January 2001

In this issue

3 Sending e-mail from REXX
6 A master catalog compare program for alias records
10 Dynamic allocation of datasets
15 Write to operator REXX function
22 Reordering VARY commands in SYS1.PARMLIB members
27 Cleaning volumes
42 Virtual storage map
52 Vary disk off-line during IPL using VOLSER patterns
63 Re-entrant programming
68 GDG transfer between systems – modified
72 MVS news

© Xephon plc 2001
Editor
Jaime Kaminski

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.

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

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 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.
Sending e-mail from REXX

INTRODUCTION
The following REXX procedure and ISPF panel illustrates a simple e-mail facility that can be used to send an e-mail from an ISPF session. The REXX program uses the REXX built-in function SOCKET to access the TCP/IP socket interface – this API is documented in the manual *IP Applications Programming Interface Guide*.

The SMTP (Simple Mail Transfer Protocol) commands and data flows are documented in RFC821, information about which can be found on the Internet.

REXX PROCEDURE

/* REXX */

server = 'your-server-name'
address ISPEXEC "DISPLAY PANEL(email)"
if server = '' | toaddr = '' then
   do
      say 'you need to fill in the panel !'
      exit
   end

call initialize
r = Socket('Connect',socket_id,'AF_INET 25' ip_address)
if word(r,1) ¬= Ø then
   call handle_error 'Connect',r
data = read_Socket( )
if substr(data,1,3) ¬= '220' then /* SMTP server Welcome message */
   call handle_error 'no 220 message',data
r = write_socket('Helo Rexx.Email'crlf)
data = read_socket( )
if substr(data,1,3) ¬= '250' then
   call handle_error 'no 250 message',data
r = write_socket('Mail From: <>'crlf)
data = read_socket( )
if substr(data,1,3) ¬= '250' then
   call handle_error 'no 250 message',data
r = write_socket('Rcpt To: <' || toaddr || '>' || crlf)
data = read_socket( )
if substr(data,1,3) ¬= '250' then
   call handle_error 'no 250 message',data
r = write_socket('Data'crlf)
data = read_socket()
if substr(data,1,3) ≠ '354' then
call handle_error 'no 354 message',data
r = write_socket('From: ' || from || crlf)
r = write_socket('To: ' || to || crlf)
r = write_socket('Subject: ' || subject || crlf || crlf)
r = write_socket(msg1 msg2 msg3)
r = write_socket(end_of_message)
data = read_socket()
if substr(data,1,3) ≠ '250' then
call handle_error 'no 250 message',data
say data
r = write_socket('Quit'crlf)
data = read_socket()
if substr(data,1,3) ≠ '221' then
call handle_error 'no 221 message',data
say data
r = Socket('Close',socket_id)
if word(r,1) ≠ Ø then
call handle_error 'Close',r
r = Socket('Terminate','Email')
if word(r,1) ≠ Ø then
say 'Error : Terminate 'r
exit7
handle_error :
parse arg type,text
say 'Error : ' type
if text ≠ '' then
say text
if initialized then
do
say socket('SocketSetStatus')
say socket('Terminate','Email')
end
exit
initialize :
true = 1
false = Ø
initialized = false
crlf = 'ØD25'x
end_of_message = 'ØD254ØD25'x
expose_variables = 'socket_id'
r = Socket('Initialize','Email')
if word(r,1) ≠ Ø then
call handle_error 'Initialise',r
r = socket('GetHostByName',server)
if word(r,1) ≠ Ø then
call handle_error 'GetHostByName',r
ip_address = word(r,2)
r = Socket('Socket',2,'Stream','Ø')
if word(r,1) ¬= Ø then
call handle_error 'Socket',r
else
    socket_id = word(r,2)
    r = Socket('SetSockOpt',socket_id,'Sol_Socket','So_ASCII','On')
    if word(r,1) ¬= Ø then
        call handle_error 'SetSockOpt',r
    initialized = true
return

write_socket : procedure expose (expose_variables)
    parse arg data
    r = Socket('Write',socket_id,data)
    if word(r,1) ¬= Ø then
        call handle_error 'Write'
return (r)

read_socket : procedure expose (expose_variables)
    r = Socket('Read',socket_id)
    parse var r src . data
    if src ¬= Ø then
        call handle_error 'Read',r
return (data)

ISPF PANEL
)
)ATTR
% TYPE(TEXT)   COLOR(WHITE)
# TYPE(TEXT)   COLOR(BLUE) SKIP(ON)
_ TYPE(INPUT)  COLOR(YELLOW) CAPS(OFF)
)BODY
% -----------------------------%#e-mail%---------------------------------
#
#SMTP Server name :_server  
#  Recipient's email Id
#Email :_toaddr  
#From :_from  
#To :_to  
#Subject :_subject  
#Message :_msg1       
#  _msg2       
#  _msg3       
)END
A master catalog compare program for alias records

The following REXX program uses the Catalog Search Facility (CSI) to extract alias entries from two catalogs and identify any missing or inconsistent entries. The CSI is documented in the Managing Catalogs manual and there are several code samples in SYS1.SAMPLIB. While this particular example examines only alias records, a similar approach could be used for other entry types to create a comprehensive catalog compare utility.

SAMPLE OUTPUT

Beginning ALIAS Check for CATALOG.MCAT1 and CATALOG.MCAT2

ABCD is not in CATALOG.MCAT1

/* entries missing from CATALOG.MCAT1 */

DEFINE ALIAS ( NAME( ABCD ) )
RELATE( CATALOG.USERCAT1 )
CATALOG(CATALOG.MCAT1)

THE SOURCE

/* Rexx */
parse arg catalog1 ',' catalog2
call initialize
if ¬valid_catalog(catalog1) then
   do
      say catalog1' is not a valid catalog'
      exit
   end
if ¬valid_catalog(catalog2) then
   do
      say catalog2' is not a valid catalog'
      exit
   end
cat1_alias = get_alias(catalog1,'cat1')
cat2_alias = get_alias(catalog2,'cat2')
parse var cat1_alias cat1_alias ',' cat1_related
parse var cat2_alias cat2_alias ',' cat2_related
say ','
say 'Beginning ALIAS Check for 'catalog1' and 'catalog2
say ' ' do forever select
    when (cat1_alias < cat2_alias) then do
        say cat1_alias ' is not in 'cat2.catalog
        say ' ' call punch_it 'out2',cat1_alias,cat1_related,cat2.catalog
        cat1_alias = get_alias('catalog.mcatc','cat1')
        parse var cat1_alias cat1_alias ',' cat1_related
    end
    when (cat1_alias = cat2_alias) then do
        if cat1_related ¬= cat2_related then do
            say cat1_alias' has different associations -'
            say ' ' 'cat1_related' in 'cat1.catalog
            say ' ' 'cat2_related' in 'cat2.catalog
            say ' '
        end
        cat1_alias = get_alias('catalog.mcatc','cat1')
        cat2_alias = get_alias('catalog.mcatd','cat2')
        parse var cat1_alias cat1_alias ',' cat1_related
        parse var cat2_alias cat2_alias ',' cat2_related
    end
    when (cat1_alias > cat2_alias) then do
        say cat2_alias ' is not in 'cat1.catalog
        say ' ' call punch_it 'out1',cat2_alias,cat2_related,cat1.catalog
        cat2_alias = get_alias('catalog.mcatd','cat2')
        parse var cat2_alias cat2_alias ',' cat2_related
    end
    end
    if cat1.eof & cat2.eof then leave
end
if out1.open then "execio Ø diskw out1 (finis"
if out2.open then "execio Ø diskw out2 (finis"
exit
initialize :
    true = 1
    false = Ø
    high_values = copies('FF'x,8)
    pl = copies(' ',4)          /* rsn code/return code from CSI */
    cat1.csifield = copies(' ',200)
    cat1.results = '00002000'x || copies('00'x,8192-4)
    cat1.eof = false
    cat2.csifield = copies(' ',200)
get_alias : procedure expose (expose_variables)

