/*--------------------------------------------------------------------*/
/*  Create table which will be used to show query results fetched     */
/*  from the VSAM inventory dataset. The Ispf table will consist of   */
/*  the following fields:                                             */
/*  K1: Member name   K2: Dataset name   K3: Volume serial number     */
/*  K4: Member creation date             K5: Member last-update date  */
/*  K6: User who updated the member last                              */
/*--------------------------------------------------------------------*/
Ran = Random(1,1ØØ)                    /* Build table_name randomly. */
Table_name = 'MEMTA' || Ran
"Ispexec Tbcreate" Table_name  "Names(K1,K2,K3,K4,K5,K6)"
Do p = 1 to Record.Ø
  Pepe = Record.p
  Parse Var Pepe K1 K2 K3 K4 K5 K6
  "Ispexec Tbadd" Table_name           /* Add a record to the table. */
End
qc = p - 1          /* Number of Members found against user's query. */
/*--------------------------------------------------------------------*/
/* Present query result table to the user.                           */
/*--------------------------------------------------------------------*/
LAB2:
"Ispexec Tbtop" Table_name             /* Move to top of table.      */
/*--------------------------------------------------------------------*/
/* Initialize the message, cursor, and cursor row parameters.        */
/*--------------------------------------------------------------------*/
Message = 'Msg( )'
Cursor  = 'Cursor( )'
Csrrow  = 'Csrrow(1)'
Address Ispexec
/*--------------------------------------------------------------------*/
/* Loop forever until the PF3 or PF4 key is pressed on the "Query     */
/* Output Panel", which is PanelOUT.                                 */
/*--------------------------------------------------------------------*/
Do Forever  /* Do-forever DENIS */
    "Tbdispl" Table_name  "Panel(PanelOUT)" Message Cursor Csrrow,
             "Autosel(no) Position(crp)"
     /*-------------------------------------------------------------*/
     /* Get the variables from PanelOUT.                           */
```

```
     /*-------------------------------------------------------------*/
     "Vget (L,Pcmd,Spfkey) Profile"
     /*-------------------------------------------------------------*/
     /* If user DOESN'T CHOOSE any member, then process other actions */
     /* realised by user on PanelOUT.                               */
     /*-------------------------------------------------------------*/
     If Index('BEDPGC',L) = Ø Then
      Do    /* BENIM */
       /*-----------------------------------------------------------*/
       /* If user presses PF3 or PF4 key, close the table and goto  */
       /* Label LAB1 to display again PanelIN panel so that user    */
       /* can make another query.                                   */
       /*-----------------------------------------------------------*/
       If (Spfkey = PFØ3 | Spfkey = PFØ4 | Pcmd = Cancel ) Then
        Do
          "Tbend" Table_name
          Address Tso
          Signal LAB1
        End
/*-------------------------------------------------------------------*/
/* Process commands entered by user on Query Output Panel (PanelOUT). */
/*-------------------------------------------------------------------*/
If Pcmd = 'LSTNG' Then Call Get_lstng /* Get listing of query result. */
If Pcmd = 'AVOLS' Then Call Get_vols  /* Get all volumes of Dcollect. */
If Index(Pcmd,'SORT') <> Ø Then /*Sort the table by user-chosen field.*/
Do  /* Canim */
 Flg = Ø
 Select
  When(Index(Pcmd,'DSET')  <>Ø) Then "Tbsort" Table_name
"Fields(K2,C,D)"
  When(Index(Pcmd,'MEMBER')<>Ø) Then "Tbsort" Table_name
"Fields(K1,C,D)"
  When(Index(Pcmd,'VOLUME')<>Ø) Then "Tbsort" Table_name
"Fields(K3,C,D)"
  When(Index(Pcmd,'CHGDT') <>Ø) Then "Tbsort" Table_name
"Fields(K5,Y1,D)"
  When(Index(Pcmd,'UID')   <>Ø) Then "Tbsort" Table_name
"Fields(K6,C,A)"
  When(Index(Pcmd,'CRTDT') <>Ø) Then "Tbsort" Table_name
"Fields(K4,Y1,D)"
  Otherwise  Do
             Fld = Word(Pcmd,2)      /* Field name entered with SORT. */
             Flg = 1
             Call Showmsg 2          /* "Sort field error" message.   */
             End
 End
If Flg = Ø Then Call Showmsg 16    /* "Table is sorted by: " message. */
End /* Canim */
      End /* BENIM */
     /*-------------------------------------------------------------*/
```

```
     /* User DOES CHOOSE a member from the query result table to     */
     /* process (Edit, Browse, Delete, Print, Copy, Getinfo).        */
     /*-------------------------------------------------------------*/
      ELSE
       DO /* OGLUM  */
         If Ztdsels > 1 Then     /* If user chooses more than 1 row.  */
                        Do
                         Call Showmsg 17 /* "Choose just one row" msg.*/
                         Address Tso
                         Signal LAB2   /* Display query result table. */
                        End
     R = CRP
     R = ABS(R)
  /*-------------------------------------------------------------------*/
  /* Remove the trailing blank characters from variables.              */
  /*-------------------------------------------------------------------*/
  Mem  = Strip(K1)
  Dset = Strip(K2)
  Vol  = Strip(K3)
  /*-------------------------------------------------------------------*/
  /* Process the action character entered by the user on the "Query    */
  /* Output Panel". It can be one of 6 values :                        */
  /* "E" (Edit member), "B"(Browse member), "D"(Delete member), "P"    */
  /* (Print member), "G"(Get more information) or "C"(Copy member).    */
  /*-------------------------------------------------------------------*/
 SELECT
  When (L = 'B') Then Call Proc_Browse_Member
  When (L = 'E') Then Call Proc_Edit_Member
  When (L = 'D') Then Call Proc_Delete_Member
  When (L = 'C') Then Call Proc_Copy_Member
  When (L = 'P') Then Call Proc_Print_Member
  When (L = 'G') Then Call Proc_Get_Meminfo
  Otherwise Nop
 END
       END /* OGLUM  */
   Ztdsels = Ø                 /* Position table as user last saw it. */
   "Tbtop"   Table_name
   Csrrow  = 'Csrrow('Crp')'
   "Tbskip" Table_name "Number("Ztdtop")"
 End    /* Do-forever     DENIS  */
RETURN    /* End_of_the_procedure PRESENT_RESULTS */
/*====================================================================*/
/*====================================================================*/
PROC_GET_MEMINFO:
/*-------------------------------------------------------------------*/
/* This procedure is aimed for getting statistics for a member chosen */
/* by user on Query Output panel in a real-time manner.              */
/*-------------------------------------------------------------------*/
If Sysdsn(DSET"("MEM")") <> 'OK'
               Then Call Showmsg 1        /* "Member not exist" msg. */
```

