125

VM update

January 1997

In this issue

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: 100325.3711@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 817 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

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

Editor
Trevor Eddolls

Subscriptions and back-issues

A year's subscription to *VM Update*, comprising twelve monthly issues, costs £165.00 in the UK; $250.00 in the USA and Canada; £171.00 in Europe; £177.00 in Australasia and Japan; and £175.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.00 ($21.00) each including postage.

VM Update on-line

Code from *VM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label. Code is also available from our bulletin boards in the USA (630 980 4581 or 4751) or the UK (01635 30998); you will need the user-id and password shown on your address label.

VM batch FTP

We needed to automatically move files from our VM system to a Novell server without using the Network File System or LANRES. To do this VMBFTP EXEC was created. From this base the functionality was expanded to include Unix, VAX/VMS, and VM Shared File System directories. Other system types could be added to the EXEC with only minor modifications.

I use a service machine which calls a REXX EXEC that interfaces to the SFS on VM to move files between our VM/ESA 1.2.2 (TCP/IP at Release 2.2, FTP at Release 2.3) and several different target systems. These are Novell file servers (running HellSoft FTP Version 1.10), VAX/VMS, Unix (Solaris), and VM SFS. The server runs the VMBFTP EXEC every 15 minutes on my system, and it is very flexible in how it can handle new files and destinations.

The real key is having an SFS directory tree set up to match the directory tree on the target system.

The SFS directory tree looks like this:

```
SERVER.Volume.Root_Dir.Sub_Dir.Sub_Dir2.Sub_Dir3.Sub_Dir4.Sub_Dir5
```

To get a file to Novell server LOCAL1 on volume SYSTEM in the directory /LOCAL/PCMODS/DOWNLOAD, the file would reside in an SFS directory called LOCAL1.SYSTEM.LOCAL.PCMODS. DOWNLOAD. The problem with this is that the SFS system will allow only eight subdirectory levels, while other systems will allow more (how many depends on the target system).

The program requires a list of ‘servers’ in the root directory of the VM service machine (this is the SERVER LIST file). Each server is the name of a root of an SFS tree. In that root directory is a file called ‘IP ADDRESS’ containing an IP address, user name, password, and system type. Note: the only users who have authority on VM to read the root level files should be the service machine and the administrator setting up the IP ADDRESS file.

The program will then access each directory in the SFS tree to see if there are any files to FTP. If any are found, it will do a size check. Sizes are defined as a maximum number of 4KB blocks in a given time frame. If the file is too large, the program will delay moving the file until 'after hours' (defined as a time with no size limit). These time frames and limits are defined in an array called 'ZONE'. My service machine also takes care of LPRing files to several printers so I care about the amount of time it may take to move a file during the day.

To accommodate the lowest common denominator (the PC file system) the file will be renamed during the FTP to an eight-character filename and a three-character extension (filetype). The software will assign the extension as the number of days (including the current day) since 1 January of the current year (REXX Date(D) function).

To prevent overwriting a file on the target system, an FTP session is opened against the target system and a list (LS) command is executed. The output of the LS is scanned to look for name conflicts. If the file-id already exists on the target system, the last character of the extension is changed. The new character will be the first character in the alphabet (A-Z) that is unused as a tie breaker.

For example: two files are located in a SFS directory on VM called SAMPLE FILEA and SAMPLE FILEB (for the purpose of example today is 6 January), and they are downloaded to a PC. The first file (FILEA) will be called SAMPLE.006. The second file will then find the name SAMPLE.006 in use and will scan for SAMPLE.00A. The PC will report no file by that name, so it will issue a message to the VM service machine's console (and send a file with that name to any VM user-ids defined as part of a nickname FTPERR), then FTP the file with the new name SAMPLE.00A. Tomorrow (7 January) a file on VM called SAMPLE DATA will be downloaded with the name SAMPLE.007, and no messages will be issued.

This scheme will allow for 26 duplicate file names every 10 days. You may want to customize the code in this area if it proves to be an issue with your site.

If the software does not see a message "226 Transfer complete" (or "250 Transfer successful" message from a VM system) from the FTP, the program will issue a warning message to the console. If the FTP is successful to the target system, the program will delete the file from the SFS directory on VM (cleaning up ready for the next time). If the FTP is unsuccessful, a message will be written to the console, then the file will be retained for the next iteration of the program and it will try to FTP again.

Use this procedure to add new directories/servers for use.

Adding a new destination to the Auto FTP is done by VMBFTP:

- Are you adding a directory on a server that does not already have an Auto FTP?

  Skip to step 1

- Are you adding a new directory on a server that already receives an Auto FTP?

  Skip to step 6

On the target system you need to have completed the following:

1 Install and have functioning FTP software.

2 Create a user-id with R/W access to the target volume.

3 Pass on to the VMBFTP support person the following information:

   - IP address for the new server.
   - User name to log-on with.
   - Password for this user-id.
   - Volume/directory layout that includes **all** directories from the root level. (This is used to build an SFS tree to match the PC volumes.)
   - System type (VMS, Unix, Novell, VMSFS).

On the host mainframe side:

4 Create an SFS tree with the root name matching the target server’s name. This can be a name that is only used internally in VM. For example, I will call it ‘PC1’.

5 Create a file in the root directory (PC1) called ‘IP ADDRESS’. This file has several tokens on one record. They are positional parameters:

- The IP address (in dot notation or a named system).
- The user name that is to be used to sign-on to the remote system.
- Password to use for the remote system.
- System type. Currently acceptable systems are: Novell, VMS, VM SFS (VM Shared File System), and Unix.

  Note: for Unix, enter the user-id and password in the correct case (mixed, upper, lower). No translation to upper or lower case will occur for these fields.

A sample IP ADDRESS file would look like:

```
1.1.255.1 USERID Password System_Type
```

6 Create a subdirectory with the name of the volume in the VM SFS (PC1.Volid).

7 Create a subdirectory with the same path to the download destination in the VM SFS (PC1.Volid.Path).

8 Give Write and NewWrite authority to the directories to the VM service machine (we use File control directories).

9 On the service machine’s root directory (or on some accessed mini-disk or directory) you will need to create or modify a file called SERVER LIST. This file contains a list of the root directories in the default filepool to search for files to download. All entries are used in an ACCESS command so they **must** end with a full stop/period (.) to comply with the syntax of the ACCESS command for a root directory. Comments are allowed in the file. Comment lines start with the characters ‘* ’ in columns 1 and 2 (the space **must** be present).

A sample SERVER LIST file would look like:

```
* This is a file of SFS high-level directories.
* Always include a period after the directory Id.
```

```
PC1.
SERVERB.
```

The service machine will not need any code changes to have any of these modifications done. The service machine is looping 24 hours a day, and at the top of each quarter hour it will read the SERVER LIST file. It will then access directory tree(s) in turn on the VM system. If it finds a file in any directory (**except** the root directory), it will FTP the file to the same location on the target system.

To have a file moved to the target directory, copy it to the correct VM shared file directory and the service machine will move the file at the next iteration of the program.

SERVICE MACHINE PROFILE EXEC

```
/* PROFILE EXEC for VMBFTP service machine */
'CP SP CON * START'
trace error
Processed = 0
Rscs_Cntr = 0
W = Time(R)      /* Reset timer   */
Do Forever
   'CP CLO CON'
   Do Day = 1 to 24           /* Run this loop once per hour         */
      Do Hour = 1 to 4        /* Do this loop every 15 minutes       */
         Call VMBFTP          /* Let's check for FTP items in the SFS*/
         Do QtrHour = 1 to 3  /* Do this every  5 minutes            */
            Do fivemin = 1 to 5    /* Do this loop every minute      */
               Do Min = 1 to 2
            /* Do this loop every 30 seconds (for fast response time).*/
                  Wait_Start = Time(r)
               /* Print routing by class to RSCS printers */
               /* Print routing to remote TCP printers W. LPR     */
                  Wait = 30 - (Time(e) + .5)
                  Parse Var Wait Seconds '.' Milsecs
                  'EXECIO 0 CP ( STRING SLEEP' Seconds 'SEC'
                  /* Do overnight processing at 3 am */
                  If Time(H) = 3 & ¬Processed then do
                              /* Call EXEC to run overnight process  */
                     Processed = 1
                  End
                  Else If Time(H) ¬= 3 & Processed then do
    /* Reset processed flag to allow for tomorrow night's processing */
                     Processed = 0
                  End
               End   /* Min */
               /* Do any process every minute  */
            End   /* Five Min */
            /* Do any process every 5 minutes */
         End   /* Qrt Hour */
      End   /* Hour */
   End   /* Day */
End   /* Forever */
Exit
```