arg catalog_name,instance
p2 = instance || '.csifield' /* see SYS1.MACLIB(IGGCSI) */
p3 = instance || '.results'
iggcsi00_call = false
interpret "test = symbol('"instance".catalog')"
if test ≠ 'VAR' then
  do
    interpret instance".catalog = "catalog_name
    interpret p2" = overlay('**','p2',1) /* csifiltk */
    interpret p2" = overlay('"catalog_name","p2",45')/* csicatnm */
    interpret p2" = overlay('X','p2',133) /* csidtyps */
    interpret p2" = overlay('Y','p2',151) /* csis1cat */
    interpret p2" = overlay('000l','p2',153) /* csinumen */
    interpret p2" = overlay('NAME','p2',155) /* csifldnm */
    iggcsi00_call = true
  end
else
  do
    interpret "csiusdln = c2d(substr("p3",9,4))"
    interpret "i = "instance".pointer"
    interpret "csiresum = substr("p2",150,1)"
    if i > csiusdln then
      if csiresum = 'Y' then
        iggcsi00_call = true
      else
        do
          interpret instance".eof = true"
          alias = high_values
        end
      end
    else
      do
        interpret "alias = strip(substr("p3",i+2,44),'T')"
        interpret "length = c2d(substr("p3",i+46,2))"
        interpret "related = strip(substr("p3",i+46+6,44),'T')"
        interpret instance".pointer = i + 46 + length"
      end
    end
  end
if iggcsi00_call then
  do
    interpret "adress linkpgm 'IGGCSI00 p1 "p2" "p3"'
    if rc ≠ 0 then
      signal CSI_ERROR
      interpret "alias = strip(substr("p3",65+2,44),'T')"
      interpret "length = c2d(substr("p3",65+46,2))"
  end
interpret "related = strip(substr("p3",65+46+64),'.T')"
interpret instance".pointer = 65 + 46 + length"
end
return (alias',related)
valid_catalog : procedure expose true false p1 arg name
  if name = '' then
    return (false)
  p2 = copies(' ',200)
p3 = 'ØØØØ1ØØØ'x || copies('ØØ'x,496-4)
p2 = overlay(name,p2,1) /* csifiltk */
p2 = overlay('Y',p2,151) /* csis1cat */
p2 = overlay('ØØØØ'x,p2,153) /* csinumen */
address linkpgm 'IGGCSIØØ p1 p2 p3'
  if rc ¬= Ø then
    signal CSI_ERROR
  if substr(p3,15,1) = '4Ø'x then /* csicflg */
    return (false) /* not found */
  if strip(substr(p3,17,44),'.T') = name then /* csicname */
    return (true) /* must be the master catalog */
  if substr(p3,66,1) = 'U' then /* csietype */
    return (true) /* must be a connected catalog */
  else
    return (false) /* not a catalog */
punch_it : procedure expose true out1.open out2.open arg ddname,alias_name,related_name,catalog_name
interpret "alloc_done = "ddname".open"
  if ¬alloc_done then
    do
      interpret ddname".open = true"
      interpret "'allocate file("ddname") sysout(T)'"
      rec.1 = left(' /* entries missing from 'catalog_name' */',8Ø)
      rec.2 = left(' ',8Ø)
      interpret "'execio 2 diskw "ddname" (stem rec.'"
      end
      rec.1 = left(' DEFINE ALIAS ( NAME( 'alias_name' )',8Ø)
      rec.1 = overlay(' -',rec.1,7Ø)
      rec.2 = left(' RELATE( 'related_name' ) )',8Ø)
      rec.2 = overlay(' -',rec.2,7Ø)
      rec.3 = left(' CATALOG('catalog_name')',8Ø)
      rec.4 = left(' ',8Ø)
      interpret "'execio 4 diskw "ddname" (stem rec.'"
    end
    return
CSI_ERROR :
    say 'CSI error raised at 'sigl
    say ' rc 'rc
    say ' reason is 'c2d(substr(p1,3,1))
exit 8

Systems Programmer (UK) © Xephon 2001

Dynamic allocation of datasets

The following program performs dynamic allocation of new datasets, sequential or PDS, by issuing an SVC 99 with the appropriate request block. It was designed to be called from another program with four parameters – DDname, DSN, number of tracks for primary space, and directory blocks. If the directory blocks number is zero, a sequential file is created. The secondary space is approximately one fourth of the primary space, with a minimum of one track. Allocation is always in tracks. DSN is the physical name of the dataset, and the DDname is associated with it on creation, so the calling program can open it.

Other characteristics are a disposition of (NEW,CATLG,DELETE), to say it in JCL style, release unused space (RLSE) and free DDname associated after the file is opened and closed by the calling program. This allows a single DDname to be used for several datasets created by one program.

DCB stuff like RECFM, BLKSZ or LRECL are left unassigned, and it is up to the calling program to define the DCB with this characteristic prior to opening the file. Upon return, R0 and R15 contain the reason code and the return code. R15 should be zero, if allocation was successful.

This program was created to allow an input file to be split in to an undefined number of smaller files. Since I do not know that number beforehand, I want to create the files dynamically as needed, and always use the same DD for the output. For this task, and as an example, I outline here the skeleton of such a program.

Open input file
Loop1 until end of input
   Define a new name for output file to field DSN1
   CALL DYNALOC1,(DD1,DSN1,TRKS,BLKS)
   Open OUTDCB
   Loop2 for desired number of recs
   Read input
   Write output
Endloop2
Close OUTDCB
Endloop1

OUTDCB DCB DSORG=PS,RECFM=,LRECL=,DDNAME=DDOUT
This is just a suggestion, and many other uses are possible. I tried to document the program in such a way that it is easy to understand, even if you are not familiar with DYNALLOC (or SVC 99) request blocks. You can change my default options, or add others of your choice, by creating other keys and the associated text pointers. For more detailed information, refer to application development guide: Authorized Assembler Language Programs, GC28-1645.

**DYNALLOC1**

```
*====================================================================*
*                                                                    *
* DYNALOC1 - DYNALLOC FUNCTION 1 - Allocate new dataset.             *
*                                                                    *
* Parameters:                                                        *
*  Parm1: DDNAME - CL8  DDname to allocate.                          *
*  Parm2: DSNAME - CL44 Datasetname to allocate.                     *
*  Parm3: TRACKS - H    Primary space in tracks.                     *
*  Parm4: BLOCKS - H    Directory blocks for PDS (Ø for sequential). *
*                                                                    *
*  Returns: Retcode in R15 and reason code in RØ, as set by SVC99.   *
*                                                                    *
* The dataset is allocated with NEW,CATLG,DELETE, Free DD on close. *
* The secondary space is calculated to be about 25% of the primary   *
* space, with a minimum of one track. Unit is SYSDA.                 *
* Assembler produces a warning for alignment error. Ignore.          *
*                                                                    *
*====================================================================*
```

```assembly
&PROGRAM SETC  'DYNALOC1'
&PROGRAM CSECT
&PROGRAM AMODE 31
&PROGRAM RMODE 24
SAVE (14,12)
LR    R12,R15
USING &PROGRAM,R12
ST    R13,SAVEA+4
LA    R11,SAVEA
ST    R11,8(R13)
LR    R13,R11
B     CONTINUA
DC    CL16' &PROGRAM 1.0'
DC    CL8'&SYSDATE'
```
CONTINUADS ØH
LR R2,R1 Copy parameter pointer to R2.
L R3,Ø(Ø,R2) Load DDname parm address in R3
MVC DDNAME1,Ø(R3) Move DDNAME
L R3,4(Ø,R2) Load DSN parm address IN R3
MVC DSNAME1,Ø(R3) Move DSNAME
L R3,8(Ø,R2) Load TRACKS parm address in R3
MVC DYSPPRI1+1(2),Ø(R3) Move tracks to primary space
LH R3,DYSPPRI1+1 Load tracks value (2 bytes)
SRL R3,2(Ø) Divide by 4 (secondary space)
LTR R3,R3 Zero result?
BH STORTRAC No, jump ahead.
LA R3,1(Ø,R3) Yes, force result to 1.

STORTRAC EQU *
STH R3,DYSPSEC1+1 Store secondary space (2 bytes).
L R3,12(Ø,R2) Load BLOCKS parm address in R3
MVC DYBLOCK1+1(2),Ø(R3) Load blocks value (2 bytes).

LENGTHS EQU * Find DDname and DSName length.
XR R9,R9
LA R9,8(Ø,R9) R9=8: maximum length of DDname
LA R4,DDNAME1 Load DDname address.
XR R8,R8 Clear character counter.
BAL R1Ø,FINDSPC Call sub to count characteres.
STH R8,DYDDLENG Store returned length.

XR R9,R9 Same thing for DSN
LA R9,44(Ø,R9) 44 is maximum length
LA R4,DSNAME1
BAL R1Ø,FINDSPC
STH R8,DYDSLENG

LA R1,DYNADDR
DYNALLOC Call SVC 99.