```
 Else
   Do /* LOKUM */
     /*-----------------------------------------------------------*/
     /* Get real-time member statistics for a specific member.    */
     /*-----------------------------------------------------------*/
     "Lminit  Dataid(ZIR)   Dataset('"DSET"')"
     "Lmopen  Dataid("ZIR") Option(Input)"
     "Lmmlist Dataid("ZIR") Option(List) Member(MEM) Stats(Yes)"
     Lmrc = Rc
     If Lmrc = Ø Then Call Proc_Fetch_Stats
                 Else Call Showmsg 6  /* "Can not get stats" message. */
     "Lmclose Dataid("ZIR")"
     "Lmfree  Dataid("ZIR")"
   End      /* LOKUM */
RETURN    /* End_of_the_procedure PROC_GET_MEMINFO */
/*===========================================================*/
/*===========================================================*/
PROC_BROWSE_MEMBER:
/*-----------------------------------------------------------*/
/* This procedure is aimed for browsing any member chosen by user.    */
/*-----------------------------------------------------------*/
If Sysdsn(DSET) <>  'OK' Then
                       Call Showmsg 5   /* "Dset not exist" message. */
Else
    If Sysdsn(DSET"("MEM")") <> 'OK'
                Then Call Showmsg 1    /* "Member not exist" message. */
                Else Do
                     "Browse Dataset('"DSET"("MEM")')"
                     If RC <> Ø Then
                     Do
                      Erc = Rc
                      Call Showmsg 3      /* "Cannot browse" message. */
                     End
                     End
RETURN    /* End_of_the_procedure PROC_BROWSE_MEMBER */
/*===========================================================*/
/*===========================================================*/
PROC_EDIT_MEMBER:
/*-----------------------------------------------------------*/
/* This procedure is aimed for editing any member chosen by user.     */
/*-----------------------------------------------------------*/
If Sysdsn(DSET) <>  'OK' Then
                       Call Showmsg 5   /* "Dset not exist" message. */
Else
    If Sysdsn(DSET"("MEM")") <> 'OK'
                Then Call Showmsg 1    /* "Member not exist" message. */
                Else Do
                      "Edit Dataset('"DSET"("MEM")')"
                      If RC > 4 Then
                         Do
```

```
                        Erc = Rc
                        Call Showmsg 4   /* "Cannot edit" message. */
                      End
                   End
RETURN    /* End_of_the_procedure PROC_EDIT_MEMBER */
/*====================================================================*/
/*====================================================================*/
PROC_DELETE_MEMBER:
/*--------------------------------------------------------------------*/
/* This procedure is aimed for deleting any member chosen by user.    */
/*--------------------------------------------------------------------*/
If Sysdsn(DSET) <>  'OK' Then
                      Call Showmsg 5   /* "Dset not exist" message. */
Else
If Sysdsn(DSET"("MEM")") =  'OK' Then
          Do  /* MARTAPVP-Do */
            /*----------------------------------------------------*/
            /* Decrease by 1 "number of query results" value after  */
            /* user physically deletes one member.                */
            /*----------------------------------------------------*/
            Qc = Qc - 1     /* Update "the number of records" value.*/
            Pan = PANEL3
            /*----------------------------------------------------*/
            /* Show user the "Delete Member Confirmation" panel.    */
            /*----------------------------------------------------*/
            Call POPUP Pan 3 53 "MEMBER DELETE"
            If Answer = "Y" Then
                DO  /* ZEYCAN-Do */
             "Tbdelete" Table_name            /* Delete from table.   */
            Address Tso
            Delete  Dset'('Mem')'            /* Delete from dataset. */
            Drc = Rc
            If Drc = Ø Then
                  Do
                   Call Write_Journal    /* Write a journal record.  */
                   Call Showmsg 9          /* "Member deleted." msg. */
                  End
                      Else
                   Call Showmsg 1Ø     /* "Member not deleted." msg. */
                End /* ZEYCAN-End */
           Else Call Showmsg 11     /* "Member won't be deleted" msg. */
           End  /* MARTAPV-End */
           Else Call Showmsg 1             /* "Member not exist" msg. */
RETURN    /* End_of_the_procedure PROC_DELETE_MEMBER */
/*====================================================================*/
/*====================================================================*/
PROC_COPY_MEMBER:
/*--------------------------------------------------------------------*/
/* This procedure is aimed for copying any member chosen by user.     */
/*--------------------------------------------------------------------*/
```