VMBFTP EXEC

```
/*  This EXEC will perform an FTP for moving files from VM/ESA
    system to remote systems (PC fileservers, VMS, UNIX, and other VM
    systems).

Input files:   Server List (Root Directory). This file contains a list
                           of SFS root directories that correspond to
                           remote system names.

Output files:  WARNING MAIL (user mail). This is mail that the server
                           may create to warn of errors.  It will be
                           sent to an internal nickname of FTPERR
*/
Mail_Opts = 'NOACK NOLOG NOEDIT NOPROMPT'
Server_List = 'SERVER LIST'
Mail_Fl   = 'TEMP MAILFILE A'
Error_Log = 'FTPERROR LOG F'
              /* a log file - File mode can be determined by the site */
LS_Output = ''    /* null string */
Low_Alpha = 'abcdefghijklmnopqrstuvwxyz'
Up_Alpha  = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
Mixed_Alpha = 'ZXWVTRQPONMLKJIHGFECB '        /* File modes to ck for */
Max_Window = 5 /* The max number of minutes the process will run and
                  FTP files outbound.  If the duration of the routine
                  extends beyond Max_Window, a message will be placed
                  on the console for each file skipped.  They will be
                  processed on the next iteration of the routine.
               */
/* ZONE. Array is used to determine the max file size by time of day */
Run_Time1 = Time()
Run_Time = Time(m)
                 /* Static start time - used for Zones and Max_Window*/
Zone.Ø = 4     /* Number of time zones per day */
Zone.Ø.Ø = 3   /* Number of items to check per zone  */
Zone.1.1 = Ø       /* Zone.n.1 = Start time for the zone (Time(m))  */
Zone.1.2 = 48Ø     /* Zone.n.2 = End time for zone                  */
Zone.1.3 = Ø       /* Zone.n.3 = Max number of blocks allowed to    */
Zone.2.1 = 481      /*   during this time zone.  When = Ø , any size */
Zone.2.2 = 1Ø8Ø     /*   file can be moved.                          */
Zone.2.3 = 2ØØØ
Zone.3.1 = 1Ø81
Zone.3.2 = 132Ø
Zone.3.3 = 6ØØØ
Zone.4.1 = 1321
Zone.4.2 = 144Ø
Zone.4.3 = Ø
/* Find out what the max allowable file size to transfer is for now */
Do Lp = 1 to Zone.Ø
   If Run_Time > Zone.Lp.1 & Run_Time <= Zone.Lp.2 then do
      If Zone.Lp.3 = Ø then do
         Max_Blocks = 99999
      End
      Else do
         Max_Blocks = Zone.Lp.3
      End
   End
End
/* Let's start out by finding an open file mode I can use with the SFS
   directories later.
*/
'Pipe',
   'COMMAND QUERY DISK ',
   '| Drop 1 ',
   '| Specs 13.1 1 ',
   '| Join * ',
   '| Var  Disk_Fm' /*
   '| Console '     */
Do Lp = 1 to Length(Mixed_Alpha) Until
Pos(Substr(Mixed_Alpha,Lp,1),Disk_Fm) = Ø
   Last = Lp
End
If Last = Length(Mixed_Alpha) then do
   Call Mail_Error VMDFTPØØ1 'No open filemodes'
End
Else do
   Open_Fm = Substr(Mixed_Alpha,Last,1)
End
/*  Find out which servers I am to hit with this run... */
'Pipe',
   '<' Server_List ,
   '| NFind *_',   /* Drop any recs that start with '* ' comments */
   '| Strip ',
   '| STEM Servers. ' /*
   '| Console '       */
Do Lp = 1 to Servers.Ø
   /* For each server I have a file in the root directory that contains
```

```
      an IP address and a password for the server.
   */
   'PIPE CMS ACCESS VMSYSU:'||Servers.Lp Open_Fm '(FORCERW '
   'PIPE ',
    '< IP ADDRESS' Open_Fm ,
    '| Strip',
    '| VAR IP_Addr' /*
    '| CONSOLE'        */
    /* OK, now let's get a list of all the subdirs on this server */
   'Pipe',
   '  CMS LISTDIR' Servers.Lp,
   '| Drop 2',
   '| Specs 4.17Ø 1',
   '| Stem Dir_Ids.' /*
   '| Console'      */
   /* Now lets see if I have any files to deal with in this tree */
   Do Lp2 = 1 to Dir_Ids.Ø
      'Pipe CMS ACCESS' Dir_Ids.Lp2 Open_Fm '(FORCERW'
      'Pipe',
      '   CMS Listfile * *' Open_Fm '(DATE NOH',
      '| Stem File_Ids.' /*
      '| Console'          */
      /* Ok, we have a list of the files that exist in this directory,
         let's see if the size/time allows for a download now... */
      Do Lp3 = 1 to File_Ids.Ø
      Parse Var File_Ids.Lp3 M_Fn M_Ft M_Fm M_Fmt M_Lrecl M_Recs M_Blks,
             M_Dt M_Time .
        If M_Fn ¬= 'DMSLSTØØ2E' then do /* No files found by Listfile */
            If M_Blks <= Max_Blocks then do
               /* See if we are in the window for this run still... */
               If Time(m) - Run_Time < Max_Window then do
                  Call Transfer_File M_Fn M_Ft Open_Fm Dir_Ids.Lp2
               End
               Else do
             Say 'File' Dir_ids.Lp2 M_Fn M_Ft 'not moved - Passed time',
                   'limit for this iteration'
               End
            End
            Else do
        Say 'File' Dir_Ids.Lp2 M_Fn M_Ft 'not moved - over block limit',
                  ' -' Max_Blocks
            End
         End
      End
   End

End
'PIPE CMS RELEASE' Open_Fm
Return
```

```
Mail_Error: Procedure Expose Mail_Opts Mail_Fl Error_Log
Parse arg Msg_Num  Description
Say Msg_Num Description
Txt = Date() Time() Msg_Num Description
'Pipe VAR Txt | >>' Error_Log
Subject_Text = 'VMBFTP FTP ERROR' Msg_Num
'PIPE CMS ERASE' Mail_Fl
'PIPE',
    'VAR Description ',
    '| >' Mail_Fl
'FINIS  * NOTEBOOK *'
'SENDFILE' Mail_Fl FTPERR
/* 'MAIL FTPERR ( FILE' Mail_Fl Mail_Opts   */
Return
Transfer_File: Procedure Expose IP_Addr Low_Alpha Up_Alpha,
    LS_Output FTP_Output. Mail_Opts Mail_Fl Error_Log
Parse arg Parm_String
Parse Var Parm_String Fn Ft Fm FilePool ':' Server'.' Dir_Name
Parse Var Ip_Addr IP_Address Remote_User_Name Password System_Type .
File_Id = Fn||'.'||Ft||'.'||Fm
Out_Ft = Right(Date(d),3,'Ø')
File_Out = Fn||'.'||Out_Ft
Valid_Server_type = Ø
Opn_Brkt = X2C(AD)   /* Open square bracket for VMS systems  */
Clo_Brkt = X2C(BD)   /* Close square bracket for VMS systems */
/* Now let's look at the System Type (from IP ADDRESS file) and set up
  the remote directory name in the correct format (Novell, VMS, UNIX)
*/
Select
   When Translate(System_Type) = 'NOVELL' then do
      /* Format of Dir_Id is now '/Volume/Root/Dir_1/Dir_2/' */
      Dir_Id = Translate('.'||Dir_Name||'.','/','.')
      Dir_Id = Translate(Substr(Dir_Id,1,Length(Dir_Id)-1),
                                                    Low_Alpha,Up_Alpha)
      Valid_Server_type = 1
   End /* Novell */
   When Translate(System_Type) = 'VMS' then do
      Parse var Dir_Name VMS_Volume '.' VMS_Dir
      /* Format of Dir_Id is now 'Vol_Id:{Root.sub1.sub2}' (changing to
         square brackets for VMS.                                    */
      Dir_Id = VMS_Volume||':'||Opn_Brkt||VMS_Dir||Clo_Brkt
      Valid_Server_type = 1
   End /* VMS */
   When Translate(System_Type) = 'UNIX' then do
      Dir_Name = Translate(Dir_Name,Low_Alpha,Up_Alpha)
      /* Format of Dir_Id is now '/root/dir_1/dir_2/'   */
      Dir_Id = Translate('.'||Dir_Name||'.','/','.')
      Valid_Server_type = 1
   End /* Unix */
   When Translate(System_Type) = 'VMSFS' then do
      /* Make sure that we have a sub dir - if not put a period at the
         end of the root dir name   */
      If Pos('.',Dir_Name) = Ø then do
         /* We have a root directory so stick a period on it */
         Dir_Name = Dir_Name||'.'
      End
      /* Format of Dir_Id is now 'Filepool:dir.dir'   */
      Dir_Id = Server||':'||Dir_Name
      Valid_Server_type = 1
   End /* VMSFS */
   Otherwise do
      Call Mail_Error VMDFTPØØ9 'System type' System_Type 'for server',
          Server 'is undefined.'
      Valid_Server_type = Ø
   End /* Otherwise */
End /* Select System Type */
If Valid_server_Type then do
   Alt_File_Out = ''
   Remote_File = File_Exists_Remotely(File_Out)
   If Remote_File then do
                     /* We already have a remote file with same ID */
      Remote_Hit = 1
      Do Fn_Id = 1 to 26 while Remote_Hit
         Ltr = Substr(Up_Alpha,Fn_Id,1)
         Alt_File_Out =
Translate(Fn||'.'||Substr(Out_Ft,1,2)||Ltr,Low_Alpha,Up_Alpha)
         Remote_Hit = FTP_Result_Scan('::'||Alt_File_Out||'::')
         Say 'Remote scanning for ' Alt_File_Out '= ' Remote_Hit Time()
      End
      If Remote_Hit then do
                        /* We have too many dup files to live with */
          'RELO' Fn Ft Server'.'Dir_Name 'TO VMBFTP.WORK'
          Call Mail_Error VMDFTPØØ2
                     'Excessive remote duplicates-'Server'.'Dir_Name,
              File_Id 'RELOcated to VMSYSU:VMBFTP.WORK'
      End
      Else do
         File_Out = Alt_File_Out
         Remote_File = Ø
         Call Mail_Error VMDFTPØØ3
                               'File to be renamed on remote system',
             Dir_Id File_Id 'as' File_Out
      End
   End
   If ¬Remote_File then do
      Queue Remote_User_Name Password
      Queue 'CWD' Dir_Id
      Queue 'PUT' File_Id File_Out
      Queue 'QUIT'
      'Pipe ',
         '  CMS FTP' Ip_Address,
         '| Stem FTP_Output.'  ,
         '| CONSOLE'
      /* Inspect for errors */
      Str1 = '::226 Transfer::'
      Str2 = '::5Ø1 Write::'
      Str3 = '::53Ø Login::'
      Str4 = '::Unable to connect::'
      Str5 = '::553_::'
      Str6 = '::55Ø FTP server does not::'
      Str7 = '::25Ø Transfer completed successfully::'
      Successful_Tran = FTP_Result_Scan(Str1)
      Select
         When FTP_Result_Scan(Str1) | FTP_Result_Scan(Str7) then do
            /* Successful transfer */
            'PIPE CMS ERASE' Fn Ft Fm
         End
         When FTP_Result_Scan(Str2) then do
         Call Mail_Error VMDFTPØØ4 'Write error to remote system (5Ø1)',
                 Dir_Id File_Id 'as' File_Out
             xx = Delete_Remote_File(File_Out)
          End
          When FTP_Result_Scan(Str3) then do
         Call Mail_Error VMDFTPØØ5 'Login error to remote system (53Ø)',
                Remote_User_Name
          End
          When FTP_Result_Scan(Str4) then do
         Call Mail_Error VMDFTPØØ6 'Unable to connect to foreign host.',
                 IP_Address
          End
          When FTP_Result_Scan(Str5) then do
          Call Mail_Error VMDFTPØØ8 'Foreign system denied put request',
                 '(553).' IP_Address Dir_Id File_Id
          End
          When FTP_Result_Scan(Str6) then do
            Call Mail_Error VMDFTPØ1Ø 'Foreign VM system does not have',
                 'filepool adminstrator authority (55Ø).' IP_address,
                  Dir_ID File_Id
          End
          Otherwise do
     Call Mail_Error VMDFTPØØ7 'Unknown error' Ip_Address Dir_Id File_Id
          End
       End  /* Select for result scan */
   End  /* no remote file */
End /* Valid server Type */
Return
File_Exists_Remotely:
```

```
                 Procedure Expose Ip_Address Remote_User_Name Password,
    Dir_Id Low_Alpha Up_Alpha LS_Output FTP_Output.
/* This is a function that will go to the remote system at the specified
   directory and look for a file that is already there with the same
   File Id.  If it is found, the function will return the value 1.  If
   the file is not found, it will return a value of Ø.
*/
Parse arg File_Id .
Low_File_Id = Translate(File_Id,Low_Alpha,Up_Alpha)
Queue Remote_User_Name Password
Queue 'CWD' Dir_Id
Queue 'LS'
Queue 'QUIT'
'Pipe ',
   '  CMS FTP' Ip_Address,
   '| Stem FTP_Output.' /*
   '| CONSOLE'             */
/* Inspect output for duplicate file id on the drive - IN LOWER CASE */
Str = '::'||Low_File_Id||'::'
File_Match = FTP_Result_Scan(Str)
If Low_File_Id = Ls_Output then do  /* The file was in the LS */
   Exit_Cd = 1
End
Else Do
   Exit_Cd = Ø
End
Return Exit_Cd
FTP_Result_Scan: Procedure Expose FTP_Output. LS_Output
/* Inspect output for a match with the passed parameter.  If a match
   is found, then send back the balance of the line in field called
   'LS_Output'.
*/
Parse arg . "::" Locate_String "::" .
LS_Output = ''
/*
Say '>>>>>>>>>>>>>>>>>>>>>>>>>>>> FIND STRING 'Locate_String
'<<<<<<<<<<<'
*/
'Pipe',
   '  Stem FTP_Output.',
   '| StrFind /'Locate_String'/',
   '| Var Ls_Output' /*
   '| Console '        */
/*
Say '==========================LS_Output='Ls_Output'=================='
*/
If Ls_Output = 'LS_OUTPUT' then do
   Exit_Cd = Ø /* No match found */
End
```

```
Else Do
   Exit_Cd = 1  /* Match found */
End
Return Exit_Cd
Delete_Remote_File:
                  Procedure Expose Ip_Address Remote_User_Name Password,
    Dir_Id LS_Output FTP_Output.
/* This is a function that will go to the remote system at the specified
   directory and delete a specified file.  This is done as a clean up
   procedure for a failed FTP.  This procedure will return a 1 if it is
   successful in deleting the file.
*/
Parse arg File_Id .
Queue Remote_User_Name Password
Queue 'CWD' Dir_Id
Queue 'DELETE' File_Id
Queue 'QUIT'
'Pipe ',
   '  CMS FTP' Ip_Address,
   '| Stem FTP_Output.'/*
   '| CONSOLE'            */
/* Look for an OK message for the delete now. */
   Str = '::2ØØ OK.::'
   File_Match = FTP_Result_Scan(Str)
Return File_Match
```

Lawrence E Rondot
Systems Programmer
Indiana University Purdue University Fort Wayne (USA)

Renaming files

The CMS RENAME command does not allow you to rename a file where only parts of the filename or filetype are changed for the target file. The following procedure overcomes this limitation.