EXITØ EQU * Exit. R15 (Return code) and
L R13,SAVEA+4 R0 (reason code) are kept as
L R14,12(R13) set by DINALLOC.
LM R1,R12,24(R13) If everything OK, R15 is zero.
BR R14

FINDSPC EQU * Find first space or low-value in
CLI Ø(R4),X'40' string pointed by R4.
BE FINDSPCF R8 returns the number of chars.
CLI Ø(R4),X'ØØ' R9 is maximum length.
BE FINDSPCF
LA R8,1(Ø,R8) Increment counter.
LA R4.1(Ø,R4)  Increment address.
CR R8,R9     Compare to max length.
BL FINDSPC

FINDSPCF BR RIØ  RIØ is the return address.

SAVEA DS 18F  Register save area

DYNADDR DS ØF  Dynalloc areas.
DC X'8Ø'  Last pointer flag (high bit on).
DC AL3(DYNREQUE)  Request block pointer.

DYNREQUE DS ØCL2Ø  Length of this request area.
DYNLENGT DC X'14'  Verb code Ø1 - DSN allocation.
DYNFLAGS DC H'Ø'  Error reason code
DYNERRCD DC H'Ø'  Informational reason code
DYNLISAP DC A(DYNTXTPT)  Address of text pointers.
DYNRBEXT DC F'Ø'  No request block extension.
DYNFLAG2 DC F'Ø'  Flags for authorized functions.

DYNTXTPT EQU *  Text (keys) pointers
DC A(DYDDNAME)
DC A(DYSNAME)
DC A(DYSTATUS)
DC A(DYDISP1)
DC A(DYDISP2)
DC A(DYTYPE)
DC A(DYSPR)
DC A(DYSEC)
DC A(DYBLK)
DC A(DYRLS)
DC A(DYUNIT)
DC X'8Ø'  Last text pointer has high bit on.
DC AL3(DYCLOS)

DYDDNAME DC X'ØØ01'  DDname key
DC X'ØØ01'
DYDDLENG DC X'ØØ08'  Length (stored by this program).
DDNAME1 DC CL8' '  DDname loaded from parameter 1.

DYDSNAME DC X'ØØ02'  Datasetname key
DC X'ØØ01'
DYDSLENG DC X'ØØ02'  Length (stored by this program).
DSNAME1 DC CL44' '  DSN loaded from parameter 2.

DYSTATUS DC X'ØØ04'  Key for first DISP
DC X'ØØ01'
DC X'ØØ01'
DC X'04'    First DISP=(NEW
*  
DYDISP1 DC X'0005'    Key for second DISP  
DC X'0001'  
DC X'0001'  
DC X'02'   Second DISP=(,CATALOG
*  
DYDISP2 DC X'0006'    Key for third DISP  
DC X'0001'  
DC X'0001'  
DC X'04'   Third DISP=(,,DELETE
*  
DYSPTYPE DC X'0007'    Key for allocation type TRACKS  
DC X'0000'  This key has no text (zero occurs).
*  
DYSPPRI DC X'000A'    Key for primary quantity  
DC X'0001'  
DC X'0003'   Length of DYSPPRI1 must be 3 bytes.  
DYSPPRI1 DC X'000000'   Primary value (from parameter 1).
*  
DYSPSEC DC X'000B'    Key for secondary quantity  
DC X'0001'  
DC X'0003'   Length is also 3 bytes.  
DYSPSEC1 DC X'000000'   Secondary value loaded by program.
*  
DYBLOCK DC X'000C'    Key for directory blocks.  
DC X'0001'  
DC X'0003'  
DYBLOCK1 DC X'000000'   Dir block number from parameter 4.
*  
DYRLSE DC X'000D'    Key for release space (RLSE).  
DC X'0000'   No text entry needed (Ø occurs)
*  
DYUNIT DC X'0015'    Unit type key  
DC X'0001'  
DYUNLENG DC X'0005'   Length of DYUNIT1 (SYSDA = 5 bytes)  
DYUNIT1 DC CL8'SYSDA'   Unit type text (max 8 bytes).
*  
Dyclose DC X'001C'    Key for FREE DD on close.  
DC X'0000'   No text needed.
*  
YREGS  
END  

Luis Paulo Ribeiro  
Systems Programmer  
Edinfor (Portugal)  

© Xephon 2001
Write to operator REXX function

The function should be invoked from within an MVS REXX EXEC that is typically running in background as opposed to foreground. This function causes a message to be written using the standard MVS WTO (Write To Operator) request.

This REXX function accepts a single argument – the message. This argument is mandatory, and must be less than 127 bytes in length (this is a simple WTO restriction). The function returns a standard return code indicating success or failure. The syntax of the function is:

```
REXWTO(-message-)
```

In keeping with standard REXX practices, the message specified may be a variable. The value of the message may be in upper, lower or mixed case. The function returns an integer. This integer will indicate success or failure. An example of the function being invoked:

```
RC = REXWTO('Please call 206-555-1212 on completion');
```

The different values that may be returned are as follows:

0 – Normal
4 – Message length > 126 bytes
8 – Invalid number of arguments.

An example of the function being used:

```
/* REXX ******************************************************/
... 
MSG = 'Successful execution'; 
RC = REXWTO(MSG); 
if RC = 0 then 
   exit; 

... 
```

```
REXWTO
```

```
     TITLE 'REXX FUNCTION TO ISSUE WTO'
     PRINT NOGEN

* 
*     PROGRAM:  REXWTO
```

OUTPUT MESSAGE TO OPERATOR

ATTRIBUTES:
REENTRANT
AMODE: 31
RMODE: ANY
AUTHORIZATION: NONE

ABSTRACT:
REXX FUNCTION REQUIRES A SINGLE ARGUMENT - THE MESSAGE TO BE
OUTPUT TO THE OPERATOR USING A WTO.
THE FUNCTION WILL RETURN ONE VALUE:
THE STANDARD RETURN CODE

USAGE:
RET_CODE = REXWTO(OUTPUT_MSG);

RET_CODE VALUES:
Ø     . NORMAL
4     . MESSAGE > 126 BYTES
8     . INVALID NUMBER OF ARGUMENTS

TITLE 'EQUATES, MACROS && CONTROL BLOCKS USED'
RØ    EQU  Ø
R1    EQU  1
R2    EQU  2
R3    EQU  3
R4    EQU  4
R5    EQU  5
R6    EQU  6
R7    EQU  7
R8    EQU  8
R9    EQU  9
R1Ø   EQU  1Ø
R11   EQU  11
R12   EQU  12
R13   EQU  13
R14   EQU  14
R15   EQU  15

TITLE 'MAIN CSECT PROCESS'

CALLED ROUTINES:
NONE

MACROS AND CONTROL BLOCKS USED:
IRXARGTB . MAP ARGUMENT TABLE
IRXEFPL  . MAP EXTERNAL FUNCTIONS PLIST
IRXEVALB . MAP EVALUATION BLOCK
IRXSHVB  . MAP SHARED VARIABLE BLOCK
STORAGE  . STORAGE ACQUIRE AND RELEASE

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
REXXTO  CSECT
REXXTO  AMODE 31
REXXTO  RMODE ANY
LA  R14,Ø(R14)  . VALIDITY OF R14
BSM  R14,R0  . CURRENT ADDRESSING MODE
BAKR  R14,R0  . ESTABLISH LINKAGE
LR  R12,R15  . 12 -> EPA
USING REXWTO,R12  . CSECT ADDRESSABILITY
STORAGE OBTAIN,  . ACQUIRE DYNAMIC AREA *
  ADDR=(R13),  *
  LENGTH=DYNLEN,  *
  SP=Ø
MVC  $(4(R13),=C'F1SA')  . INDICATE FORMAT OF SAVE AREA
USING DYNAREA,R13  . DSECT ADDRESSABILITY
SLR  R8,R8  . SET RETURN CODE
ST  R8,RETCODE  . SAVE RETURN CODE
BAS  R1Ø,REXXVECT  . REXX VECTOR PROCESSING
BAS  R1Ø,ARGUMENT  . PROCESS ARGUMENT
LTR  R8,R8  . Q. ARGUMENT VALID?
BNZ  AØØ1  . A. YES
BAS  R1Ø,DOWTO  . OUTPUT MESSAGE
*
AØØ1  EQU  *
*
BAS  R1Ø,TERMINAT  . TERMINATION
STORAGE RELEASE,  . RELEASE DYNAMIC STORAGE *
  ADDR=(R13),  *
  LENGTH=DYNLEN,  *
  SP=Ø
SLR  R15,R15  . 15 - RETURN CODE
PR  . ADIOS
TITLE 'REXX VECTOR PROCESSING'
*
PROCESS THE TWO ARGUMENTS PASSED TO REXX FUNCTIONS
*
THE ADDRESS OF THE REXX ENVIRONMENT BLOCK (OPTIONAL)
*
THE ADDRESS OF THE EXTERNAL FUNCTION PARAMETER LIST
*
*
REGISTER USAGE
*
Ø  . -> ENVIRONMENT BLOCK
1  . -> EXTERNAL FUNCTION PLIST
2  . -> PARSED PARAMETER LIST
*
REXXVECT  EQU  *
*
EREGRØ,R1  . EXTRACT CALLER'S REGISTERS
ST  RØ,REXX  . SAVE REXX ENVIRONMENT BLOCK ->
ST  R1,EFPL  . SAVE EXTERNAL FUNCTION PLIST
USING  EFPL,R1  . IRXEFPL DSECT ADDRESSABILITY
L  R2,EFPLARG  . 2 -> PARSED ARGUMENT LIST
ST  R2,ARGTAB  . SAVE
L  R2,EFPLEVAL  . 2 -> EVALUATION BLOCK VECTOR
L    R2,0(,R2)       . 2 -> EVALUATION BLOCK
ST   R2,@EVALBLK    . SAVE
DROP  R1            . DSECT NOT REQUIRED