```
If Sysdsn(DSET) <>  'OK' Then
                      Call Showmsg 5   /* "Dset not exist" message. */
Else
If Sysdsn(DSET"("MEM")") =  'OK' Then
       Do   /* MARTINYA */
         Pan = PANEL7
         /*-----------------------------------------------------*/
         /* Show user the "Copy Member" panel.                  */
         /*-----------------------------------------------------*/
         Call POPUP Pan 4 6Ø "MEMBER COPY"
         If Sysdsn(TDSET) = 'OK' Then
            Do  /* MARTITA */
              /*-------------------------------------------------------*/
              /* Issue a Listdsi command to retrieve the DSORG of the */
              /* Target dataset entered by user in Member Copy panel. */
              /*-------------------------------------------------------*/
              x = Listdsi(Tdset)
              If SYSDSORG <> PO Then
                       Call Showmsg 26 /* "Target dset is not PO" msg.*/
              Else
              If Sysdsn(TDSET"("TMEM")") <> 'OK' Then
                    Call Proc_Icegener    /* Copy member by Icegener. */
                    Else Call Showmsg 21 /*"Target member exist." msg */
            End /* MARTITA */
                    Else Call Showmsg 2Ø /*"Target dset not found" msg*/
       End  /* MARTINYA */
       Else Call Showmsg 1                 /* "Member not exist" msg. */
RETURN    /* End_of_the_procedure PROC_COPY_MEMBER */
/*====================================================================*/
/*====================================================================*/
PROC_PRINT_MEMBER:
/*--------------------------------------------------------------------*/
/* This procedure is aimed for printing any member chosen by user.    */
/*--------------------------------------------------------------------*/
Address Ispexec
Zwinttl = Member Printing
"Vput (Zwinttl) Shared"
"Addpop Row(6) Column(25)"
"Display Panel(PANEL5)"
"Rempop"
If Resp='Y' Then
        Do        /* DO-Bebegim      */
         Address Tso
         "Pr Dataset("Dset"("Mem")) Cchar Class("Class") Dest("Dest")"
         Address Ispexec
         If Rc=Ø Then Call Showmsg 18        /* "Member printed" msg. */
                 Else Call Showmsg 14      /* "Error on Printds" msg. */
        End       /* DO-Bebegim      */
        Else Call Showmsg 19       /* "Member won't be printed." msg  */
RETURN    /* End_of_the_procedure PROC_PRINT_MEMBER */
```

```
/*====================================================================*/
/*====================================================================*/
PROC_FETCH_STATS:
/*--------------------------------------------------------------------*/
/* Fetch member statistics, which will differ depending on the Recfm */
/* value of the dataset in which user-chosen member resides.         */
/*--------------------------------------------------------------------*/
Call Proc_Get_Recfm   /* Get Recfm of the dataset.                    */
Firstch = Left(Zrecfm,1,1)
 SELECT /* Analyze Recfm value.  */
  When (Firstch  = 'U')                Then Call Show_Stats_U
  When (Firstch  = 'V' | Firstch = 'F') Then Call Show_Stats_FV
  Otherwise                            Say "Unknown Recfm = " Zrecfm
 END
RETURN    /* End_of_the_procedure PROC_FETCH_STATS */
/*====================================================================*/
/*====================================================================*/
PROC_GET_RECFM:
/*--------------------------------------------------------------------*/
/* Find out the Recfm of the PO dataset by LISTDSI command.          */
/*--------------------------------------------------------------------*/
IF SYSDSN(Dset) = 'OK' THEN
  DO                              /* If the base dataset exists, use */
    x = LISTDSI(Dset)             /* the LISTDSI function.            */
    IF x = Ø THEN                 /* If the function code is Ø, get   */
            Zrecfm = Sysrecfm     /* Recfm value.                     */
    ELSE
      DO
        SAY "Can not determine Recfm of dataset."
        SAY Sysmsglvl1              /* Display the system messages   */
        SAY Sysmsglvl2             /* and codes for LISTDSI command. */
        SAY 'Function code from LISTDSI is' x
        SAY 'Sysreason code from LISTDSI is' Sysreason
      END
  END
ELSE  Call Showmsg 5                    /* "Dset not exist" message. */
RETURN    /* End_of_the_procedure PROC_GET_RECFM  */
/*====================================================================*/
/*====================================================================*/
SHOW_STATS_U:
/*--------------------------------------------------------------------*/
/* Show statistics of an user-selected member which belongs to a data */
/* set of Recfm=U.                                                    */
/*                                                                    */
/* NOTE:                                                              */
/* The following variables will be stored in the function pool, so it */
/* is not necessary to use VPUT/VGET services to store/retrieve them. */
/* (They become available to session as soon as they're set.)         */
/*--------------------------------------------------------------------*/
If  ZLSIZE   = "" Then ZLSIZE   = "N/A" /* Load module size in hex.   */
```

```
If  ZLTTR    = "" Then ZLTTR    = "N/A" /* TTR of the member.         */
If  ZLALIAS  = "" Then ZLALIAS  = "N/A" /* Alias name.               */
If  ZLAC     = "" Then ZLAC     = "N/A" /* Authorization code.       */
If  ZLAMODE  = "" Then ZLAMODE  = "N/A" /* AMODE of the member.      */
If  ZLRMODE  = "" Then ZLRMODE  = "N/A" /* RMODE of the member.      */
If  ZLATTR   = "" Then ZLATTR   = "N/A" /* Load module attributes.   */
/*-------------------------------------------------------------------*/
/* Display "MORE INFO ON MEMBER" panel, which is PANEL2.             */
/*-------------------------------------------------------------------*/
Pan = PANEL2
Call POPUP Pan 16 47 "MEMBER STATISTICS *Load Module*"
RETURN    /* End_of_the_procedure SHOW_STATS_U  */
/*===================================================================*/
/*===================================================================*/
SHOW_STATS_FV:
/*-------------------------------------------------------------------*/
/* Show statistics of an user-selected member which belongs to a data */
/* set of Recfm=F/FB/FBA or Recfm=V/VB/VBA.                          */
/*                                                                   */
/* NOTE:                                                             */
/* The following variables will be stored in the function pool, so it */
/* is not necessary to use VPUT/VGET services to store/retrieve them. */
/* (They become available to session as soon as they're set.)        */
/*-------------------------------------------------------------------*/
If  ZLVERS   = "" Then ZLVERS   ="N/A" /* Version number.           */
If  ZLMOD    = "" Then ZLMOD    ="N/A" /* Modification number.      */
If  ZLCDATE  = "" Then ZLCDATE  ="N/A" /* Creation date.            */
If  ZLMDATE  = "" Then ZLMDATE  ="N/A" /* Last change date.         */
If  ZLMTIME  = "" Then ZLMTIME  ="N/A" /* Last change time.         */
If  ZLCNORC  = "" Then ZLCNORC  ="N/A" /* Current number of rows.   */
If  ZLINORC  = "" Then ZLINORC  ="N/A" /* Beginning num. of records. */
If  ZLMNORC  = "" Then ZLMNORC  ="N/A" /* Num. of changed records.  */
If  ZLUSER   = "" Then ZLUSER   ="N/A" /* Last user to change member. */
If  ZSCLM    = "" Then ZSCLM    ="N/A" /* Member is modified by SCLM? */
/*-------------------------------------------------------------------*/
/* Display "MORE INFO ON MEMBER" panel, which is PANEL1.             */
/*-------------------------------------------------------------------*/
Pan = PANEL1
Call POPUP Pan 19 47 "MEMBER STATISTICS"
RETURN    /* End_of_the_procedure SHOW_STATS_FV */
/*===================================================================*/
/*===================================================================*/
POPUP:
/*-------------------------------------------------------------------*/
/* This procedure is used to show PANEL1, PANEL2, PANEL3, PANEL6, and */
/* PANEL7 panels in the form of POPUP.                               */
/*-------------------------------------------------------------------*/
Arg Pnl Rw Cl Tit
Zwinttl = Tit
"Vput (Zwinttl) Shared"
```