SYNTAX

```
RENAMCH fn ft fm fnneu ftneu fmneu
```

where 'fn ft fm' are the files to be renamed (wildcards are allowed) and 'fnneu ftneu fmneu' are the new file names. Corresponding parts (that should be identical) can be specified by the ‘=’ sign. The ‘=’ is only valid at the end of a name.

EXAMPLES

```
     RENAMCH MAK* TEST Z MAT= = =
```

renames all files with a filename beginning with ‘MAK’ and a filetype of ‘TEST’ on filemode ‘Z’ to files with a filename where the first three characters are changed to ‘MAT’ and the filetype remains the same.

```
     RENAMCH TEST JOB* X PROD JCL= =
```

renames all files with a filename of ‘TEST’ and a filetype beginning with ‘JOB’ on filemode ‘X’ to files with a filename of ‘PROD’ and a filetype where the first three characters are changed from ‘JOB’ to ‘JCL’.

RENAMCH EXEC

```
/*********************************************************************/
/* RENAME with changing parts of fn or ft                            */
/*********************************************************************/
/* Call:   RENAMCH fn ft fm fnneu ftneu fmneu                        */
/*               fn ft fm            : files to be renamed           */
/*                                   : (wildcards allowed)           */
/*               fnneu ftneu fmneu   : target file names             */
/*                                   :   corresponding parts are     */
/*                                   :   identified by =             */
/*                                   :   (= must be the last charact.*/
/*                                   :    of fnneu ftneu)            */
/*********************************************************************/
/* Examples:  RENAMCH MAK* TEST Z MAT= = =                           */
/*            RENAMCH TEST JOB* X PROD JCL= =                        */
/*********************************************************************/
trace off
parse upper arg fn ft fm fnneu ftneu fmneu .
if fn = '?' then signal hilfe
'MAKEBUF'
'LISTFILE' fn ft fm '(EXEC ARGS'
if rc ¬= Ø then signal fileerr
'EXECIO * DISKR CMS EXEC A (FINIS'
anz = queued()
```

```
do i = 1 to anz
   pull p1 p2 fname ftype fmode p3 p4 pn
   p3neu = p3
   p4neu = p4
   posi = pos('=',fnneu)
   if posi > Ø then do
      if posi ¬= length(fnneu) then signal gleicherr
      if posi = 1 & length(fnneu) ¬= 1 then signal gleicherr
      p3neu = substr(fnneu,1,posi-1) || substr(fname,posi)
   end
   else p3neu = fnneu
   posi = pos('=',ftneu)
   if posi > Ø then do
      if posi ¬= length(ftneu) then signal gleicherr
      if posi = 1 & length(ftneu) ¬= 1 then signal gleicherr
      p4neu = substr(ftneu,1,posi-1) || substr(ftype,posi)
   end
   else p4neu = ftneu
   zeile.i = p1 p2 fname ftype fmode p3neu p4neu pn
end
'ERASE CMS EXEC A'
'EXECIO' anz 'DISKW CMS EXEC A (FINIS STEM ZEILE.'
'EXEC CMS RENAME % % %' fmneu
'DROPBUF'
exit

gleicherr:
say 'Error: Equal sign (=) may be specified only at the end of fn/ft'
'DROPBUF'
exit

fileerr:
say 'Error: No files found'
exit

/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
hilfe:
'VMFCLEAR'
address cms 'type renamch exec * 1 15'
```

Dr Reinhard Meyer (Germany) © Xephon 1997

DASD Space Manager (DSM)

GENERAL DESCRIPTION

DASD space manager (DSM) is designed both for single mainframe installations and for installations with two mainframes that share DASD. DSM is useful for systems programmers in their everyday work.

DSM provides the following possibilities:

- Viewing the allocation map for a selected volume in a single mainframe installation.
- Viewing the allocation map for a selected volume in a two-mainframe installation.
- Viewing an ordered gaps map for all volumes or a selected volume.
- Viewing an ordered overlaps map for all volumes or a selected volume.
- Interactive mini-disk allocation, using two vertically separated logical screens – one for VMUSERS DIRECT and the other for DSM.

The iterative interactive procedure of DASD space allocation with DSM is shown in Figure 1.

The maximum size of VMUSERS DIRECT that may be processed is 25,000 input lines and 3,000 mini-disks descriptions.

DSM is written in Assembler and REXX.

BASIC SOFTWARE FOR DSM EXECUTION

DSM executes in CMS with VM/SP Release 5.

STARTING DSM

DSM EXEC has no parameters. At installations with two mainframes,

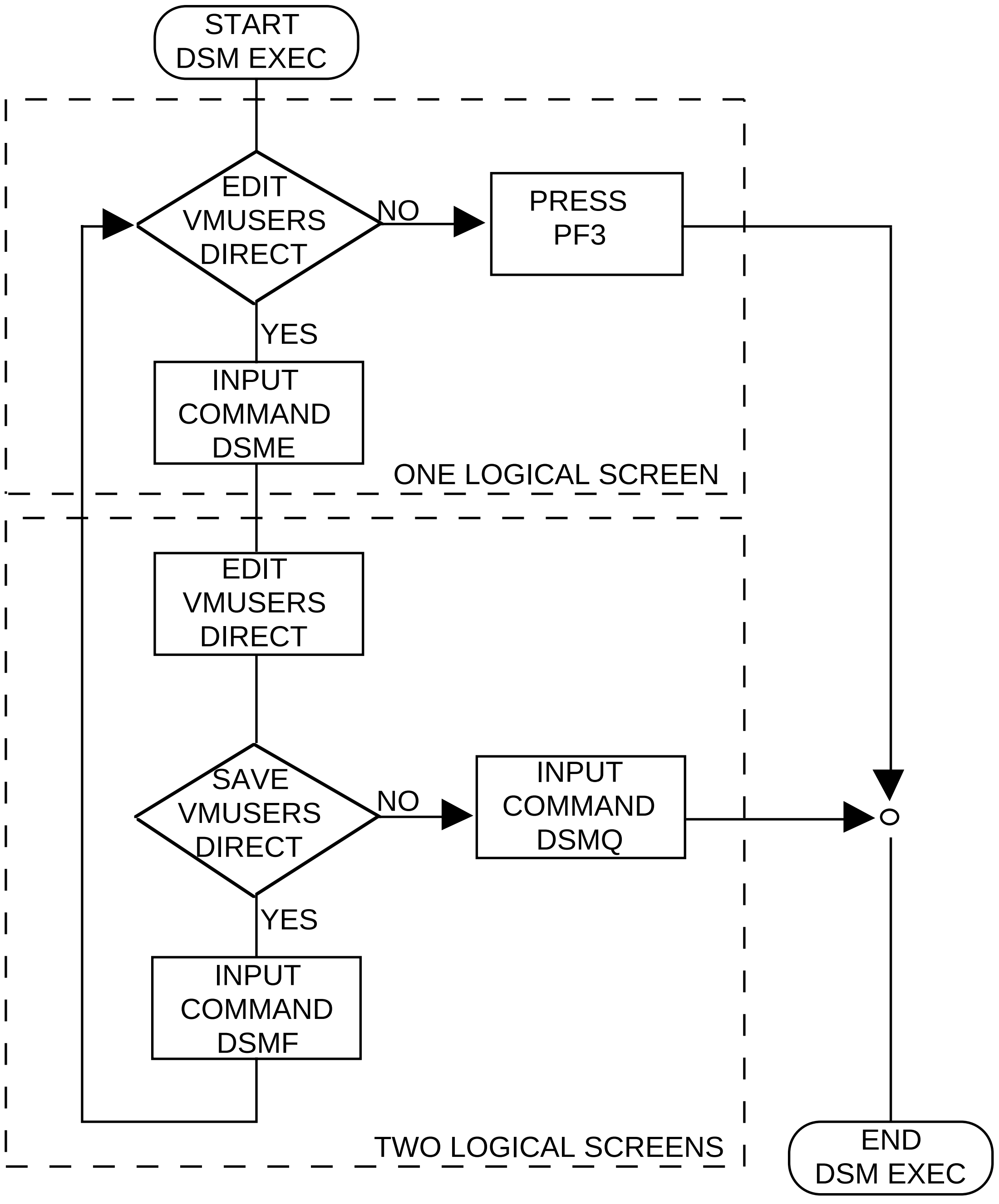

Figure 1: Interactive procedure

simultaneous execution is not allowed. It cannot modify VMUSERS DIRECT on two nodes concurrently.

Before starting DSM, a PROFILE DSM should be created. The file PROFILE DSM contains constant information about a specific installation.

The following record formats are used:

- NODE <nodeid> <mode> [<smode>] (1)
- NODE <nodeid> <smode> (2)
- OWNS <volid> (3)
- <dev type> <max cyl number> (4)

Format (1) defines the node identifier and disk on which VMUSERS DIRECT resides. For two-mainframe installations <smode> defines a shared disk, on which a copy of VMUSERS DIRECT is made, after its modification. The first record always describes a local or single host. Format (2) defines the shared disk in an installation with two mainframes containing VMUSERS DIRECT for the remote host. Format (3) defines non-shared DASD in two-mainframe installations, which may have identical volids. Format (4) determines the maximum cylinder or block numbers for a specific device type.