*  
BR   R1Ø            . RETURN
TITLE 'PROCESS INPUT ARGUMENT'
*  
PROCESS ARGUMENT - VALIDATE ETC.
*  
ONE MANDATORY ARGUMENT - MAX 126 BYTES, MIN ONE BYTE
*  
*  
REGISTER USAGE
  *  
    1                             . ARGUMENT COUNT
    2                             . -> CURRENT ARG TABLE ENTRY
    3                             . -> SHARED VARIABLE BLOCK
    4                             . WORK
    5                             . -> CURRENT ARGUMENT VALUE
    6                             . CURRENT ARGUMENT LENGTH
    7                             . -> SAVED VALUE
    8                             . WORK
    9                             . ARGUMENT LENGTH
    10                            . RETURN
  
ARGUMENT EQU *
  *
SLR  R1,R1          . 1 - ZERO (ARGUMENT COUNT)
ICM  R2,15,@ARGTAB  . 2 -> ARGUMENT TABLE
BZ   CØØØ2          . BRANCH IF ZERO
USING ARGTABLE_ENTRY,R2 . DSECT ADDRESSABILITY
  *
LM    R4,R5,ARGTABLE_ARGSTRING_PTR
LTR   R5,R5         . 4 -> ARGUMENT STRING
BM   CØØØ2          . A. YES - NEGATIVE
BZ   CØØØ3          . A. YES - ZERO
LA    R1,1(,R1)     . INCREMENT ARGUMENT COUNT
CH    R5,=Y(L'MSG)  . Q. VARIABLE NAME TOO GREAT?
BH    CØØØ3          . A. YES - ERROR
MVI   MSG,C' '      . INITIALIZE MESSAGE AREA
MVC   Ø(1,R6),Ø(R4) . MOVE BYTE TO SAVE MESSAGE
LA    R4,1(,R4)     . 4 -> NEXT BYTE OF MESSAGE
LA    R6,1(,R6)     . 6 -> NEXT BYTE OF SAVED MSG
BCT   R5,CØØØ1      . LOOP THROUGH MESSAGE
STH   R7,MSGLEN     . SAVE LENGTH
  *
CØØØ1 EQU *
  *
MVC   Ø(1,R6),Ø(R4) . MOVE BYTE TO SAVE MESSAGE
LA    R4,1(,R4)     . 4 -> NEXT BYTE OF MESSAGE
LA    R6,1(,R6)     . 6 -> NEXT BYTE OF SAVED MSG
BCT   R5,CØØØ1      . LOOP THROUGH MESSAGE
STH   R7,MSGLEN     . SAVE LENGTH
  *
© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
LA R2,ARGTABLE_NEXT-ARGTABLE_ENTRY(R2)
* . 4 -> ARGUMENT STRING
* . 5 - ARGUMENT STRING LENGTH
LM R4,R5,ARGTABLE_ARGSTRING_PTR
LTR R5,R5 . Q. LENGTH NEGATIVE?
BM C0004 . A. YES
LA R8,8 . SET RETURN CODE
B C0004
*
C0002 EQU *
*
CH R1,=H'1' . Q. VALID NUMBER OF ARGUMENTS?
BE C0004 . A. YES
LA R8,8 . SET RETURN CODE
B C0004 . CONTINUE
*
C0003 EQU * . ARGUMENT LENGTH ERROR
*
LA R8,4 . SET RETURN CODE
*
C0004 EQU *
*
DROP R2 . DSECT NOT REQUIRED
ST R8,RETCODE . SAVE RETURN CODE
BR R1Ø
TITLE 'OUTPUT THE MESSAGE'
* BUILD THE WTO AREA
* ISSUE THE WTO
*
* REGISTER USAGE
* 2 . MESSAGE LENGTH
* 3 . -> MESSAGE
*
DOWTO EQU *
*
LH R2,MSGLEN . 2 - MESSAGE LENGTH
AH R2,=H'4' . INCREMENT FOR HEADER
STH R2,MSGLEN . AND SAVE
XC MCSFLAGS,MCSFLAGS . ZAP MCS FLAGS
LA R3,MSG . 3 -> MESSAGE
LA R3,Ø(R2,R3) . 3 -> DESCRIPTOR CODES
XC Ø(2,R3),Ø(R3) . SET DESCRIPTOR CODES
MVC 2(2,R3),=X'0020' . SET ROUTE CODE 11
WTO MF=(E,WTOAREA) . ISSUE WTO
BR R1Ø
TITLE 'TERMINATION ROUTINE'
* SET UP REXX FUNCTION RETURN CODE
* PUT RETURN VALUE INTO REXX EVALUATION BLOCK
*
* REGISTER USAGE
**TERMINATE**

```
SLR R1, R1       . 1 - ZERO
LA R2, RETDATA   . 2 -> OUTPUT DATA
MVC RETDATA, SPACES . INITIALIZE OUTPUT
L R3, RETCODE    . 3 - RETURN CODE
LTR R3, R3       . Q. RETURN CODE NEGATIVE?
BNM EØØØ1       . A. NO
MVI Ø(R2), C'-'  . OUTPUT NEGATIVE SIGN
LA R1, 1(R1)     . INCREMENT BYTES OUTPUT
LA R2, 1(R2)     . 2 -> NEXT OUTPUT BYTE

EØØØ1 EQU *
```

```
CVD R3, DWORD     . PACK IT
MVC VARWORK, MASK8 . MOVE EDIT MASK TO WORK AREA
ED VARWORK, DWORD+4 . EDIT THE DATA
LA R3, VARWORK    . 3 -> EDITED DATA
LA R4, L'VARWORK  . 4 - LENGTH OF EDITED DATA

EØØØ2 EQU *
```

```
CLI Ø(R3), C' '   . Q. SIGNIFICANT?
BNE EØØØ3       . A. YES
LA R3, 1(R3)     . 3 -> NEXT BYTE
BCT R4, EØØØ2    . LOOP

EØØØ3 EQU *
```

```
MVC Ø(1, R2), Ø(R3) . MOVE OUT BYTE
LA R1, 1(R1)       . INCREMENT BYTES OUTPUT
LA R2, 1(R2)       . 2 -> NEXT OUTPUT BYTE
LA R3, 1(R3)       . 3 -> NEXT INPUT BYTE
BCT R4, EØØØ3     . LOOP
ST R1, #RETDATA   . NUMBER OF BYTES

L R2, @EVALBLK    . 2 -> EVAL BLOCK
USING EVALBLOCK, R2 . DSECT ADDRESSABILITY
L R3, EVALBLOCK_EVSIZE . 3 - LENGTH
CH R3, =H'3'     . Q. AT LEAST THREE DOUBLES?
BL EØØØ4        . A. NO
MVC EVALBLOCK_EVDATA(4), RETDATA . SET RESULT
```
MVC EVALBLOCK_EVLLEN(4), RETDATA
DROP R2
*
EØ004 EQU *
*
BR R1Ø
DROP R13
TITLE 'DYNAMIC AREA'
DYNAREA DSECT
DS 18F
DWORD DS D . FOR CVD
@ARTAB DS F . -> ARGUMENT TABLE
@EFPL DS F . -> REXX EXT FUNCTION PLIST
@EVALBLK DS F . -> EVAL BLOCK
@REXX DS F . -> REXX ENVIRONMENT BLOCK
#RETDATA DS F . LENGTH OF RETURNED DATA
RETCODE DS F . RETURN CODE
REASCODE DS F . REASON CODE
*
WTOAREA DS ØF . WTO MESSAGE AREA
MSGLEN DS H . LENGTH OF MESSAGE
MCSFLAGS DS XL2 . MCS FLAGS
MSG DS CL126 . MESSAGE
DS XL2 . DESCRIPTOR CODES
DS XL2 . ROUTING CODES
*
RETDATA DS CL8 . RETURN DATA
VARWORK DS CL8 . VARIABLE NUMBER WORK
DS ØF
SHVARBLK DS CL(SHVBLEN) . SHARED VARIABLE BLOCK AREA
DYNLEN EQU *-DYNAREA
TITLE 'IBM SUPPLIED DSECTS'
IRXARGTB . ARGUMENT TABLE
IRXEFPFL . EXTERNAL FUNCTION PARAM LIST
IRXEVALB . EVALUATION BLOCK
IRXSHVBL . SHARED VARIABLE REQUEST BLOCK
TITLE 'CONSTANTS'
REXWTO CSECT
*
MASK8 DC X'4Ø2Ø2Ø2Ø2Ø2Ø212Ø' . EDIT MASK
SPACES DC BC' ' . SPACE FILL
*
LTORG
END REXWTO