```
"Addpop Row("Rw") Column("Cl")"
"Display Panel("Pnl")"
"Vget Spfkey Profile"
Do While( Spfkey=PFØ3 | Spfkey = PFØ4 )
   "Rempop"
   "Addpop Row("Rw") Column("Cl")"
   "Display Panel("Pnl")"
   "Vget Spfkey Profile"
End
"Rempop"
RETURN    /* End_of_the_procedure POPUP */
/*====================================================================*/
/*====================================================================*/
SHOWMSG:
/*--------------------------------------------------------------------*/
/* This procedure shows utility error messages to the user in a       */
/* box-style panel.                                                   */
/*--------------------------------------------------------------------*/
Address Ispexec
Arg gul
Mesaj = ""
Mesaj2= ""
If gul=1 Then Do
              Mesaj = "Member "MEM "does not exist."
              Mesaj2= "This member may have been deleted after the",
              "Inventory is built."
              End
If gul=2 Then Mesaj = "Sort field error. "Fld" is an invalid field."
If gul=3 Then Do
              Mesaj = "Can not BROWSE member "MEM" in dataset :"
              Mesaj2= DSET". Rc= "Erc
              End
If gul=4 Then Do
              Mesaj = "Can not EDIT member "MEM" in dataset :"
              Mesaj2= DSET". Rc= "Erc
              End
If gul=5 Then Do
 Mesaj  = "Dataset" DSET "is not found. "
 Mesaj2 = "It's an uncataloged dataset or deleted after INVENTORY is
built."
               End
If gul=6 Then Mesaj= "Can not get member statistics. LMMLIST Rc="Lmrc
if gul=7 Then Mesaj= "No records found. (query realized by IDCAMS)"
if gul=8 Then Mesaj= "Vsam Inventory dataset doesn't exist.",
                     "Build it by submitting 'JCL1'."
if gul=9 Then Do
                 Mesaj  = "Member "MEM" is deleted from the dataset:"
                 Mesaj2 = DSET
              End
if gul=1Ø Then Do
```

```
               Mesaj  = "Member "MEM" is NOT deleted from the dataset:"
                Mesaj2 = DSET". Rc= "Drc
               End
if gul=11 Then Do
              Mesaj  = "Member "MEM" won't be deleted from the dataset:"
                Mesaj2 = DSET"."
               End
if gul=12 Then Mesaj = "To terminate the PRINTING dialog, first press",
                       "<Enter> then <PF3> key."
if gul=13 Then Mesaj = "Report is printed."
if gul=14 Then Mesaj = "Error on the 'PRINTDS' command. RC = "RC
if gul=15 Then Mesaj = "Report will not be printed."
if gul=16 Then Mesaj = "Table is sorted by:"Word(Pcmd,2)"."
if gul=17 Then Mesaj = "Please choose just one row at a time."
if gul=18 Then Mesaj = "Member is printed."
if gul=19 Then Mesaj = "Member will not be printed."
if gul=2Ø Then Mesaj = "Target dataset "Tdset" doesn't exist."
if gul=21 Then Do
               Mesaj = "Enter a member which is not in dset "Tdset","
               Mesaj2= "since the member "Tmem" is already in there."
               End
if gul=22 Then Mesaj = "Member has been copied to "Tdset"("Tmem")."
if gul=23 Then Mesaj = "Member not copied to "Tdset"("Tmem").
Icegener_RC="Prc
if gul=24 Then Mesaj = "DFSORT error. RC = "Rc_Sort
if gul=25 Then Mesaj= "No records found. (query realized by DFSORT)"
if gul=26 Then Mesaj= "Dataset "TDSET "is not a partitioned dataset."
Pnl = PANEL4
 Zwinttl = " "  /* No title */
"Vput (Zwinttl) Shared"
"Addpop Row(8)  Column(76)"
"Display Panel("Pnl")"
"Rempop"
RETURN    /* End_of_the_procedure SHOWMSG  */
/*====================================================================*/
/*====================================================================*/
GET_LSTNG:
/*--------------------------------------------------------------------*/
/* This procedure is used for browsing the "Query Result" listing.    */
/* User will even get to print the report after having browsed it.    */
/*--------------------------------------------------------------------*/
Address Ispexec
"Browse Dataset("DsetOU")"
Zwinttl = Report Printing
"Vput (zwinttl) Shared"
"Addpop Row(6) Column(25)"
"Display Panel(PANEL5)"
"Vget Spfkey Profile"
Do While( Spfkey = PFØ3 | Spfkey = PFØ4 )    /* Do-DIVINA     */
  Call Showmsg 12     /* Tell user how to get out of the PRT panel. */
```