Examples of file PROFILE DSM in user-id MAINT are:

For a single mainframe installation:

```
NODE NODEONE B
335Ø 554
338Ø 884
```

For a two-mainframe installation:

```
- PROFILE DSM in MAINT at node NODEONE
  NODE NODEONE B C
  NODE NODETWO F
  OWNS VMSRES
  OWNS VMPKØ1
  335Ø 554
  338Ø 884

- PROFILE DSM in MAINT at node NODETWO
  NODE NODETWO B C
  NODE NODEONE F
```

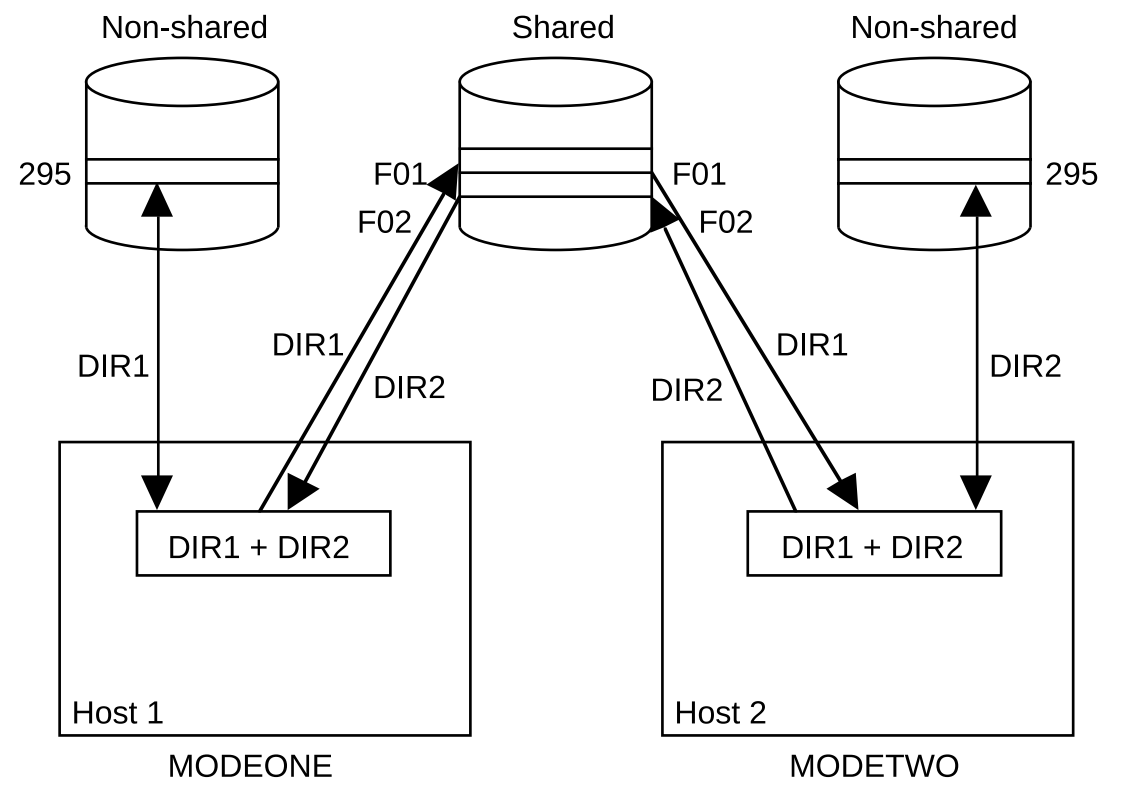

Figure 2: Synchronization

```
OWNS VMSRES
OWNS VMPKØ1
335Ø 554
338Ø 884
```

VMUSERS DIRECT synchronization in installations with two mainframes is shown in Figure 2.

DSM EXECUTION

DSM is PFK and command driven. The commands are XEDIT macros.

PFK settings for volume selection are:

- PF1 – sort by cuu
- PF2 – sort by devtype

- PF3 – exit
- PF4 – begin
- PF5 – end
- PF6 – sort by volume id
- PF7 – backward
- PF8 – forward
- PF9 – view general gaps map
- PF10 – view general overlaps map
- PF11 – view volume allocation map
- PF12 – cursor.

Columns 'Cuu' and 'Devt' contain corresponding allocation values, starting from cylinder or block 0 – usually this is user-id $ALLOC$ with a real cuu and device type.

PF11 gives a view of the allocation map of a volume. It is selected by placing the cursor in any column of the corresponding row on the screen.

Commands that can be used with the volume selection screen are:

- DSME – creates two logical screens vertically separated and reads VMUSERS DIRECT for editing. The second logical screen is for DSM.
- DSMF – issues the XEDIT command FFILE to end VMUSERS DIRECT editing. It is used only in installations with two mainframes. It creates a copy of VMUSERS DIRECT on a shared disk, and calls DSM again to check new DASD space allocation.
- DSMQ – issues the XEDIT command QQUIT to ignore any VMUSERS DIRECT modifications and ends the edit session.

The commands DSMF and DSMQ should be issued in the VMUSERS DIRECT logical screen, and the DMSE command should be entered in the DSM logical screen. Otherwise, the commands are ignored by DSM.

PFK settings for the general gaps map screen are:

- PF1 – sort by volume-id
- PF2 – sort by gaps size
- PF3 – return to volume selection
- PF4 – top
- PF5 – end
- PF7 – backward
- PF8 – forward.

PFK settings for the general overlaps map screen are:

- PF3 – return to volume selection
- PF4 – top
- PF5 – end
- PF7 – backward
- PF8 – forward.

PFK settings for the volume allocation map screen are:

- PF2 – refresh
- PF3 – return to volume selection
- PF4 – top
- PF5 – end
- PF7 – backward
- PF8 – forward
- PF9 – view volume gaps map
- PF10 – view volume overlaps map.

PFK settings for volume gaps map screen and PFK settings for volume overlaps map screen are the same as for the volume allocation map screen.

Free disk space on disk A is used always as a work area for DSM.

When editing large VMUSERS DIRECT files, the DSMF command response time may be slightly increased.

DSM converts a device type FB-512 to a four-byte string F512 and that way displays it on screen.

The following messages may appears in the Note column on DSM screens:

- *DVC* – device type check, unknown device type.
- *GAP* – not allocated, free DASD space.
- *OVL* – two or more mini-disk allocations are overlapping, possible allocation error.

Message *DVC* means, that the record format (4) for this unit in PROFILE DSM is not correct or is not available. In this case the file PROFILE DSM must be corrected, before make any new allocations on the volume.

For installations with two mainframes, if two allocations have identical user-id, cuu, start, and size for a particular shared volume, but the first allocation is made in node 1 and the second allocation is made in node 2, then no message is displayed in the Note column. Instead of the second allocation appearing in the Node 2 column, the corresponding node identifier is displayed. So, if for some allocations both columns Node 1 and Node 2 are not empty, this means that the same allocation was made in both nodes. In this case, the corresponding user-id is not protected from loss of data on the shared disk when it concurrently works in node 1 and node 2 – both having write access to the disk.

DSMINSTL EXEC

```
/******************************************************************/
/***                   - installation aid           ***        ***/
/*** DSMINSTL         dsm install                   ***        ***/
/***                                                ***        ***/
/******************************************************************/
/***   SIZE ØØØ39  VER 1.Ø MOD ØØ                              ***/
/******************************************************************/
  CLRSCRN
  MESSAGE = 'user request'
  SAY ' --- Start DSM 1.Ø installation - reply Y or N'
  PULL REPLY
  IF REPLY ¬= 'Y' THEN
  SIGNAL ERROR
  SET CMSTYPE HT
  SIGNAL ON ERROR
  MESSAGE = ' assemble '$DIRIN
  ASSEMBLE $DIRIN
  ERASE $DIRIN LISTING A
  MESSAGE = ' load     '$DIRIN
  LOAD $DIRIN '(' NOMAP NOLIBE AUTO
  MESSAGE = ' genmod   '$DIRIN
  GENMOD
  ERASE $DIRIN TEXT A
  MESSAGE = ' assemble '$MAPOUT
  ASSEMBLE $MAPOUT
  ERASE $MAPOUT LISTING A
  MESSAGE = ' load     '$MAPOUT
  LOAD $MAPOUT '(' NOMAP NOLIBE AUTO
  MESSAGE = ' genmod   '$MAPOUT
  GENMOD
  ERASE $MAPOUT TEXT A
  SIGNAL OFF ERROR
  SET CMSTYPE RT
  SAY ' --- DSM 1.Ø installed sucsessfully'
  EXIT
ERROR:
  SET CMSTYPE RT
  SAY ' --- DSMINSTL not properly executed -> 'MESSAGE
```

DSM EXEC

```
/*****************************************************************/
/***                                                ***        ***/
/*** DSM          disk space manager                ***        ***/
/***                                                ***        ***/
/*****************************************************************/
/***   SIZE ØØ118  VER 1.Ø MOD Ø2  TIME 13:19:Ø5               ***/
/*****************************************************************/
  SET CMSTYPE HT
  MAKEBUF
  LISTFILE PROFILE DSM '* ( STACK'
  IF RC ¬= Ø THEN
  DO
    SET CMSTYPE RT
    SAY '---' PROFILE DSM 'not found'
    EXIT
  END
  DROPBUF
  SET CMSTYPE RT
  MAKEBUF
  EXECIO '*' DISKR PROFILE DSM '(FINI'
  NODE_1 = ''
  NODE_1S = ''
  NODE_2 = ''
  NODE_2I = ''
  J = Ø
  OWNS = ''
  K = Ø
  UNIT = ''
  DO I = 1 TO QUEUED()
    PULL TAG P
    P = STRIP(P)
    IF TAG = 'NODE' THEN
    DO
      PARSE VAR P ID MODE SMODE
      IF MODE = '' THEN
      DO
        SAY '--- MODE or SMODE not defined'
        DROPBUF
        EXIT
      END
      STATE VMUSERS DIRECT MODE
      IF RC ¬= Ø THEN
      DO
        SAY '--- VMUSERS DIRECT' MODE 'not found'
        DROPBUF
        EXIT
      END
      IF I = 1 THEN
      DO
        NODE_1 = ID MODE
        NODE_1S = SMODE
      END
      ELSE
      DO
        NODE_2 = P
        NODE_2I = MODE
      END
    END
    ELSE
    IF TAG = 'OWNS' THEN
    DO
      K = K + 1
      OWNS = OWNS LEFT(P, 6)
    END
    ELSE
    DO
      J = J + 1
      IF SUBSTR(TAG, 1, 1) = 'F' THEN
      TAG = 'F512'
      UNIT =  UNIT TAG RIGHT(P, 6,'Ø')
    END
  END
  DROPBUF
  IF NODE_2 ¬= '' & NODE_1S = '' THEN
  DO
    SAY '--- SMODE not defined'
    EXIT
  END
  IF J = Ø THEN
  DO
    SAY '--- UNIT info not defined'
    EXIT
  END
  SET CMSTYPE HT
  IF NODE_1S ¬= '' THEN
  DO
    AC FØ1 NODE_1S
    IF RC ¬= Ø THEN
    DO
      SET CMSTYPE RT
      SAY '--- Shared disk FØ1 not available'
      EXIT
    END
  END
  IF NODE_2I ¬= '' THEN
  DO
    AC FØ2 NODE_2I
    IF RC ¬= Ø THEN
    DO
      SET CMSTYPE RT
      SAY '--- Shared disk FØ2 not available'
      EXIT
    END
  END
  GLOBALV SELECT DSM PUT 'NODE_1 NODE_1S NODE_2 OWNS J K UNIT'
  SET CMSTYPE HT
REFRESH:
  END = 'Y'
  GLOBALV SELECT DSM PUT 'END'
  ERASE $$$ $$$ A
  X $$$ $$$ A '(' PROF DSM  WIDTH 7Ø
  EXECIO '*' DISKR $$$ $$$ A '(' FINI M 28 43 FIFO
  X $$ $$ A '(' PROF DSMV
  GLOBALV SELECT DSM GET 'END'
  IF END ¬= 'Y' THEN
  SIGNAL REFRESH
  ERASE $$$ $$$ A
```