Dave Loveluck
Consultant (USA) © Xephon 2001
Reordering VARY commands in SYS1.PARMLIB members

INTRODUCTION

A COMMAND\textsubscript{xx} parmlib member often contains VARY OFF commands. Since at IPL all devices are placed on-line, VARY OFF commands are necessary to put off-line devices that belong to other LPARs, or for some other reason we do not want them on-line. In theory, only VARY OFF commands are necessary, but since the addresses can be specified in ranges, sometimes we put a large range offline and then put a few on-line. This way, things can become a little confusing. For example, consider the following group of commands:

\begin{verbatim}
COM='VARY 04F0-052C,OFFLINE'
COM='VARY 0510,ONLINE'
COM='VARY 03E0-04FA,OFFLINE'
\end{verbatim}

Here you can observe that there are two OFFLINE ranges overlapping, besides the ONLINE that splits the first range in two. Furthermore, ONLINE lines are position-dependent. If that line was the first in the group, it would become useless. This type of situation is not only confusing but also dangerous. Unfortunately, these things can happen out of a moment’s hurry to solve some situation, or out of lack of patience to write ranges correctly. And, if there are many lines involved, things are difficult to sort out, so one tends to leave things as they are, out of fear of making mistakes.

To deal with this type of problem, I wrote an EXEC that reads a COMMAND\textsubscript{xx} member and writes another member containing only the correct OFFLINE ranges. For example, the group above would be rewritten to the equivalent:

\begin{verbatim}
COM='VARY 03E0-050F,OFFLINE'
COM='VARY 0511-052C,OFFLINE'
\end{verbatim}

All the other lines in the file that are not VARY device commands are left unchanged. For example, VARY PATH or VARY CN are left unchanged. My criterion to recognize a device is a valid hexadecimal address, either single, part of a range, or a group of ranges. I also check if an address range is valid, that is, if the second address is greater than
the first. In my output file, VARY commands will begin at the same line where the first VARY device appears in the original file.

VARYOFF

/* REXX MVS */
/* */
/* VARYOFF */
/* This program reads a COMMANDxx member of SYS1.PARMLIB and writes */
/* an output file containing VARY ON and VARY OFF commands of the */
/* original file reordered and grouped as VARY OFF only commands. */
/* This only applies if VARY arguments are devices (hex addresses). */
/* All other VARY commands as well as other lines remain unchanged. */
/* */
/*===================================================================*/

arg ficin .
if ficin = "" then do
    say "Input file?"
    pull ficin .
    if ficin = "" then exit
end
ficin = strip(ficin,""")
say "Output file?"
pull ficout .
ficout = strip(ficout,""")
x = Ø
y = Ø
z = Ø
jmax = Ø
jmin = 99999
vary_first_line = Ø
call free_ddnames
call read_ficin
call separate_lines
call reorder_varyoffs
call write_ficout
saida:
call free_ddnames
exit

/*==================================================================*/
/*                          Subroutines                             */
/*==================================================================*/

free_ddnames:
z = msg(off)
"free dd(ficindd)"
"free dd(ficoutdd)"
return

read_ficin:
"alloc dd(ficindd) da('"ficin"') shr"
if rc <>Ø then do
    say "Error allocating" ficin rc
    signal saida
end
do k = 1 to 99999
    execio 1 diskr ficindd
    if rc <>Ø then leave
    pull ficinline.k
end
maxficinline = k-1
execio Ø diskr ficindd "(finis"
return

separate_lines:
do x = 1 to maxficinline
    linha = ficinline.x
    linhaØ = space(linha,Ø)
    v1 = pos("COM='V",linhaØ)
    v2 = pos("COM='VARY",linhaØ)
    v3 = pos("COMMAND='V",linhaØ)
    v4 = pos("COMMAND='VARY",linhaØ)
    of = pos("",OFFLINE",linhaØ)
    on = pos("",ONLINE",linhaØ)
    if (v1>Ø|v2>Ø|v3>Ø|v4>Ø) & (of>Ø|on>Ø) then do
        call get_vary_range
        if valid_address = Ø then do
            z = z+1
            out_others.z = linha
            iterate x
        end
        if vary_first_line = Ø then do
            vary_first_line = x
            if v1>Ø then cc = "COM='V"
            if v2>Ø then cc = "COM='VARY"
            if v3>Ø then cc = "COMMAND='V"
            if v4>Ø then cc = "COMMAND='VARY"
        end
    end
else do
    z = z+1
    out_others.z = linha
end
end
return
reorder_varyoffs:
k_prev = -1
jmax = jmax+1
table.jmax = 1
do k = jmin to jmax
   if table.k <> Ø then do
      if k_prev = -1 then iterate k
   else do
      k_ant = k-1
      if k_prev = k_ant then do
         y = y+1
         temp1 = right(d2x(k_prev),4,"Ø")
         out_vary.y = temp1",OFFLINE"
      end
      else do
         y = y + 1
         temp1 = right(d2x(k_prev),4,"Ø")
         temp2 = right(d2x(k_ant),4,"Ø")
         out_vary.y = temp1"-"temp2",OFFLINE"
      end
      k_prev = -1
   end
   else do
      if k_prev = -1 then k_prev = k
   end
end
return

get_vary_range:
valid_address = 1
if of > 1 then sym = Ø
if on > 1 then sym = 1
if v2 > Ø | v4 > Ø then do
   parse var linha . "'VARY" linha1 ",0"
end
else do
   parse var linha . "'V" linha1 ",0"
end
linha1 = space(linha1,Ø)
linha1 = strip(linha1,"(")
linha1 = strip(linha1,"")
linha1 = translate(linha1," ",")"
do k = 1 to words(linha1)
   www = word(linha1,k)
   parse var www range_1"-"range_2
   if datatype(range_1,"X") = Ø then do
      valid_address = Ø
      leave k
end
ranged = x2d(range_1)
if range_2 = "" then range2 = ranged
else range2 = x2d(range_2)
if range2 < ranged then do
  say "Invalid range: " range_1 range_2
  signal saida
end
do j = ranged to range2
  table.j = sym
  if j > jmax then jmax = j
  if j < jmin then jmin = j
end
end
return

write_ficout:
"alloc dd(ficoutdd) da('ficout') shr"
if rc <>Ø then do
  say "Error allocating" ficout rc
  signal saida
end
z = z+1
do z1 = 1 to z
  if z1 = vary_first_line then do
    do y1 = 1 to y
      queue cc out_vary.y1"
      execio 1 diskw ficoutdd
    end
  end
  if z1 = z then leave z1
  queue out_others.z1
  execio 1 diskw ficoutdd
end
execio Ø diskw ficoutdd "(finis"
return

Systems Programmer (UK) © Xephon 2001

Subscribers to *MVS Update* will no doubt be interested to know that Xephon publishes a quarterly journal devoted exclusively to TSO and ISPF topics. If you have a site license, it is very easy to add a subscription to *TSO/ISPF Update*. To find out more details, visit our Web site at [www.xephon.com/tsoispfupdate.html](http://www.xephon.com/tsoispfupdate.html), or contact us at any of the addresses on page 2 of this issue.
Cleaning volumes

This program can be run to ‘clean up’ selected volumes. It was written specifically to process LOG datasets (created by an application) that are deleted if empty, otherwise renamed and migrated to cartridge to prevent the work volumes from filling up. If datasets are found that are already renamed, these are migrated (and the empty ones are deleted). It can easily be updated to process whatever criteria are required to perform a similar function at other sites (for example you may not want to perform the rename with a time stamp, or might want more filtering). The program has two modes – ‘NORMAL’, where the standard processing described above is carried out, and ‘DELETE’, which can be specified in an emergency to delete the selected files. A report is produced detailing what actions have been taken.