```
  "Rempop"
  "Browse Dataset("DsetOU")"
  "Addpop"
  "Display Panel(PANEL5)"
  "Vget Spfkey Profile"
End                                          /* Do-DIVINA End */
"Rempop"
If Resp='Y' Then
            Do        /* DO-Martinya     */
              Address Tso
              "Pr Dataset("DsetOU")  Class("Class") Dest("Dest")"
              Address Ispexec
              If Rc=Ø Then Call Showmsg 13    /* "Report printed" msg.*/
                      Else Call Showmsg 14  /* "Error on Printds" msg.*/
            End       /* DO-Martinya END */
            Else Call Showmsg 15   /* "Report won't be printed." msg. */
RETURN    /* End_of_the_procedure GET_LSTNG */
/*====================================================================*/
/*====================================================================*/
GET_VOLS:
/*--------------------------------------------------------------------*/
/* This procedure finds out the scope of the utility in terms of disk */
/* volume serial numbers and then display them on the panel "PANEL6". */
/*--------------------------------------------------------------------*/
Flag = 1
Type = 'S'                    /* It will be (S)pecific search.        */
Membrec ="      $ "  /* If 5th character is '$', it is VOLUME record. */
Address Tso
Call Read_By_Repro
Tot =""    /* Variable to hold all disk volume serial numbers.       */
Do m = 1 to Record.Ø
  Tot=Tot||Strip(Substr(Record.m,1Ø))
End
/*--------------------------------------------------------------------*/
/* Build a compound variable to hold all volsers (All_vols.).         */
/*--------------------------------------------------------------------*/
Do l = 1 to Length(Tot)/6
 All_vols.l = Substr(Tot,1,6)
 Tot = Substr(Tot,7)
 End
All_vols.Ø = l
Call Sort_Array            /* Sort All_vols. variable.                */
/*--------------------------------------------------------------------*/
/* Build back the Tot variable which will have ordered volsers.       */
/*--------------------------------------------------------------------*/
Tot = ""
Do i = 1 to All_Vols.Ø - 1
  Tot = All_vols.i || " " || Tot
End
/*--------------------------------------------------------------------*/
```

```
/* Prepare all variables to show all volser information in the ISPF   */
/* panel PANEL6.                                                      */
/*--------------------------------------------------------------------*/
Parse Var Tot R1  R2  R3  R4  R5  R6  R7  R8  R9  R1Ø,
              R11 R12 R13 R14 R15 R16 R17 R18 R19 R2Ø,
              R21 R22 R23 R24 R25 R26 R27 R28 R29 R3Ø,
              R31 R32 R33 R34 R35 R36 R37 R38 R39 R4Ø,
              R41 R42 R43 R44 R45 R46 R47 R48 R49 R5Ø,
              R51 R52 R53 R54 R55 R56 R57 R58 R59 R6Ø,
              R61 R62 R63 R64 R65 R66 R67 R68 R69 R7Ø,
              R71 R72 R73 R74 R75 R76 R77 R78 R79 R8Ø,
              R81 R82 R83 R84 R85 R86 R87 R88 R89 R9Ø,
              R91 R92 R93 R94 R95 R96 R97 R98 R99 R1ØØ
Address Ispexec
Pan = PANEL6
Call POPUP Pan 2Ø 1 "DCOLLECT SCOPE"
RETURN    /* End_of_the_procedure GET_VOLS  */
/*====================================================================*/
/*====================================================================*/
WRITE_JOURNAL:
/*--------------------------------------------------------------------*/
/* This procedure creates a record for each member deleted by the     */
/* utility. The record will contain the following fields:             */
/* User who deleted the member, Deletion time & date, Dataset from    */
/* which the member is deleted.                                       */
/*--------------------------------------------------------------------*/
Address Tso
Rs = Sysdsn(Logdset)
If Rs<>"OK"  Then
   DO /* Do-Selin */
    "Alloc Fi(Jrnl) Space(1,1) Cylinders Lrecl(9Ø) Recfm(F,B)
Blksize(Ø),
    Reuse Dsorg(Ps) New Catalog Da("Logdset")"
     "Free F(Jrnl)"
    "Alloc Da("Logdset") F(Jrnl) Old"
    "Newstack"
    Queue "         JOURNAL LOG FOR DELETED MEMBERS"
    Queue " "
    Queue "USERID  DELETION DATE & TIME VOLUME DATASET & MEMBER NAME"
    Queue "======= ================== ====== ====================="
    Queue ""
    "Execio * Diskw Jrnl (Finis"
    "Delstack"
    "Free File(Jrnl)"
   END  /* Do-Selin END */
"Alloc Da("Logdset") F(Jrnl) Mod"
"Newstack"
Queue Userid() Date() Time() Vol Dset"("Mem")"
Queue ""  /* End of the stack. */
"Execio 1 Diskw Jrnl (Finis"
```

```
"Delstack"
"Free File(Jrnl)"
RETURN    /* End_of_the_procedure WRITE_JOURNAL */
/*====================================================================*/
/*====================================================================*/
SORT_ARRAY:
/*--------------------------------------------------------------------*/
/* This procedure sorts the array in the All_vols. stem variable,     */
/* which contains all disk volume volsers within the DCOLLECT scope.  */
/*--------------------------------------------------------------------*/
Swapped = 1
Do While Swapped
   Swapped = Ø
   Do i = 1 to (All_vols.Ø-1)
      j = i + 1
      If Substr(All_vols.i,1Ø7,1Ø) < Substr(All_vols.j,1Ø7,1Ø)
        Then Do
               Hold       = All_vols.i
               All_vols.i = All_vols.j
               All_vols.j = Hold
               Swapped    = 1
             End
   End i
End
RETURN /* End-of-the-procedure SORT_ARRAY */
/*====================================================================*/
/*====================================================================*/
PROC_ICEGENER:
/*--------------------------------------------------------------------*/
/* This procedure is aimed for member-copy operation.                 */
/*--------------------------------------------------------------------*/
Address Tso
"Alloc Fi(Sysut1)   Da('"Dset"("Mem")')   Shr Reuse"
"Alloc Fi(Sysut2)   Da('"Tdset"("Tmem")') Shr Reuse"
"Alloc Fi(Sysin)    Dummy Reuse"
"Alloc Fi(Sysprint) Dummy Reuse"
"Ispexec Select Pgm(ICEGENER)"
Prc = Rc
 If Rc = Ø Then Call Showmsg 22   /* "Member copy is not OK." message */
           Else Call Showmsg 23   /* "Member copy is OK."     message */
"Free File(Sysut1,Sysut2,Sysin)"
RETURN /* End-of-the-procedure PROC_ICEGENER */
/*====================================================================*/
```

SIL_REXX3

```
/* REXX */
/*--------------------------------------------------------------------*
* REXX NAME :                                                         *
```