DSM XEDIT

```
/**********************************************************************/
/***                                                  ***          ***/
```

```
/*** DSM           disk space manager                 ***         ***/
/***                                                  ***         ***/
/********************************************************************/
/***   SIZE ØØØ27  VER 1   MOD Ø1  TIME 16:56:ØØ                  ***/
/********************************************************************/
  X $$$ $$$ A '('  WIDTH 7Ø
  HI = '1DF8'X
  LO = '1DFØ'X
  ADDRESS CMS
  GLOBALV SELECT DSM GET 'NODE_1 NODE_2 OWNS J K UNIT'
  IF NODE_2 ¬= '' THEN
  PROC = 2
  ELSE
  PROC = 1
  $DIRIN PROC NODE_1 NODE_2 K OWNS
  IF RC = 13 THEN
  SAY HI'    No more memory is available - use DEFINE STORAGE'LO
  ELSE
  DO
    ADDRESS XEDIT DMSXMS 28 34 52 64 19 24
    ADDRESS XEDIT ':1'
    ADDRESS CMS $MAPOUT J UNIT
  END
  ADDRESS XEDIT QQUIT
```

DSMF XEDIT

```
/********************************************************************/
/***                                                  ***         ***/
/*** DSMF          disk space manager                 ***         ***/
/***                                                  ***         ***/
/********************************************************************/
/***   SIZE ØØØ37  VER 1.Ø MOD ØØ  TIME 13:49:19                  ***/
/********************************************************************/
  EXT '/FN'
  IF SUBSTR(FNAME.1, 1, 1) ¬= '$' THEN
  DO
    ADDRESS CMS
    GLOBALV SELECT DSM GET 'NODE_1 NODE_1S'
    PARSE VAR NODE_1 . MODE
    QUERY DISK MODE '(' STACK
    PULL
    PULL . CUU .
    SET CMSTYPE HT
    ACCESS CUU MODE
    ADDRESS XEDIT FFILE
    ADDRESS CMS
    ACCESS CUU MODE'/A'
    IF NODE_1S ¬= '' THEN
    COPYFILE VMUSERS DIRECT MODE '= =' NODE_1S '(OLDDATE REP'
    IF RC ¬= Ø THEN
    DO
      SET CMSTYPE RT
      SAY '--- Copy of VMUSERS DIRECT' NODE_1S 'failed'
      SET CMSTYPE HT
    END
    ELSE
    DO
      END = 'R'
      GLOBALV SELECT DSM PUT 'END'
    END
    ADDRESS XEDIT QQUIT
  END
```

DSMOM XEDIT

```
/**********************************************************************/
/***                                                  ***          ***/
/*** DSMOM        disk space manager                  ***          ***/
/***                                                  ***          ***/
/**********************************************************************/
/***   SIZE ØØØ42  VER 1.Ø MOD ØØ  TIME 12:Ø6:Ø8                   ***/
/**********************************************************************/
  PFØ1 ONLY    NULLKEY
  PFØ2 ONLY    NULLKEY
  PFØ3 ONLY    QQUIT
  PFØ4 ONLY    ':1'
  PFØ5 ONLY    BOT
  PFØ6 ONLY    NULLKEY
  PFØ7 ONLY    '-19'
  PFØ8 ONLY    19
  PFØ9 ONLY    NULLKEY
  PF1Ø ONLY    NULLKEY
  PF11 ONLY    NULLKEY
  PF12 ONLY    NULLKEY
  ENT  ONLY    NULLKEY
  CMDLINE      OFF
  CURLINE      ON  4
  CURS         SCR 4 1
  MSGLINE      OFF
  NUMB         OFF
  PREFIX       NULL
  SCALE        OFF
  SERIAL       OFF
  STAY         ON
  TOFEOF       OFF
  RESER  1 HI COPIES(' ', 48) '*** Disk Space Manager ***'
  RESER  2 HI COPIES(' ', 49) '***  Ver 1.Ø (C) DG''95 ***'
  RESER 24 NO '   3 Exit  4 Top  5 End  7 +  8 -'
  DO I = 1 TO QUEUED() BY 2
    PULL LINE
    INPUT LINE
    PULL
 END
 RESER 3 HI COPIES(' ', 5) LEFT('Node 1', 8) LEFT('Node 2', 8)       ,
            'Userid   Volume  Cuu Devt Start  End    Size   Note'
 ':1'
```

DSMDM XEDIT

```
/*********************************************************************/
/***                                                  ***          ***/
/*** DSMDM        disk space manager                  ***          ***/
/***                                                  ***          ***/
/*********************************************************************/
/***   SIZE ØØØ44  VER 1.Ø MOD Ø1  TIME 16:55:41                   ***/
/*********************************************************************/
  PFØ1 ONLY    MACRO DSMED
  PFØ2 ONLY    MACRO DSMR
  PFØ3 ONLY    QQUIT
  PFØ4 ONLY    ':1'
  PFØ5 ONLY    BOT
  PFØ7 ONLY    '-19'
  PFØ8 ONLY    19
  PFØ9 ONLY    ALL '/GAP'
  PF1Ø ONLY    ALL '/OVL'
  PF11 ONLY    NULLKEY
  PF12 ONLY    NULLKEY
  ENT  ONLY    NULLKEY
  CMDLINE      OFF
  CURLINE      ON  4
  CURS         SCR 4 1
  MSGLINE      OFF
  MSGMODE      OFF
  NUMB         OFF
  PREFIX       NULL
  SCALE        OFF
  SERIAL       OFF
  SHADOW       OFF
  STAY         ON
  TOFEOF       OFF
  RESER  1 HI COPIES(' ', 48) '*** Disk Space Manager ***'
  RESER  2 HI COPIES(' ', 49) '***  Ver 1.Ø (C) DG''95 ***'
  RESER 24 NO '   2 Refr  3 Exit  4 Top  5 End  7 +  8 - ' ,
         '9 Gaps  1Ø Ovrs'
  DO I = 1 TO QUEUED() BY 2
    PULL LINE
    INPUT LINE
```

```
   PULL
 END
 RESER 3 HI COPIES(' ', 5) LEFT('Node 1', 8) LEFT('Node 2', 8)       ,
           'Userid   Volume  Cuu Devt Start  End    Size   Note'
 ':1'
```

DSMV XEDIT

```
/*******************************************************************/
/***                                                  ***      ***/
/*** DSMV         disk space manager                  ***      ***/
/***                                                  ***      ***/
/*******************************************************************/
/***   SIZE 00045  VER 1.0 MOD 02  TIME 13:32:16                ***/
/*******************************************************************/
  PF01 ONLY    DMSXMS 8 10
  PF02 ONLY    DMSXMS 12 15 8 10
  PF03 ONLY    QQUIT
  PF04 ONLY    ':1'
  PF05 ONLY    BOT
  PF06 ONLY    DMSXMS 1 6
  PF07 ONLY    '-17'
  PF08 ONLY    17
  PF09 ONLY    MACRO DSMG
  PF10 ONLY    MACRO DSMO
  PF11 ONLY    MACRO DSMD
  PF12 ONLY    CURS SCR 4 1
  CMDLINE      BOTTOM
  CURLINE      ON  3
  CURS         SCR 4 1
  MSGLINE      OFF
  NUMB         OFF
  PREFIX       NULL
  SCALE        OFF
  SERIAL       OFF
  STAY         ON
  TOFEOF       OFF
  RESER  1 HI COPIES(' ', 48) '*** Disk Space Manager ***'
  RESER  2 HI COPIES(' ', 49) '***  Ver 1.0 (C) DG''95 ***'
  RESER  3 HI COPIES(' ', 6)'Volume  Cuu Devt'
  RESER 22 NO '1 S(C) 2 S(D) 3 Exit 4 Bgn 5 End 6 S(V)'          ,
              '7 + 8 - 9 Gaps 10 Ovrs 11 Map 12 Curs'
  RESER 23 NO 'Cmd DSMF DSMQ DSME'
  BUF = ''
  DO I = 1 TO QUEUED()
    PULL NEW
    IF SUBSTR(BUF, 1, 7) = SUBSTR(NEW, 1, 7) THEN
    ITERATE
    BUF = NEW
```

```
     INPUT BUF
  END
  ':1'
```

DSME XEDIT

```
/*******************************************************************/
/***                                                 ***        ***/
/*** DSME           disk space manager               ***        ***/
/***                                                 ***        ***/
/*******************************************************************/
/***   SIZE ØØØ2Ø  VER 1.Ø MOD ØØ  TIME 13:Ø3:Ø5                ***/
/*******************************************************************/
  EXT '/FN/SCR'
  IF SUBSTR(FNAME.1, 1, 1) = '$' THEN
  DO
    IF SUBSTR(SCREEN.1, 1, 1) = 'S' THEN
    DO
      PREF OFF
      SCR W 6Ø 2Ø
      ADDRESS CMS GLOBALV SELECT DSM GET 'NODE_1'
      PARSE VAR NODE_1 . MODE
      X VMUSERS DIRECT MODE
    END
  END
```

DSMO XEDIT

```
/*******************************************************************/
/***                                                 ***        ***/
/*** DSMO          disk space manager                ***        ***/
/***                                                 ***        ***/
/*******************************************************************/
/***   SIZE ØØØ17  VER 1.Ø MOD ØØ  TIME 12:14:5Ø                ***/
/*******************************************************************/
  ADDRESS CMS
  DO FOREVER
    EXECIO '*' DISKR $$$ $$$ A '(LO /OVL/ Z 67 69 FIFO'
    IF RC ¬= Ø THEN
    LEAVE
  END
  FINIS $$$ $$$ A
  XEDIT $ $ A '(' PROF DSMOM
  ADDRESS XEDIT SOS PF12
```

DSMG XEDIT

```
/*******************************************************************/
/***                                                 ***        ***/
/*** DSMG          disk space manager                ***        ***/
/***                                                 ***        ***/
/*******************************************************************/
/***   SIZE ØØØ16  VER 1.Ø MOD ØØ  TIME 11:54:21                ***/
/*******************************************************************/
  ADDRESS CMS
  DO FOREVER
    EXECIO '*' DISKR $$$ $$$ A '(LO /GAP/ Z 67 69 FIFO'
    IF RC ¬= Ø THEN
    LEAVE
  END
  FINIS $$$ $$$ A
  X $ $ A '(' PROF DSMGM
```

DSMD XEDIT

```
/*******************************************************************/
/***                                                 ***        ***/
/*** DSM           disk space manager                ***        ***/
/***                                                 ***        ***/
/*******************************************************************/
/***   SIZE ØØØ28  VER 1.Ø MOD Ø1  TIME 16:55:48                ***/
/*******************************************************************/
  EXT '/CURS/LI'
  IF CURSOR.3 <= Ø THEN
  EXIT
  PF12 CURS SCR CURSOR.1 1
  ':'CURSOR.3
  STACK 1 1 8
  PULL VOLUME .
  CL ':18'
  CR CENTRE('Peeked at' TIME(), 56, '-')
  ':'LINE.1
  ADDRESS CMS
  VOLUME = LEFT(STRIP(VOLUME), 7)
  DO FOREVER
    EXECIO '*' DISKR $$$ $$$ A '(LO /'VOLUME'/ Z 28 34 FIFO'
    IF RC ¬= Ø THEN
    LEAVE
  END
  FINIS $$$ $$$ A
  XEDIT $ $ A '(' PROF DSMDM
  ADDRESS XEDIT SOS PF12
```