SOURCE

TITLE 'CLEANUP - CLEAN UP FILES ON DISK'
******************************************************************************
*                                                                     *
* CLEANUP:  SCAN SELECTED VOLUMES, LOCATING ANY REQUIRED SEQUENTIAL      *
*           FILES ON THEM, AND PROCESSING THEM ACCORDING TO P ARMS.      *
*                                                                     *
* FOR 'SYSTMONE' VOLUMES STARTING 'SY1S' ARE SCANNED.                   *
* FOR 'SYSTMTWO' VOLUMES STARTING 'SY2S' ARE SCANNED.                   *
*                                                                     *
* P ARMS:    PASSED VIA 'PARM=...'. BOTH MANDATORY:                    *
*                                                                     *
* - SYSTEM NAME ('SYSTMONE' OR 'SYSTMTWO')                            *
* - RUN TYPE ('NORMAL' OR 'DELETE')                                    *
*                                                                     *
* NORMAL = DELETE EMPTY LG1, LG2 AND LG3 FILES                        *
* RENAME LG1, LG2 AND LG3 FILES, APPENDING A DATE/TIME STAMP          *
* FOR LG1 FILES START 'OFFLDLG1'                                      *
* FOR LG2 FILES START 'OFFLDLG2'                                      *
* FOR LG3 FILES START 'OFFLDLG3'                                      *
*                                                                     *
* DELETE = DELETE EMPTY SEQUENTIAL FILES                              *
* RENAME LG1 FILES, APPENDING A DATE/TIME STAMP                       *
* FOR LG2 OR LG3 FILES ISSUE A 'SCRATCH' MACRO TO DELETE THEM         *
* FOR LG1 FILES START 'OFFLDLG1'                                      *
******************************************************************************
* THIS PARM WOULD ONLY BE USED IF WE WERE * DESPERATE FOR SPACE AFTER SYSTEM CRASHES * AND DIDN'T HAVE TIME TO MIGRATE THEM OFF * TO CARTRIDGE BEFORE BRINGING IT UP AGAIN. *

***********************************************************************
PRINT NOGEN
***********************************************************************
* HOUSEKEEPING...
***********************************************************************
CLEANUP CSECT
CLEANUP AMODE 31
CLEANUP RMODE 24
BAKR R14,Ø SAVE CALLER DATA ON STACK
LR R12,R15 GET ENTRY POINT
LA R11,2Ø48(R12) LOAD SECOND BASE
LA R11,2Ø48(R11)
USING CLEANUP,R12,R11 ADDRESSABILITY
L R9,Ø(R1) SAVE ADDR OF PARM IN R9
***********************************************************************
* GET THE NAME OF OUR JOB (FOR WTO MESSAGES)... *
***********************************************************************
EXTRACT TIOTADDR,FIELDS=TIOT GET TIOT ADDRESS
L R3,TIOTADDR LOAD IT
MVC WTO1+8(8),Ø(R3) MOVE JOBNAME/STCNAME TO WTOS
MVC WTO2+8(8),Ø(R3)
MVC WTO3+8(8),Ø(R3)
MVC WTO4+8(8),Ø(R3)
MVC WTO5+8(8),Ø(R3)
***********************************************************************
* ENSURE PARMS EXIST, AND ARE FOR A VALID SYSTEM AND MODE... *
***********************************************************************
CLC Ø(2,R9),=H'ØØ15' PARM LENGTH = 15?
BNE BADPARM NO...INVALID
CLC 2(8,R9),SYSTMONE IS IT SYSTMONE?
BE ITSCAR1 YES..
CLC 2(8,R9),SYSTMTWO IS IT SYSTMTWO?
BE ITSCAR2 YES..
B BADPARM NO...INVALID

ITSCAR1 DS ØH
MVC HEAD1+1ØØ(8),SYSTMONE SET UP HEADING
MVI DSNAME+9,C'1' MAKE IT 'SYST1'
MVI VOLUME+2,C'1' LOOK ON 'SY1SØ' VOLUMES
B CHEKMODE CHECK FOR VALID MODE

ITSCAR2 DS ØH
MVC HEAD1+1ØØ(8),SYSTMTWO SET UP HEADING
MVI DSNAME+9,C'2' MAKE IT 'SYST2'
MVI VOLUME+2,C'2' LOOK ON 'SY2SØ' VOLUMES

CHEKMODE DS ØH
MVI SWITCH,NORM SET SWITCH FOR NORMAL MODE
CLC 1Ø(7,R9),NORMAL IS IT NORMAL?
BE INITBFL  YES...
MVI SWITCH,DEL  SET SWITCH FOR DELETE MODE
CLC 10(7,R9),DELETE  IS IT DELETE?
BE INITBFL  YES..initialize coffilt parms
B BDPARM  NO..INVALID MODE

***********************************************************************
* INITIALIZE BUFFER LIST HEADER (BFLH) AND ELEMENTS (BFLE)...  *
***********************************************************************

INITBFL  DS ØH

MVC HEAD1+6(6),11(R9)  MOVE MODE TO HEADING

LA R2,BFLHDEF  ADDRESS WORKAREA
LA R3,BFSIZE  LOAD LENGTH TO CLEAR
XR R5,R5  Ø PADDING BYTE
MVCL R2,R4  SET BUFFER LIST AREA TO ØS

LA R1,BFLHDEF  TEMP ADDRESSABILITY TO BFLH
USING BFLMAP,R1
MVI BFLNOE,BUFFNUM  SET NUMBER OF BUFFER ELEMENTS
OI BFLHFL,BFLHDSCB  ID AS DSCB BUFFER ELEMENT LIST
LA R2,BFLHDEF+BFLHLN  R2 -> 1ST BUFFER LIST ELEMENT
USING BFLE,R2  TEMP ADDRESSABILITY
LA R3,DSCBDEF  R3 -> 1ST DSCB BUFFER
LA R4,BUFFNUM  R4 = NO. OF ELEMENTS & BUFFERS

BFLEINIT  DS ØH

OI BFLEFL,BFLECHR  REQUEST CCHHR ON RETURN
MVI BFLETH,DSCBSIZE  SET BUFF LEN TO FULL DSCB SIZE
ST R3,BFLEBUF  SET A(DSCB BUFFER)
LA R2,BFLELN(R2)  R2 -> NEXT BUFFER LIST ELEMENT
LA R3,DSCBSIZE(R3)  R3 -> NEXT DSCB BUFFER
BCT R4,BFLEINIT  LOOP THROUGH ALL ELEMENTS
DROP R1,R2  DROP TEMP ADDRESSABILITY

***********************************************************************
* INITIALIZE THE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT...  *
***********************************************************************

XC FCLDEF(FCLSIZE),FCLDEF  SET FCL AREA TO Ø
LA R1,FCLDEF  R1 -> FCL HEADER
USING FCLMAP,R1  TEMP ADDRESSABILITY TO FCL
MVC FCLID,=C’FCL ’  SET EYECATCHER ‘FCL ’
MVC FCLCOUNT,=H’1’  SET NUMBER OF FCL ELEMENTS = 1
OI FCLFLAG,FCLIEQF1  RETURN ONLY FORMAT1 DSCBS
LA R2,FCLHEND  R2 -> 1ST (ONLY) FCL ELEMENT
USING FCLDSN,R2  TEMP ADDRESSABILITY
MVC FCLDSNLG,DSNAMELN  SET LENGTH OF DSN PATTERN
LA R3,DSNAME  GET DSN PATTERN ADDRESS
ST R3,FCLDSNA  ...SAVE IN FCL

***********************************************************************
* OPEN SYSPRINT DD AND WRITE OUT HEADINGS. IF THE OPEN FAILS CONTINUE *
* WITHOUT WRITING ANYTHING...                                       *
***********************************************************************
OPEN (SYSPRINT,(OUTPUT))
TM SYSPRINT+48,X'10'    OPEN OK?
BNO BADOPEN            NO...
PUT SYSPRINT,HEAD1   YES..PRINT HEADINGS
PUT SYSPRINT,HEAD2
PUT SYSPRINT,HEAD3
B SCANVOLS

BADOPEN DS 0H

WTO1 WTO 'CLEANUP: SYSPRINT OPEN FAILED - NO REPORT AVAILABLE'
OI PRTNOP+1,X'F0'    SET NO-OP TO SKIP PRINTING
OI CLSNOP+1,X'F0'    SET NO-OP TO SKIP CLOSE

***********************************************************************
* Scan through the UCBs, looking for volumes starting with the string *
* RELEVANT TO THE SYSTEM PASSED TO US...
***********************************************************************

SCANVOLS DS 0H

WTO2 WTO 'CLEANUP: SEQUENTIAL FILE CLEANUP STARTING...'