```
* Rexx3     ( Called from the Jcl JCL2.)                              *
* FUNCTION : This REXX is used for calling Member Search Utility in   *
*            batch. It reads the query parameters from the Input data *
*            set, fetches corresponding Inventory records, then write *
*            them to the SYSTSPRT dataset.                            *
* DATASETS USED :                                                     *
* Exp.Memsrch.Vsam     (In) -> Member Inventory dataset        (VSAM) *
* Exp.Memsrch.Output  (Out) -> Query  Result    dataset        (SEQL) *
* Exp.Memsrch.Batch   (In)  -> Batch Input      dataset        (SEQL) *
*---------------------------------------------------------------------*/
"Prof Nopref"
Status = Msg('Off')
Trace Off
DsetVS  = Exp.Memsrch.Vsam    /* Inventory    dataset.   */
DsetOU  = Exp.Memsrch.Output  /* Query Output dataset.   */
DsetBA  = Exp.Memsrch.Batch   /* Query Input   dataset.  */
/*---------------------------------------------------------------------*/
/* Read Query Input dataset and process each of its record.          */
/*---------------------------------------------------------------------*/
"Alloc Fi(YY) Da("DsetBA") Shr Reuse"
"Execio * Diskr "YY" (Stem Denis. Finis"
Do k = 7 to Denis.Ø  /* First 7 records are dummy. */
 Parse Var Denis.k Membrec Type
 Membrec = Strip(Membrec)
 Type    = Strip(Type)
Say "---------------------------------------------------------------
-----------------"
 Say "QUERY : "k-6
 Say "Member name.......:" Membrec
 Drop Record.  /* Un-assign the variable Record.  */
 If Index('SG',Type) <> Ø Then Call Read_By_Repro   /* Read by IDCAMS.*/
                          Else Call Read_By_Dfsort  /* Read by DFSORT.*/
End
"Free Fi(YY)"
EXIT Ø
/*=====================================================================*/
/*=====================================================================*/
READ_BY_REPRO:
"Alloc Fi(In) Da("DsetVS") Shr Reuse"
IF Sysdsn(DsetOU) = 'OK' Then Delete DsetOU
"Alloc Fi(Out) Blksize(Ø) Reuse Lrecl(86) New Catalog Recfm(F B),
Da("DsetOU") Dsorg(PS) Tracks Space(5 1)"
If Type = S Then
           Do
            Line1 = "Infile(In) Outfile(Out) Fromkey('"Membrec" ') "
            Line2 = "Tokey('"Membrec" ')"
           End
          ELSE
           Do
            Line1 = "Infile(In) Outfile(Out) Fromkey('"Membrec"') "
```

```
            Line2 = "Tokey('"Membrec"')"
           End
"Repro " Line1 Line2
RC_Repro = Rc
"Free Fi(In Out)"
"Alloc Fi(XX) Da("DsetOU") Shr Reuse"
"Execio * Diskr "XX" (Stem Record. Finis"
Num_rec = Record.Ø   /* Number of records found against user query.*/
Call Write_To_Sysprint
If Num_rec = Ø Then Say "No records found against this query."
Say ""
Say ""
"Free Fi(XX)"
RETURN    /* End_of_the-procedure READ_BY_REPRO */
/*====================================================================*/
/*====================================================================*/
READ_BY_DFSORT:
"Alloc Fi(SortIN) Da("DsetVS") Shr Reuse"
IF Sysdsn(DsetOU) = 'OK' Then Delete DsetOU
"Alloc Fi(SortOUT) Blksize(Ø) Reuse Lrecl(86) New Catalog Recfm(F B),
Da("DsetOU") Dsorg(PS) Tracks Space(5 1)"
"Alloc Fi(Sysout) Space(2,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse"
"Alloc Fi(Sysin)  Space(1,1) Track Lrecl(8Ø) Recfm(f) Reuse"
If Type = T Then   /* Member-search type : Sub-string   */
   DO      /* 999 */
     Copy_sysin1= " SORT FIELDS=(1,86,CH,A)"
     Copy_sysin2= " INCLUDE FORMAT=SS,COND=(1,8,EQ,C'"Membrec"')"
     "Newstack"
     Queue Copy_sysin1
     Queue Copy_sysin2
     Queue ""
     "Execio 3 Diskw Sysin (Finis"
     "Delstack"
   END     /* 999 */
If Type = M Then   /* Member-search type : Mask         */
   DO      /* 888 */
    If Length(Membrec) <> 8 Then
                      Do
                       Say "Search Type.......: MASK"
                       Say "Please enter a mask field of 8 characters."
                       Say ""
                       Say ""
                       Return
                      End
    "Newstack"
    j = Ø
    Do i = 1 to 8
      If Substr(Membrec,i,1) <> '%' Then
            Do
               j = j + 1
```

```
               Array.j = i
             End
    End
    Array.Ø = j
    Queue " SORT FIELDS=(1,86,CH,A)"
    p = Array.1
    s = Substr(Membrec,p,1)
    If Array.Ø = 1 Then
         Queue " INCLUDE FORMAT=SS,COND=("p",1,EQ,C'"s"')"
                   Else
         Queue " INCLUDE FORMAT=SS,COND=("p",1,EQ,C'"s"',AND,
"
    Do i = 2 to Array.Ø
      p = Array.i
      s = Substr(Membrec,p,1)
      If i <> Array.Ø Then Queue "         "p",1,EQ,C'"s"',AND,"
                       Else Queue "         "p",1,EQ,C'"s"')"
    End
    Queue ""
    "Execio * Diskw Sysin (Finis"
    "Delstack"
  END      /* 888 */
"Call 'Sys1.Sortlpa(Sort)'"
Rc_Sort = Rc
"Free Fi(Sortin,Sortout,Sysout)"
If Rc_Sort = Ø Then Do
                     "Alloc Fi(XX) Da("DsetOU") Shr Reuse"
                     "Execio * Diskr "XX" (Stem Record. Finis"
                     Num_rec = Record.Ø   /* Number of records found.*/
                     Call Write_To_Sysprint
                     "Free Fi(XX)"
                     If Num_rec = Ø Then
                           Say "No records found against this query."
                    End
               Else Say "DFSORT error. RC="Rc_Sort
Say ""
Say ""
RETURN    /* End_of_the-procedure READ_BY_DFSORT */
/*====================================================================*/
/*====================================================================*/
WRITE_TO_SYSPRINT:
If Type = "S" Then Say "Search type.......: SPECIFIC"
If Type = "G" Then Say "Search type.......: GENERIC"
If Type = "T" Then Say "Search Type.......: SUB-STRING"
If Type = "M" Then Say "Search Type.......: MASK"
Say "Number of records.:" Num_rec
Say " "
Say "Member   Data-set-Name                                  Volume
Crtdte",
    "  Chgdte   Uid    "
```