DSMGM XEDIT

```
/*******************************************************************/
/***                                                  ***      ***/
/*** DSMGM        disk space manager                  ***      ***/
/***                                                  ***      ***/
/*******************************************************************/
/***   SIZE ØØØ44  VER 1.Ø MOD ØØ  TIME 12:Ø6:Ø8               ***/
/*******************************************************************/
  PFØ1 ONLY    DMSXMS 28 33
  PFØ2 ONLY    DMSXMS D 58 63
  PFØ3 ONLY    QQUIT
  PFØ4 ONLY    ':1'
  PFØ5 ONLY    BOT
  PFØ6 ONLY    NULLKEY
  PFØ7 ONLY    '-19'
  PFØ8 ONLY    19
  PFØ9 ONLY    NULLKEY
  PF1Ø ONLY    NULLKEY
  PF11 ONLY    NULLKEY
  PF12 ONLY    NULLKEY
  ENT  ONLY    NULLKEY
  CMDLINE      OFF
  CURLINE      ON  4
  CURS         SCR 4 1
  MSGLINE      OFF
  MSGMODE      OFF
  NUMB         OFF
  PREFIX       NULL
  SCALE        OFF
  SERIAL       OFF
  STAY         ON
  TOFEOF       OFF
  RESER  1 HI COPIES(' ', 48) '*** Disk Space Manager ***'
  RESER  2 HI COPIES(' ', 49) '***  Ver 1.Ø (C) DG''95 ***'
  RESER  3 HI COPIES(' ', 5) LEFT('Node 1', 8) LEFT('Node 2', 8)     ,
           'Userid   Volume  Cuu Devt Start  End    Size   Note'
  RESER 24 NO                                                      ,
  '   1 S(Vol)  2 S(Size)  3 Exit  4 Top  5 End  7 +  8 -'
  DO I = 1 TO QUEUED() BY 2
    PULL LINE
    INPUT LINE
    PULL
  END
  ':1'
```

DSMQ XEDIT

```
/*******************************************************************/
/***                                                  ***      ***/
```

```
/*** DSMQ          disk space manager                 ***         ***/
/***                                                  ***         ***/
/*****************************************************************/
/***   SIZE ØØØ19  VER 1.Ø MOD ØØ  TIME 13:41:39                  ***/
/*****************************************************************/
  EXT '/FN'
  IF SUBSTR(FNAME.1, 1, 1) = '$' THEN
  DO
    QQUIT
    QQUIT
  END
  ELSE
  DO
    QQUIT
    SCR 1
  END
```

DSMR XEDIT

```
/*****************************************************************/
/***                                                  ***         ***/
/*** DSM           disk space manager                 ***         ***/
/***                                                  ***         ***/
/*****************************************************************/
/***   SIZE ØØØ1Ø  VER 1.Ø MOD ØØ  TIME 12:59:37                  ***/
/*****************************************************************/
  ALL
  ':1'
```

$DIRIN ASSEMBLE

```
******************************************************************
****                                                  ***        ****
**** DSM          disk space manager                  ***        ****
****                                                  ***        ****
******************************************************************
****   SIZE ØØ234  VER 1.Ø MOD Ø1  TIME 14:54:36                  ****
******************************************************************
*                                                                *
$DIRIN   CSECT
         PRINT NOGEN
         USING *,12
         ST    14,RG14
         MVC   TOPROC(1),8(1)
         MVC   NODE1(8),16(1)
         MVC   MODE1(1),24(1)
         CLI   8(1),C'2'
         BNE   READ
         MVC   NODE2S(8),32(1)
         MVC   MODE2(1),4Ø(1)
         PACK  DOUBLE(8),48(1,1)
         MVN   DOUBLE+7(1),=X'ØC'
         CVB   11,DOUBLE
         LTR   11,11
         BZ    READ
         LR    3,11
         LA    1,56(1)
         LA    2,VOLIDS
INVOL    EQU   *
         MVC   Ø(6,2),Ø(1)
         LA    1,8(1)
         LA    2,6(2)
         BCT   3,INVOL
READ     EQU   *
         DMSFREE DWORDS=25ØØØØ,ERR=RET,AREA=HIGH
         ST    1,RG1
         LR    5,1
         LA    2,DIRECT
         FSREAD (2),NOREC=25ØØØ,BSIZE=2ØØØØØØ,BUFFER=(5)
         LTR   15,15
         BNZ   ABEND
         ST    Ø,RGØ
         CLI   TOPROC,C'1'
         BE    ONLYONE
         LR    6,5
         AR    6,Ø
         MVC   MODE1(1),MODE2
         S     Ø,=F'2ØØØØØØ'
         BZ    ABEND
         LCR   7,Ø
         FSCLOSE (2)
         FSREAD (2),NOREC=25ØØØ,BSIZE=(7),BUFFER=(6)
         LTR   15,15
         BNZ   ABEND
         A     Ø,RGØ
ONLYONE  EQU   *
         BCTR  Ø,Ø
         LR    6,5
         A     6,RGØ
         ST    6,RGØ
         LR    6,5
         LA    2,8Ø
         LR    3,5
         AR    3,Ø
         LA    8,1
         FSCLOSE (2)
CYC      EQU   *
         C     6,RGØ
         BNE   CONTINUE
         MVC   NODE1(8),=8X'4Ø'
         MVC   NODE2(8),NODE2S
CONTINUE EQU   *
         CLI   Ø(6),C'*'
         BE    MISS
         LR    7,6
         BAL   14,SELWORD
         CLC   Ø(4,7),=C'USER'
         BNE   CHECKUSR
         LA    7,1(1,7)
         BAL   14,SELWORD
         MVC   USER+1(7),=7X'4Ø'
         LA    1Ø,USER
         EX    1,MVC
         B     MISS
CHECKUSR EQU   *
         CLI   USER,C'$'
         BNE   CHECKMDK
         CLC   USER(7),=C'$ALLOC$'
         BNE   MISS
CHECKMDK EQU   *
         CLC   Ø(5,7),=C'MDISK'
         BNE   MISS
         LA    7,1(1,7)
         BAL   14,SELWORD
         LA    1Ø,CUU
         EX    1,MVC
         LA    7,1(1,7)
         BAL   14,SELWORD
         CLI   Ø(7),C'F'
         BNE   NOTFB
         MVC   DEV(4),=C'F512'
         B     JUMPDEV
NOTFB    EQU   *
         LA    1Ø,DEV
         EX    1,MVC
JUMPDEV  EQU   *
         LA    7,1(1,7)
         BAL   14,SELWORD
         CLI   Ø(7),C'T'
         BE    MISS
         LA    1Ø,PSTART
         EX    1,PACK
         LA    7,1(1,7)
         BAL   14,SELWORD
         LA    1Ø,PCYLS
         EX    1,PACK
         LA    7,1(1,7)
         BAL   14,SELWORD
         MVI   VOL+6,X'4Ø'
         LTR   11,11
         BZ    MOVEID
         LR    14,11
         LA    15,VOLIDS
CHKID    EQU   *
         CLC   Ø(6,15),Ø(7)
         BE    MODID
         LA    15,6(15)
         BCT   14,CHKID
         B     MOVEID
MODID    EQU   *
         LA    1,1(1)
         CLI   NODE1,X'4Ø'
         BE    ADD2
         MVI   6(7),X'F1'
         B     MOVEID
ADD2     EQU   *
         MVI   6(7),X'F2'
MOVEID   EQU   *
         LA    1Ø,VOL
         EX    1,MVC
         UNPK  START(6),PSTART
         OI    START+5,X'FØ'
         UNPK  CYLS(6),PCYLS
         OI    CYLS+5,X'FØ'
         AP    PCYLS,PSTART
         SP    PCYLS,=P'1'
         UNPK  END(6),PCYLS
         OI    END+5,X'FØ'
         LA    Ø,EXTPLIST
         LA    1,FSCB
         ICM   1,8,=X'Ø2'
         SVC   2Ø2
         DC    AL4(1)
         C     15,=F'13'
         BE    FREEMEM
MISS     EQU   *
         BXLE  6,2,CYC
FREEMEM  EQU   *
         L     1,RG1
         DMSFRET DWORDS=2ØØØØØ,LOC=(1)
RET      EQU   *
         L     14,RG14
         BR    14
ABEND    EQU   *
         ABEND 1313
TRT      TRT   Ø(Ø,7),TRTTABLE
MVC      MVC   Ø(Ø,1Ø),Ø(7)
PACK     PACK  Ø(4,1Ø),Ø(Ø,7)
```

```
SELWORD  EQU   *
         LA    9,1Ø(7)
CHECKBGN EQU   *
         CLI   Ø(7),X'4Ø'
         BNE   THISBGN
         BXLE  7,8,CHECKBGN
THISBGN  EQU   *
         SR    1,1
         LA    15,9
         EX    15,TRT
         LA    2,8Ø
         LTR   1,1
         BZ    MISS
         SR    1,7
         BCTR  1,Ø
         BR    14
TRTTABLE EQU   *
         DC    64X'ØØ'
         DC    X'4Ø'
         DC    191X'ØØ'
RGØ      DS    F
RG1      DS    F
RG14     DS    F
DOUBLE   DS    D
EXTPLIST EQU   *
         DC    A(COMMVERB)
         DC    A(Ø)
         DC    A(Ø)
         DC    A(Ø)
COMMVERB DC    CL8'SUBCOM'
FSCB     EQU   *
FSCBCOMM DC    CL8'DMSXFLWR'
FSCBFN   DC    CL8'$$$'
FSCBFT   DC    CL8'$$$'
FSCBFM   DC    CL2'A'
FSCBITNO DC    H'Ø'
FSCBBUFF DC    A(BUF)
FSCBSIZE DC    F'7Ø'
FSCBFV   DC    CL1'F'
FSCBFLG  DC    X'ØØ'
FSCBNOIT DC    H'1'
FSCBNORD DC    F'Ø'
FSCBAITN DC    F'Ø'
FSCBANIT DC    F'1'
FSCBWPTR DC    A(Ø)
FSCBRPTR DC    A(Ø)
DIRECT   DC    CL8'VMUSERS'
         DC    CL8'DIRECT'
MODE1    DC    CL2' '
TOPROC   DS    CL1
```

```
NODE2S   DS    CL8
MODE2    DS    CL1
VOLIDS   DS    9CL6
BUF      EQU   *
NODE1    DC    CL9' '
NODE2    DC    CL9' '
USER     DC    CL9' '
VOL      DC    CL8' '
CUU      DC    CL4' '
DEV      DC    CL5' '
START    DC    CL7' '
END      DC    CL7' '
CYLS     DC    CL7' '
         DC    CL2Ø' '
PSTART   DS    PL4
PCYLS    DS    PL4
         END   $DIRIN
```