USING UCBBOB,R4    ADDRESSABILITY TO UCB
LA R4,UCBAREA      LOCATE UCB WORKAREA
XC UCBWORK,UCBWORK +INITIALIZE UCBSCAN WORKAREA

UCBLOOP DS 0H

* 
UCBSCAN COPY,
   WORKAREA=UCBWORK,
   UCBAREA=UCBAREA,
   DCEAREA=NONE,
   DCELEN=0,
   VOLSER=NONE,
   DEVN=0,
   DYNAMIC=YES,
   RANGE=ALL,
   NONBASE=NO,
   DEVCLASS=DASD,
   DEVCID=0,
   IOCTOKEN=NONE,
   LINKAGE=SYSTEM,
   PLISTVER=MAX

* 
LTR R15,R15        GOT UCB OK?
BZ UCBCHECK       YES..
C R15,=F'4'       END OF UCBs?
BE ENDUCBS       YES..TIDY UP
B BADCALL        NO...ERROR

UCBCHECK DS 0H

TM UCBSTAT,ONLINE IS IT ONLINE?
BNO UCBLOOP      NO...GET NEXT ONE
CLC UCBVOLI(6),=6X'00' REAL DASD?
BE UCBLOOP      NO...GET NEXT ONE
CLC UCBVOLI(5),VOLUME IS IT A VOLUME WE WANT?
BNE UCBLOOP     NO...GET NEXT ONE

***********************************************************************
* ANY FURTHER VOLUME NAME FILTERING CAN BE DONE HERE... *

* CLC   UCBVOLI(6),=C'SY2SØ1' IS IT SY2SØ1? *
  BE    UCBLOOP YES..IGNORE IT *

* NOW WE HAVE ONE OF THE VOLUMES WE WANT TO CHECK - OBTAIN THE *
  *REAL* UCB ADDRESS BY CALLING THE "UCBLOOK" MACRO... *

* MODESET KEY=ZERO,MODE=SUP SUPERVISOR STATE FOR "UCBLOOK"
* UCBLOOK UCBVOLI,=C'SY2SØ1' USE VOLID FROM UCB COPY
  UCBPTR=UCBADDR. SAVE UCB ADDRESS IN HERE
  LOC=ANY. UCB CAN BE ANYWHERE
  NOPIN. DON'T PIN UCB
  RANGE=ALL. *ALL* UCBS
  DEVCCLASS=DASD. MAKE SURE ITS A DISK!
  DYNAMIC=NO CHECK DYNAMIC UCBS
  ST R15,SAVER15 STASH THE RC
* MODESET KEY=NZERO,MODE=PROB BACK TO NORMAL
* L R15,SAVER15 RELOAD RC
  LTR R15,R15 OK?
  BNZ LOOKERR NO...PANIC...

* NOW WE HAVE THE ADDRESS OF THE UCB OF THE VOLUME TO CHECK - ISSUE A *
  * CVAFFILT 'ACCESS=READ' REQUEST TO SEE IF THERE ARE ANY DATASETS ON *
  * THE VOLUME THAT WE MIGHT BE INTERESTED IN... *

* CVPL   CVAFFILT ACCESS=READ,
  FCL=FCLDEF, X
  BUFLIST=BFLHDEF, X
  UCB=UCBADDR
*
  LTR R15,R15 OK?
  BNZ CVAFFILT NO...PANIC...
  LA R1,FCLDEF R1 -> FCL HEADER
  USING FCLMAP,R1 TEMP ADDRESSABILITY TO FCL
  CLC FCLDSCBR,=H'Ø' ANY DSCB'S READ?
  BE UCBLOOP NO...GET NEXT UCB
  LH R2,FCLDSCBR YES..GET COUNT
  DROP R1
  LA R3,DSCBDEF R3 -> DSCB BUFFERS
  USING DSCBMAP,R3 ADDRESSABILITY TO DSCB'S

* NOW WE HAVE A DSCB OR DSCBS ON THIS VOLUME WHICH MATCH WHAT WE ARE *
  * LOOKING FOR. THE ACTIONS WE WILL TAKE ARE: *
  * ---------FOR MODE = 'NORMAL'--------- *
  * - IF EMPTY THEN DELETE IT *
* - IF NOT EMPTY THEN RENAME IT, APPENDING A DATE/TIME STAMP *
* - IF LG1 THEN START 'OFFLDLG1' *
* - IF LG2 THEN START 'OFFLDLG2' *
* - IF LG3 THEN START 'OFFLDLG3' *
**-------------FOR MODE = 'DELETE'-------------------**
* - IF EMPTY THEN DELETE IT *
* - IF LG2 OR LG3 THEN DELETE IT *
* - (LG1) RENAME WITH DATE/TIME STAMP AND START 'OFFLDLG1' *

***********************************************************************
CLC PREVVOL(6),UCBVOLI VOLID SAME AS LAST ONE?
BE DSCBLOOP YES..DON'T MOVE TO PRINT LINE
MVC DETVOL(6),UCBVOLI NO...MOVE VOLID TO PRINT LINE
MVC PREVVOL(6),UCBVOLI SAVE NEW VOLID
MVC DETAIL(1),SPACE MAYBE SINGLE OR DOUBLE SPACE
MVI SPACE,C'Ø'
FORCE DOUBLE SPACE BETWEEN VOLS

DSCBLOOP DS ØH
MVC DETDSN(44),DS1DSNAM MOVE DSNAME TO PRINT LINE
CLC DS1DSNAM+11(3),LG1 IS IT AN LG1?
BE DSNOK YES..OK
CLC DS1DSNAM+11(3),LG2 IS IT AN LG2?
BE DSNOK YES..OK
CLC DS1DSNAM+11(3),LG3 IS IT AN LG3?
BE DSNOK YES..OK
MVC DETACT,UNKNMSG NO...MOVE 'UNKNOWN TYPE' TO PRT
BAL R9,PRTLINE PRINT 'UNKNOWN TYPE'...
B NEXTDSCB ...AND SKIP IT

***********************************************************************
* WHEN WE GET HERE WE HAVE A DSCB FOR A DATASET THAT MAY BE IN ONE OF *
* TWO FORMS; 'NORMAL' OR 'RENAMED', EG: *
* * * *
* * PROD.SYST1.LG1.B0000001 *
* OR PROD.SYST1.LG1.D920706.T120855 *
* * *
* IF THE SECOND FORM IS THE CASE THEN ONE OF THE 'OFFLDLGx' JOBS (THE *
* MIGRATE TO CART) HAS FAILED/BEEN CANCELLED. IF THIS IS THE CASE WE *
* WILL TRY TO RESTART THE MIGRATE PROC, BUT MUST ALSO HONOUR 'NORMAL' *
* OR 'DELETE' MODE PROCESSING. *
***********************************************************************

DSNOK DS ØH
AP COUNT,P1 INCREMENT COUNT
MVC DETACT,DELETEDE SET POSSIBLE DELETE MESSAGE
CLC DS1LSTAR(3),-X'000000' LAST-USED-TRACK = Ø?
BE DELETEIT YES..DELETE IT
CLI SWITCH,NORM ARE WE IN NORMAL MODE?
BE RENAMEIT YES..RENAME IT *
* NO..DELETE MODE, IF LG2 OR *
* LG3 THEN DELETE IT...
CLC DS1DSNAM+11(3),LG1 IS IT 'LG1'?
BE RENAMEIT YES..RENAME IT - ELSE DELETE...
MVC DETACT,DELETEDM SET POSSIBLE DELETE MESSAGE

***********************************************************************
* DELETE PROCESSING: NOTE THAT THIS IS A COMBINATION OF ACTIONS, VIZ *
  * SCRATCH FROM THE VTOC *
  * AND UN-CATALOGING *

*****************************************************************************
DELETEIT DS ØH
MVC CAMVOL(6),UCBVOLI SET VOLID IN CAMLST PARMS
MVC OLDNAME(44),DS1DSNAM SET UP DSNAM FOR DELETE
XC CAMSTAT,CAMSTAT SET CAMLST STATUS BYTES = Ø
XR RØ,RØ RØ MUST BE Ø FOR SCRATCH