```
Say "======== ========================================== ======
========",
    "======== ======="
Do i = 1 to Record.Ø
  Say Record.i
End
RETURN    /* End_of_the-procedure WRITE_TO_SYSPRINT */
/*================================================================*/
```

Atalay Gul
Systems Programmer
Azertia SA (Spain)

IEFACTRT bug with USS batch jobs

We recently discovered a bug in our IEFACTRT SMF exit! For USS batch jobs invoking BPXBATCH, the 'header' was produced several times:

```
J E S 2  J O B  L O G  --  S Y S T E M  S M V S  --  N O D E  J E S 2 S M V S
Ø
 12.Ø5.52 JOBØ8825 ---- THURSDAY,  Ø6 MAR 2ØØ3 ----
 12.Ø5.52 JOBØ8825  TSS7ØØØI SXSPØØ1 Last-Used Ø6 Mar Ø3 12:Ø5 System=XMØ1
Facility=BATCH
 12.Ø5.52 JOBØ8825  TSS7ØØ1I Count=54479 Mode=Warn Locktime=None Name=******
 12.Ø5.52 JOBØ8825  $HASP373 SXSPØØ1A STARTED - WLM INIT  - SRVCLASS JES_3Ø
- SYS XMØ1
 12.Ø5.52 JOBØ8825  IEF4Ø3I SXSPØØ1A - STARTED - TIME=12.Ø5.52
 12.Ø5.52 JOBØ8825  -                                          --TIMINGS
(MINS.)--
 12.Ø5.52 JOBØ8825  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB
ELAPS   SERV  $ header
 12.Ø5.52 JOBØ8825  -SXSPØØ1A STEPØ1               ØØ     48    .ØØ    .ØØ
.Ø   1152
 12.Ø5.53 JOBØ8825  -                                          --TIMINGS
(MINS.)--
 12.Ø5.53 JOBØ8825  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB
ELAPS   SERV  $ header
 12.Ø5.53 JOBØ8825  -SXSPØØ1A *OMVSEX              ØØ     48    .ØØ    .ØØ
.Ø   1279
 12.Ø5.53 JOBØ8825  -                                          --TIMINGS
(MINS.)--
 12.Ø5.53 JOBØ8825  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB
ELAPS   SERV  $ header
```

```
 12.Ø5.53 JOBØ8825  -SXSPØØ1A *OMVSEX              ØØ     84    .ØØ    .ØØ
.Ø   1898
 12.Ø5.53 JOBØ8825  -SXSPØØ1A STEPØ2               ØØ     48    .ØØ    .ØØ
.Ø    873
 12.Ø5.53 JOBØ8825  -SXSPØØ1A *OMVSEX              ØØ     48    .ØØ    .ØØ
.Ø   1254
 12.Ø5.53 JOBØ8825  -SXSPØØ1A *OMVSEX              ØØ     84    .ØØ    .ØØ
.Ø   177Ø
 12.Ø5.53 JOBØ8825  IEF4Ø4I SXSPØØ1A - ENDED - TIME=12.Ø5.53
 12.Ø5.53 JOBØ8825  -SXSPØØ1A ENDED.  NAME-SYSTEM TEAM          TOTAL CPU
TIME=   .ØØ  TOTAL
 12.Ø5.53 JOBØ8825  $HASP395 SXSPØØ1A ENDED
Ø------ JES2 JOB STATISTICS ------
-  Ø6 MAR 2ØØ3 JOB EXECUTION DATE
-           3Ø CARDS READ
-           94 SYSOUT PRINT RECORDS
-            Ø SYSOUT PUNCH RECORDS
-            5 SYSOUT SPOOL KBYTES
-         Ø.Ø1 MINUTES EXECUTION TIME
```

This happened because a USS job can have multiple steps with the same number! A jobstep can have multiple substeps, and SMF produces a record-30 subtype 4 entry for each substep.

To avoid this bug, you should test the SMF30SSN field (Sub-Step Number) before producing the header.

IEFACTRT MODIFICATION

IEFACTRT should be modified to test the SMF30SSN field:

```
…
*
*         HANDLE STEP TERMINATION CONDITIONS
*
STEPTERM DS    ØH                 ENTRY FROM STEP TERMINATION
         LA    DRKR6,WTO1TXT      GET ADDRESS OF WTO TEXT AREA
         USING LINE2,DRKR6        SET UP ADDRESSABILITY TO LINE
         L     DRKR1,PARMINDC     LOAD ADDRESS OF STEP NUMBER
         CLI   K1(DRKR1),K1       IS IT STEP 1 ? FOR HEADER
         BNE   NOTFIRST           NO
*
         LR    DRKR4,DRKR9        GET RECORD ADDRESS          MSEIPO4
         A     DRKR4,SMF3ØIOF     POINT TO ID SEGMENT         MSEIPO4
         USING SMF3ØID,DRKR4                                  MSEIPO4
*
         CLC   SMF3ØSSN,=F'Ø'
$ add this test
```