$MAPOUT ASSEMBLE

```
**********************************************************************
****                                                  ***        ****
**** DSM          disk space manager                  ***        ****
****                                                  ***        ****
**********************************************************************
****   SIZE ØØ163  VER 1.Ø MOD Ø1  TIME 15:26:39                 ****
**********************************************************************
*                                                                    *
$MAPOUT  CSECT
         USING *,12
         LR    11,14
         MVC   UNITS(1),8(1)
         PACK  DOUBLE(8),UNITS(1)
         MVN   DOUBLE+7(1),=X'ØC'
         CVB   Ø,DOUBLE
         LR    1Ø,Ø
         LA    4,16
         LA    3,UNITDATA
CYC      EQU   *
         LA    2,Ø(4,1)
         MVC   Ø(4,3),Ø(2)
         MVC   4(6,3),8(2)
         LA    3,1Ø(3)
         LA    4,16(4)
         BCT   Ø,CYC
         DMSFREE DWORDS=2625Ø,ERR=RET
         ST    1,FSCBBUFF
         LA    Ø,EXTPLIST
         LA    1,FSCB
         ICM   1,8,=X'Ø2'
         SVC   2Ø2
         DC    AL4(1)
         L     2,FSCBBUFF
         L     Ø,FSCBNORD
         LR    1,2
         AR    1,Ø
         MVC   27(6,1),=6X'FF'
         SRDL  Ø,32
         D     Ø,=F'7Ø'
         LR    3,1
         MVC   BUF(7Ø),Ø(2)
         ZAP   TEND(4),=P'-1'
CHECK    EQU   *
         PACK  START(4),BUF+44(6)
         MVN   START+3(1),=X'ØC'
         ZAP   END,TEND
         PACK  TEND(4),BUF+51(6)
         MVN   TEND+3(1),=X'ØC'
         SP    START,END
         CP    END(4),=P'-1'
         BNE   CHECKNXT
         CP    START,=P'2'
         BL    CHECKEND
         B     SETGAP
CHECKNXT EQU   *
         CP    START(4),=P'1'
         BE    CHECKEND
         BL    OVERLAP
SETGAP   EQU   *
         BAL   9,WRITEGAP
         B     CHECKEND
OVERLAP  EQU   *
         MVC   BUF+65(5),=C'*OVL*'
CHECKEND EQU   *
         LR    4,1Ø
         LA    5,UNITDATA
CHECKUNT EQU   *
         CLC   Ø(4,5),BUF+39
         BE    FINDUNIT
         LA    5,1Ø(5)
         BCT   4,CHECKUNT
         MVC   BUF+65(5),=C'*DVC*'
         SR    5,5
         B     WRITE
FINDUNIT EQU   *
         CLC   BUF+51(6),4(5)
         BNH   WRITE
         MVC   BUF+65(5),=C'*ERR*'
WRITE    EQU   *
```

```
         LA    2,7Ø(2)
         CLC   BUF+27(7),27(2)
         BNE   CONTINUE
         CLI   BUF,X'4Ø'
         BE    CONTINUE
         CLI   9(2),X'4Ø'
         BE    CONTINUE
         CLC   18(8,2),BUF+18
         BNE   CONTINUE
         CLC   44(8,2),BUF+44
         BNE   CONTINUE
         CLC   58(8,2),BUF+58
         BNE   CONTINUE
         MVC   BUF+9(8),9(2)
         LA    2,7Ø(2)
         BCTR  3,Ø
CONTINUE EQU   *
         FSWRITE '$$$ $$$ A',BUFFER=BUF,BSIZE=7Ø
         CLC   BUF+27(7),27(2)
         BE    MISS
         LTR   5,5
         BZ    NOGAP
         PACK  START(4),4(6,5)
         MVN   START+3(1),=X'ØC'
         SP    START,TEND
         BZ    NOGAP
ISGAP    EQU   *
         AP    START,=P'1'
         ZAP   END,TEND
         BAL   9,WRITEGAP
NOGAP    EQU   *
         ZAP   TEND(4),=P'-1'
MISS     EQU   *
         MVC   BUF(7Ø),Ø(2)
         BCT   3,CHECK
         FSCLOSE '$$$ $$$ A'
         L     1,FSCBBUFF
         DMSFRET DWORDS=2625Ø,LOC=(1)
RET      EQU   *
         BR    11
WRITEGAP EQU   *
         AP    END,=P'1'
         AP    START,=P'-1'
         MVC   GAPMSG+27(7),BUF+27
         UNPK  GAPMSG+44(6),END(4)
         OI    GAPMSG+49,X'FØ'
         UNPK  GAPMSG+58(6),START(4)
         OI    GAPMSG+63,X'FØ'
         FSWRITE '$$$ $$$ A',BUFFER=GAPMSG,BSIZE=7Ø
         BR    9
```

```
UNITS    DS    CL1
UNITDATA DS    CL9Ø
BUF      DS    CL7Ø
GAPMSG   DC    CL65' '
         DC    CL5'*GAP*'
START    DS    CL4
END      DS    CL4
TEND     DS    CL4
GAP      DS    CL4
DOUBLE   DS    D
EXTPLIST EQU   *
         DC    A(COMMVERB)
         DC    A(Ø)
         DC    A(Ø)
         DC    A(Ø)
COMMVERB DC    CL8'SUBCOM'
         DS    ØF
FSCB     EQU   *
FSCBCOMM DC    CL8'DMSXFLRD'
FSCBFN   DC    CL8'$$$'
FSCBFT   DC    CL8'$$$'
FSCBFM   DC    CL2'A'
FSCBITNO DC    H'Ø'
FSCBBUFF DC    A(Ø)
FSCBSIZE DC    F'21ØØØØ'
FSCBFV   DC    CL1'F'
FSCBFLG  DC    X'2Ø'
FSCBNOIT DC    H'Ø'  '
FSCBNORD DC    AL4(Ø)
FSCBAITN DC    AL4(Ø)
FSCBANIT DC    AL4(3ØØØ)
FSCBWPTR DC    A(Ø)
FSCBRPTR DC    A(Ø)
         END   $MAPOUT
```

PREPARING FOR DSM

The DSMINSTL EXEC should be used to install DSM.

DSMINSTL EXEC performs the following actions:

- ASSEMBLE $DIRIN and $MAPOUT
- LOAD $DIRIN and GENMOD
- LOAD $MAPOUT and GENMOD.

All DSM REXX texts should be on accessible mini-disks. For installations with two mainframes, mini-disks should be defined on shared DASD to synchronize VMUSERS DIRECTs in both nodes. Actual copies of VMUSERS DIRECTs are written by DSM on these mini-disks.

The following are sample mini-disk control statements for user-id MAINT in two nodes:

```
- node NODEONE -

  MDISK FØ1 338Ø 1ØØ ØØ2 SHRVOL W
  MDISK FØ2 338Ø 1Ø2 ØØ2 SHRVOL R

- node NODETWO -

  MDISK FØ1 338Ø 1Ø2 ØØ2 SHRVOL W
  MDISK FØ2 338Ø 1ØØ ØØ2 SHRVOL R
```

The virtual device addresses F01 and F02 should not be changed when sample mini-disks control statements are modified at your installation. Before first starting DSM, VMUSERS DIRECTs should be copied to the corresponding NODEONE and NODETWO disks.

Dobrin Goranov
Systems Programmer
Information Services (Bulgaria)

VMFE2E – revisited

Playing with the VMFE2E module (see *VM Update*, issue 120, August 1996, page 47) I've seen another restriction to VMFE2E.

Assume the following EXECs:

MYCALLER EXEC

```
/* */
ADDRESS 'COMMAND';
Y. = ' ';
X.Ø = 1;
X.1 = '111';
```

```
SAY 'X.1'  SYMBOL('X.1');
SAY 'X.2'  SYMBOL('X.2');
SAY 'Y.1'  SYMBOL('Y.1');
SAY 'Z.1'  SYMBOL('Z.1');
SAY 'TEST' SYMBOL('TEST');
SAY;
'EXEC MYSUBEX1';
EXIT Ø;
```

MYSUBEX1 EXEC

```
/* */
ADDRESS 'COMMAND';
'VMFE2E GET X. Y. Z. TEST';
SAY 'X.1'  SYMBOL('X.1');
SAY 'X.2'  SYMBOL('X.2');
SAY 'Y.1'  SYMBOL('Y.1');
SAY 'Z.1'  SYMBOL('Z.1');
SAY 'TEST' SYMBOL('TEST');
EXIT Ø;
```

What you should get is:

```
X.1 VAR
X.2 LIT
Y.1 VAR
Z.1 LIT
TEST LIT
```

But you get the following results from MYSUBEX1:

```
X.1 VAR
X.2 LIT
Y.1 VAR
Z.1 LIT
TEST VAR      ==> wrong
```

Compound variables seem to be handled correctly by VMFE2E, but there seems to be a problem with simple variables.

We are running VM/ESA Release 2, service level 9403 and our VMFE2E module is dated 10/02/92 9:56. Maybe this has been resolved in VM/ESA 2.1?

Thomas Rupp
Senior Systems Programmer (Austria) © Xephon 1997

Dynamic menu system for CMS

PROBLEM

We have a FOCUS application that runs on different databases, for different users. What we do now is duplicate the application procedures, and change the statements pointing to the data. (Tell me about an administration nightmare!)

We have several end users that need a specific application. This means that we must duplicate the VM definitions for each user who needs this application.

We also have end users who need different applications. For these we must 'load' their VM with all that is needed for all applications, or set up as many VM users as applications they need.

When we are done with this mess, we have to cope with security, both on access and while running the different applications.

For these reasons (and more), I have designed this simple yet quite effective dynamic menu system.

The idea is to create the procedures 'on the fly', while the user needs it. And it works for any application, not only FOCUS applications.

This is done by 'plugging in' the security part of the menu system to the CMS SYSTEM SYSPROF EXEC. From here, the end user goes to the menu part, the starting point of all productive work. In cases where security isn't needed, you may plug the menu system directly into SYSPROF EXEC.

The dynamic menu system is made of three independent parts:

- The security part
- The dynamic menus part
- The on-line administration part.

THE SECURITY PART

Security is implemented using two REXX programs and two data files. The first program, SERVPASW, runs in a disconnected service machine. It reads passwords from the data file LOG1 XPASW, and gets the access requests from the second program, LOGO EXEC.

Access is granted only if the name and password from the LOGO screen are matched in LOG1 XPASW. Any user can change his access password by typing a name, the matching password, and a new password in the relevent fields. Adding new names and passwords is done by typing in the new entries in the MENU NAME and PASSWORD field, together with the ADDPSWD in the NEW PASSWORD field. The ADDPSWD is set in the second line of the SERVPASW EXEC. Current ADDPSWD is “12345678”.

LOGO EXEC may be called by any user’s PROFILE EXEC, or, if you want to tighten your security, you may plug it in the S disk’s SYSPROF EXEC. In this case, users’ PROFILE EXECs are bypassed, and any attempt to get through by issuing IPL CMS with NOPROF will put the user back to the security part.

SERVPASW EXEC in the password checking service machine and LOGO EXEC in the user’s VM communicate in both directions using SMSG commands. If the service machine decides to grant access to the dynamic menus to a user, an authorization file, MENU XAUTH, is checked for the existence of one or more menus for this user. Then the menu program is called with the matching menu name. If security is not needed, you may call the menu program directly from the user’s PROFILE EXEC or the system’s SYSPROF EXEC.

THE DYNAMIC MENUS PART

I call ‘MENU’ a bunch of functions related to the same kind of application. Menus are defined in MENU XLINES file. Each menu has a name, a short description, and a set of ‘menu lines’. Menu lines are the execution definitions of the function we want to operate from our menu. A menu line has a name, a description, and a type. The type may be ‘EXEC’ (if it executes a CMS EXEC procedure), it may be ‘MENU’ (if it calls another dynamic menu), or ‘MENPROC’ (if it toggles the auto-generated procedure feature).The ‘REXX’ type has been added to allow the execution of REXX statements.

The second data file is the authorization file MENU XAUTH. In this file, there is an entry for each user's authorized menu. In front of each user/menu key, there is a 'stream' of flags !X!; each flag represents the user authorization to access the specific 'menu line' of this menu.The N position of a flag in the stream means that the Nth line of the menu is to be present in this user's menu.

The third data file, MENU XSTATMS, deals with a special type of menu line – the 'MENPROC' type. This type of menu lines operates an auto-generated REXX procedure. Each 'menproc' definition includes LINK, ACCESS, TDISK, and EXEC statements. LINK and ACCESS statements are CP/CMS-like. LINK passwords are slightly scrambled using REXX X2C and REVERSE functions. TDISK parameters may be 'Y' (yes, default size), 'N' (no), or Number of cylinders for temporary disk size. The EX parameter is the name of a FOCUS program to execute once the proper setup is done. When a MENPROC menu option is selected from the menu, the MAKEXEC program is initiated. It reads the proper MENPROC definitions from the MENU XSTATMS data file, and builds a REXX procedure. This procedure is run immediately, and is discarded as soon as it completes.

Samples of MENU data files are enclosed. You may add your own definitions simply by editing the samples.

THE ON-LINE ADMINISTRATION PART

One can update the data files manually, using XEDIT, or use the full screen utilities.

The on-line administration utilities would be used to:

1 Administer passwords

2 Create menus

3 Update menus

4 Delete menus

5 Update authorizations

6 Create/update MENPROCs

7 Delete MENPROCs

THE DYNAMIC MENUS SETUP

A VM user-id is needed to run the Menus password verification. This can be a regular CMS user, which can issue CP SMSG commands. It needs to have read/write access to the passwords file (LOGO XPASW) and read access to the authorization file (MENU XAUTH). The password checking server must run SERVPASW EXEC.

The dynamic menu system entry point is to be plugged in the user’s PROFILE EXEC, or in the CMS SYSTEM SYSPROF EXEC. This entry point is LOGO EXEC if you choose to use the password verification, or MENU EXEC if the dynamic menu system is all you need.

To use MENU EXEC, you need to have read access to MENU XAUTH, MENU XLINES, and MENU XSTATMS. I strongly recommend not to put these files on the same mini-disk as the password file.

The on-line administration utilities need read/write access to all data files.

SERVPASW

```
/* PASSWORDS SERVER MONITOR PROGRAM */
ADDPSWD='12345678'                 /* THIS IS THE PASSWORD TO TYPE IN  */
                                   /* "NEW PASSWORD" FIELD IN ORDER TO */
                                   /* ADD A NEW MENUID AND PASSWORD IN */
                                   /* LOG1 XPASW FILE                  */
'VMFCLEAR'
SAY DATE(J)||TIME(S)||'  =====>    MENU SYSTEM PASSWORDS VERIFICATION IS
ACTIVE    <=====  '
'CP SET SMSG ON'
DO FOREVER                                    /* INFINITE LOOP       */
SERVPASW:                                     /* LOOP STARTS HERE    */
'WAKEUP (SMSG EXT QUIET'                      /* WAIT UNTIL SMSG OR
                                                 EXTERNAL INTERRUPT  */
IF RC=6 THEN SIGNAL OUTOUT                    /* EXIT ON EXT INTRPT  */
PARSE PULL TYPE SMSGUSER UID PSWD NPSWD       /* PARSE TEXT DATA     */
           /* THIS IS TO TERMINATE MENU SYSTEM PASSWORDS VERIFICATION */
           /* YOU SIGN ON WITH USERID "EXIT" AND PASSWORD AS 'ADDPSWD'*/
IF STRIP(UID)='EXIT' & PSWD=ADDPSWD THEN DO
         'SMSG 'SMSGUSER' PSWDSHUT'
          SIGNAL OUTOUT                       /* EXIT WHEN EXIT DATA */
          END
           /* WHEN LOGO PROGRAM SENDS A MESSAGE TO TEST MENU SYSTEM   */
           /* PASSWORDS VERIFICATION STATUS, WE DO NOTHING            */
IF STRIP(UID)='TEST' THEN ITERATE
'STATE MENU XAUTH'
IF RC ¬=Ø THEN DO
'VMFCLEAR'
SAY '=====>    CANNOT FIND AUTHORIZATION FILE  !!!          '
EXIT 9999
END
/* CHECK MENU EXISTENCE */
'PIPE < MENU XAUTH ',                            /* DO WE HAVE A MENU - */
'| DROP 4 ',                                     /* - FOR THIS GUY ?    */
'| LOCATE 1-8 /'UID'/ ',
'| STEM TAFUSER.'
IF TAFUSER.Ø=Ø THEN DO                           /* NO USER MENU        */
'SMSG 'SMSGUSER' NOTAF'
SAY DATE(J)||TIME(S)||CØ98 UID' DOES NOT HAVE A WORKING MENU'
SIGNAL SERVPASW
END
'STATEW LOG1 XPASW A'
IF RC ¬=Ø THEN DO
'VMFCLEAR'
SAY '=====>    NO WRITE ACCESS TO PASSWORD FILE !!!         '
EXIT 9998
END
'PIPE (ENDCHAR ? ) < LOG1 XPASW A ',             /* EXISTING USER ?     */
'| DROP FIRST ',
'| LOCATE 1.8 /'UID'/ ',
'| A: FANOUT',
'| VAR PASWLINE',
'? A: | SPLIT MIN 28 AT /!/',
'| TAKE FIRST ',
'| SPLIT AT /!/',
'| STEM LOG1.'
IF LOG1.Ø=Ø THEN DO                              /* NEW MENU ID         */
 IF STRIP(NPSWD)='' THEN DO              /* ADD PASSWORD ¬ SUPPLIED */
 'SMSG 'SMSGUSER' NOADDPSWD'
  SAY DATE(J)||TIME(S)||CØ88 'MENU 'UID' DOES NOT EXIST AND'
  SAY DATE(J)||TIME(S)||CØ88 'ADD PASSWORD NOT SUPPLIED BY USER
'SMSGUSER
  END /* END NO ADD PASSWORD */
                  ELSE DO
 IF NPSWD=ADDPSWD THEN DO                        /* ADD PASSWORD IS OK !*/
  USER_LINE=LEFT(UID,8)||'!'||LEFT(PSWD,8)||'!'||DATE(J)||TIME(S)||'!'
 'PIPE VAR USER_LINE | >> LOG1 XPASW A'
 'SMSG 'SMSGUSER' ADDPSWDOK'
  SAY DATE(J)||TIME(S)||CØØ1 UID' WAS ADDED TO PASSWORDS DATA BASE BY
USER  'SMSGUSER
  END /* END NEW USERID      */
                  ELSE DO                        /* ADD PASSWORD ¬ OK   */
 'SMSG 'SMSGUSER' ADDPSWDNOTOK'
  SAY DATE(J)||TIME(S)||CØ98 SMSGUSER 'HAS FAILED TO ADD A NEW MENU '
 END /* END ADD NEW MENU */
END
END /* END EXISTING MENU */
            ELSE CALL CHKPSWD                  /* EXISTING USER       */
END /* END INFINITE LOOP */
EXIT
/* CHKPSWD: CHECK PASSWORD PROCEDURE */
CHKPSWD: PROCEDURE EXPOSE LOG1. PSWD NPSWD UID SMSGUSER PASWLINE
IF STRIP(PSWD)¬=STRIP(LOG1.2) THEN DO        /* NO MATCH            */
'SMSG 'SMSGUSER' NOMATCH'
SAY DATE(J)||TIME(S)||CØ99 UID' HAS FAILED TO ACCESS MENU SYSTEM ON USER
'SMSGUSER
   END                                         /* END NO MATCH        */
                                 ELSE DO       /* PSWD MATCH          */
IF STRIP(NPSWD)¬='' & STRIP(NPSWD)¬=STRIP(LOG1.2) THEN DO
USER_LINE=LEFT(UID,8)||'!'||LEFT(STRIP(NPSWD),8)||'!',
||DATE(J)||TIME(S)||'!'||LEFT(PASWLINE,87)
'PIPE < LOG1 XPASW A | CHANGE /'PASWLINE'/'USER_LINE'/ | > LOG1 XPASW A'
SAY DATE(J)||TIME(S)||CØØ3 UID' HAS CHANGED HIS ACCESS PASSWORD'
'SMSG 'SMSGUSER' CHDPSWDOK'
RETURN
END                                            /* END CHANGE PASSWORD */
SAY DATE(J)||TIME(S)||CØØ2 UID,
              ' HAS SUCCESSFULLY ACCESSED MENU SYSTEM ON USER 'SMSGUSER
'SMSG 'SMSGUSER' GOGOGO'
  END                                          /* END PSWD MATCH      */
RETURN                                       /* END CHKPSWD PROCEDURE */
OUTOUT:
SAY DATE(J)||TIME(S)||' =====>   MENU SYSTEM PASSWORDS VERIFICATION IS
DE-ACTIVATED <===== '
'CP SET SMSG OFF'
EXIT
```

Editor's note: this article will be continued next month.

Jaakov J Hazan
Technical Support Manager
Ynon Technologies & Computers (Israel)

Tell us what you have done to make working with VM/CMS easier or quicker at your site. Articles for *VM Update* can be sent to the editor, Trevor Eddolls, at any of the addresses shown on page 2. Alternatively, articles can be sent using the Internet to 100325.3711@compuserve.com. We welcome very short 'hints and tips' type articles as well as longer discussion articles. And we always welcome examples of code.

VM news

LinkAge Software has announced LinkAge Message Exchange, a message switch that promises enterprise-wide connectivity, integrated directory services, and administration and management tools for mixed environments.

The product, which is integrated with Microsoft Exchange Server and NT Server, is claimed to be the first messaging system to extend Microsoft technology into all IBM and Lotus environments. It connects users and integrates the directory services and management of most messaging systems, including MS Mail and Exchange, Lotus Notes and cc:Mail, SMTP/MIME, X.400, OfficeVision/VM, OfficeVision/MVS, OfficeVision/400, Verimation MEMO, Fischer International TAO and other host-based messaging systems that are compliant with SNADS.

There are four components: Microsoft Exchange-Notes Connector, Microsoft Exchange-cc:Mail Connector, Microsoft Exchange-SNADS Connector, and the Microsoft Exchange-OfficeVision/VM Connector. Each can be configured and managed from within the Exchange Administrator user interface. LME is also tightly integrated with other components of the Microsoft BackOffice family of server applications. LinkAge Message Exchange is available immediately.

For further information contact:
LinkAge Software, 11 Church Street, Suite 402, Toronto, ON, M5E 1W1, Canada.
Tel: (416) 862 7148.

* * *

Fischer International has unveiled Version 4.0 of TAO, its enterprise messaging and office automation product. The new version has an improved look and feel that adapts to the environment it runs in, and Windows 95 and NT clients now support OLE and Container technology, so users can send video clips, sound bites, and other large attachments, such as an Excel file, a sound clip from a sales presentation, or a video message from an organization's spokesperson.

TAO clients are available for Windows 3.*x*, Windows 95, Windows NT and OS/2 as well as large systems running OS/400, CICS, VTAM, TSO, IDMS, IMS, and CMS. The server version runs on VM, MVS, AS/400 and OS/2.

There is a new option for logging on as a live or disconnected client. Workers can take TAO on the road and access e-mail locally. The new GUI-based scheduling system features real-time, cross platform scheduling across an enterprise, with daily, weekly, and monthly calendar views. New features include drag and drop for private and public appointment scheduling and rescheduling, conflict checking for all types of appointments, and attendance confirmation from a notification.

For further information contact:
Fischer International, 4073 Mercantile Ave, Naples, FL 33942, USA.
Tel: (941) 643 1500.
Fischer International, 8 Beaumont Gate, Shenley Hill, Radlett, Herts, WD7 7AR, UK.
Tel: (01923) 859119.

xephon