```
         BNE   NOTFIRST
*
         MVC   WTO1TXT,LINE3A      MOVE HEADER LINE 1 TO OUTPUT
         BAL   DRKR14,MSGRTN       PUT OUT LINE
         MVC   WTO1TXT,LINE3       MOVE SECOND HEADER
BAL   DRKR14,MSGRTN       PUT IT OUT
*
NOTFIRST MVI   WTO1TXT,C' '        INIT OUTPUT LINE TO BLANKS
         MVC   WTO1TXT+K1(L'WTO1TXT-K1),WTO1TXT COPY BLANKS
         MVC   JOBNAME,JMRJOB      MOVE JOBNAME INTO LINE
         L     DRKR1,PSAAOLD       GET ADDRESS OF CURRENT ASCB
         L     DRKR1,ASCBJBNI-ASCB(,DRKR1) GET ADDRESS OF CHNAME
         LTR   DRKR1,DRKR1             BATCH JOB                OY51685
         BNZ   GOTASCB                 YES, GO ON               OY51685
         L     DRKR1,PSAAOLD           RELOAD ASCB              OY51685
         L     DRKR1,ASCBJBNS-ASCB(,DRKR1) GET STC/MOUNT ID     OY51685
…
```

After this modification, you get the correct output:

```
J E S 2  J O B  L O G  --  S Y S T E M  S M V S  --  N O D E  J E S 2 S M V S
Ø
 14.45.Ø8 JOBØ8836 ---- THURSDAY,  Ø6 MAR 2ØØ3 ----
 14.45.Ø8 JOBØ8836  TSS7ØØØI SXSPØØ1 Last-Used Ø6 Mar Ø3 14:45 System=XMØ1
Facility=BATCH
 14.45.Ø8 JOBØ8836  TSS7ØØ1I Count=54499 Mode=Warn Locktime=None Name=******
 14.45.Ø9 JOBØ8836  $HASP373 SXSPØØ1A STARTED - WLM INIT  - SRVCLASS JES_3Ø
- SYS XMØ1
 14.45.Ø9 JOBØ8836  IEF4Ø3I SXSPØØ1A - STARTED - TIME=14.45.Ø9
 14.45.Ø9 JOBØ8836  -                                          --TIMINGS
(MINW.)--
 14.45.Ø9 JOBØ8836  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB
ELAPS   SERV
 14.45.Ø9 JOBØ8836  -SXSPØØ1A STEPØ1               ØØ     48    .ØØ    .ØØ
.Ø    89Ø
 14.45.Ø9 JOBØ8836  -SXSPØØ1A *OMVSEX             ØØ     48    .ØØ    .ØØ
.Ø   1119
 14.45.Ø9 JOBØ8836  -SXSPØØ1A *OMVSEX             ØØ     84    .ØØ    .ØØ
.Ø   1681
 14.45.Ø9 JOBØ8836  -SXSPØØ1A STEPØ2               ØØ     48    .ØØ    .ØØ
.Ø    88Ø
 14.45.Ø9 JOBØ8836  -SXSPØØ1A *OMVSEX             ØØ     48    .ØØ    .ØØ
.Ø   1117
 14.45.Ø9 JOBØ8836  -SXSPØØ1A *OMVSEX             ØØ     84    .ØØ    .ØØ
.Ø   1641
 14.45.Ø9 JOBØ8836  IEF4Ø4I SXSPØØ1A - ENDED - TIME=14.45.Ø9
 14.45.Ø9 JOBØ8836  -SXSPØØ1A ENDED.  NAME-SYSTEM TEAM          TOTAL CPU
TIME=   .ØØ  TOTAL
 14.45.Ø9 JOBØ8836  $HASP395 SXSPØØ1A ENDED
Ø------ JES2 JOB STATISTICS ------
```

```
-  Ø6 MAR 2ØØ3 JOB EXECUTION DATE
-           3Ø CARDS READ
-           9Ø SYSOUT PRINT RECORDS
-            Ø SYSOUT PUNCH RECORDS
-            5 SYSOUT SPOOL KBYTES
-         Ø.ØØ MINUTES EXECUTION TIME
```

Xbridge Host Data Connect from Xbridge Systems now supports Microsoft Office 2003. Xbridge Host Data Connect enables Windows, Web, and mobile applications to quickly and securely access virtually all OS/390 and z/OS mainframe data and to see the data in the format of the requesting application, all in real time.

Specifically, Host Data Connect allows users to directly access VSAM (Virtual Storage Access Method), QSAM (Queued Sequential Access Method), DB2, IMS, and IAM data. Host Data Connect is built around a three-tiered architecture that uses Microsoft technology as a key component.

For further information contact:
Xbridge Systems, 15055 Los Gatos Blvd, Suite 110, Los Gatos, CA 95032, USA.
Tel: (408) 356 1515.
URL: http://www.xbridgesystems.com/product/hdc.htm.

* * *

ObjectStar International has announced Release 4.0 of ObjectStar. the product allows for the creation, integration, and adaptation of composite applications in both complex and simple, heterogeneous, enterprise environments.

The new graphical workbench allows front-end and back-end developers to build, test, and deploy integrated applications.

At the heart of the Release 4.0 product set is the ObjectStar Integration Foundation, containing a table-driven MetaStor, a business rules engine, as well as administrative and security functions. It runs on z/OS, Solaris, Windows, and Linux.

ObjectStar Database Gateways use native services to deliver access to a wide array of back-end systems, including DB2, IMS, CICS/DLI, VSAM, Model 204, CA-IDMS, as well as most SQL databases.

For further information contact:
ObjectStar, Grove House, Lutyens Close, Chineham Court, Basingstoke, Hamps RG24 8AG, UK.
Tel: (08701) 624 930.
URL: http://www.objectstar.com/news/release4.htm.

* * *

DWL has released Version 4.5 of DWL Customer, which has enhanced customer management functions to support legislative compliance issues around privacy and the US 'Do Not Call' list. DWL Customer serves as the enterprise's system of record for all customer-related information and transactions.

The latest version not only stores default state/province privacy rules, but also provides the ability to override those defaults for the capture of enhanced information about a customer regarding their opt-in and opt-out preferences, including preferred channels and contact times.

This version is DWL's first release for Z/OS users on the zSeries mainframe server. DWL Customer's database will run on zSeries and DWL Customer's application server will run on pSeries.

For further information contact:
DWL, 230 Richmond Street, East Toronto, ON, Canada M5A 1P4.
Tel: (416) 364 2045.
URL: http://www.dwl.com/na/news/press/news_press_031021_4_point_5.php.