* SCRATCH SCRLST SCRATCH THE DATASET

* LTR R15,R15 SCRATCH OK?
BZ DELETEOK YES..
MVC DETACT,NOTDEL NO...SHOW NOT DELETED
STH R15,HWORD SAVE RETC
BAL R9,CONVR15 CONVERT RETC TO PRINTABLE HEX
MVC DETACT+21(2),UNPKFLD MOVE RETC TO DETAIL LINE
MVC HWORD(2),CAMSTAT SAVE SCRATCH STATUS
BAL R9,CONVR15 CONVERT STAT TO PRINTABLE HEX
MVC DETACT+36(2),=C'Ø7' STATUS WAS 7? (IE 'IN USE')
BNE PRINTDEL NO...DON'T WORRY
MVC DETACT+4Ø(8),INUSE YES...SHOW IT WAS IN USE
B PRINTDEL GO AND PRINT DETAILS

*****************************************************************************
* DELETE WENT OK - NOW UNCATALOGUE IT... *
*****************************************************************************
DELETEOK DS ØH

* CATALOG UNCATLG UNCATELogue IT AS WELL

* LTR R15,R15 OK?
BZ PRINTDEL YES..
MVC DETACT,NOTUNCAT NO...SHOW DELETED/BUT NOT UNCAT
STH R15,HWORD SAVE RETC
BAL R9,CONVR15 CONVERT RETC TO PRINTABLE HEX
MVC DETACT+37(2),UNPKFLD MOVE RETC TO DETAIL LINE
PRINTDEL DS ØH
BAL R9,PRTLINE PRINT THE DETAIL LINE
B NEXTDSOB CHECK NEXT DSCB ON THIS VOLUME

*****************************************************************************
* RENAME PROCESSING: NOTE THAT THIS IS A COMBINATION OF ACTIONS, VIZ *
  * RENAME IN THE VTOC *
  * AND UN-CATALOGUING OF THE OLD DATASET *
  * AND CATALOGUING OF THE NEW DATASET *

*****************************************************************************
RENAMEIT DS ØH
CLI DS1DSNAM+15,C'D' ALREADY BEEN RENAMED?
BNE RENAMEØ5 NO...GO AND START OFFLDLGX
MVC NEWNAME(3Ø),DS1DSNAM YES...SAVE DSNAM FOR LATER

MVC TIMESTAMP(6),DS1DSNAM+24 ALSO SAVE TIMESTAMP IN CMD
B STARTIT GO AND START OFFLDL GX
RENAMES DS ØH
BAL R9,GETTIME GET DATE AND TIME STAMP...
* DATE AND TIME STAMPS ARE NOW
* SET UP - DO THE RENAME
* MVC OLDNAME,DS1DSNAM MOVE CURRENT DSNAME TO PARMS
MVC NEWNAME(15),DS1DSNAM EG 'PROD.SYST1.LG2.'
MVC NEWNAME+15(15),DATETIME ADD DATE/TIME STAMP TO THE END
MVC CAMVOL(6),UCBVOLI SET VOLID IN CAMLST PARMS
XC CAMSTAT,CAMSTAT SET CAMLST STATUS BYTES = Ø
XR RØ,RØ SET RØ = Ø
*
RENAME RENAMLST ISSUE THE RENAME REQUEST
*
LTR R15,R15 RENAME OK?
BZ RENAMEOK YES..
MVC DETACT,NOTRENAM NO...SHOW NOT RENAMED
STH R15,HWORD SAVE RETC
BAL R9,CONVR15 CONVERT RETC TO PRINTABLE HEX
MVC DETACT+21(2),UNPKFLD MOVE RETC TO DETAIL LINE
MVC HWORD(2),CAMSTAT SAVE RENAME STATUS
BAL R9,CONVR15 CONVERT STAT TO PRINTABLE HEX
MVC DETACT+36(2),UNPKFLD MOVE STAT TO DETAIL LINE
CLC DETACT+36(2),=C'Ø7' STATUS WAS 7? (IE 'IN USE')
BNE NORNMMSG NO...DON'T WORRY
MVC DETACT+4Ø(8),INUSE YES...SHOW IT WAS IN USE
NORNMMSG DS ØH
BAL R9,PRTL INE PRINT THE DETAIL LINE
B NEXTDSCB CHECK NEXT DSCB ON THIS VOLUME
RENAMES DS ØH
MVC DETACT,RENAMED SHOW DATASET RENAMED
MVC DETACT+11(3Ø),NEWNAME SHOW NEW NAME
BAL R9,PRTL INE PRINT THE DETAIL LINE
UNCATOLD DS ØH
*
CATALOG UNCATLG UNCATALOG THE OLD ONE
*
LTR R15,R15 OK?
BZ NOWRECAT YES..GO AND CATALOG NEW ONE
MVC DETACT,UNCATBAD NO...SHOW NOT UNCATALOGUED
STH R15,HWORD SAVE RETC
BAL R9,CONVR15 CONVERT RETC TO PRINTABLE HEX
MVC DETACT+25(2),UNPKFLD MOVE RETC TO DETAIL LINE
MVC DETACT+34(23),OLDNAME MOVE DSNAME TO DETAIL LINE
BAL R9,PRTL INE PRINT THE DETAIL LINE
NOWRECAT DS ØH
*
CATALOG CATALOG CATALOG THE NEW ONE
*
LTR R15,R15 OK?
BZ STARTIT                  YES..GO AND START OFFLDLGX
MVC DETACT,CATLGBAD        NO...SHOW NOT CATALOGUED
STH R15,HWORD              SAVE RETC
BAL R9,CONVR15             CONVERT RETC TO PRINTABLE HEX
MVC DETACT+23(2),UNPKFLD   MOVE RETC TO DETAIL LINE
MVC DETACT+32(26),NEWNAME  MOVE DSNAME TO DETAIL LINE
BAL R9,PRTLINE             PRINT THE DETAIL LINE
STARTIT DS ØH
MVI CMDSTART+13,C'2'       SET UP TO START 'OFFLDLG2'
CLC DS1DSNAM+11(3),LG2     IS IT A 'LG2' FILE?
BE STARTIT5                YES..'OFFLDLG2' ALREADY SET UP
MVI CMDSTART+13,C'3'       SET UP TO START 'OFFLDLG3'
CLC DS1DSNAM+11(3),LG3     IS IT A 'LG3' FILE?
BE STARTIT5                YES..'OFFLDLG3' ALREADY SET UP
MVI CMDSTART+13,C'1'       NO...SET UP TO START 'OFFLDLG1'
STARTIT5 DS ØH
MVC CMDDSN(3Ø),NEWNAME     MOVE DSNAME TO PARMS
MVC CMDVOL(6),UCBVOLI      MOVE VOLID TO PARMS
MVC CMDTIM(6),TIMESTAMP    MOVE TIMESTAMP TO PARMS
MODESET KEY=ZERO,MODE=SUP
XR RØ,RØ                   SET RØ = Ø
LA R1,CMDSTART             R1 -> START COMMAND
SVC 34                     ISSUE START COMMAND
MODESET KEY=NZERO,MODE=PROB
MVC DETACT(58),CMDMSG      SET UP MESSAGE
MVC DETACT+1(10),CMDSTART+4 PRINT THE DETAIL LINE
B NEXTDSCB                 CHECK NEXT DSCB ON THIS VOLUME
NEXTDSCB DS ØH
MVC DETVOL(6),=6C' '       CLEAR OUT VOLID IN PRINT LINE
LA R3,DSCBSIZE(R3)         POINT TO NEXT DSCB ENTRY
BCT R2,DSCBLOOP            LOOP THROUGH ALL DSCB'S
LA R1,FCLDEF               R1 -> FCL HEADER
USING FCLMAP,R1            TEMP ADDRESSABILITY TO FCL
MVC FCLDSCBR,=H'Ø'        RESET COUNT AT END OF DSCBS
B UCBLOOP                  CHECK NEXT UCB
DROP R1
**********************************************************************
* NO MORE UCB'S - TIDY UP AND END...                                  *
**********************************************************************
ENDUCBS DS ØH
CP COUNT,PØ                ANY DATASETS FOUND?
BNZ CLSNOP                 YES..CLOSE SYSPRINT IF REQD
MVC DETDSN(44),NONEFND     NO...SHOW THEM NONE FOUND
BAL R9,PRTLINE             PRINT THE DETAIL LINE
MVC RETC,=F'4'             SET RC=4 FOR NO DATASETS FOUND
CLSNOP NOP AFTERCLS        MAYBE BRANCH
CLOSE SYSPRINT             CLOSE SYSPRINT
AFTERCLS DS ØH
RETURN DS ØH
WTO3 WTO 'CLEANUP: SEQUENTIAL FILE CLEANUP ENDING...'

L R15,RETC  LOAD RETURN CODE
PR ,  RESTORE CALLER DATA, RETURN
***********************************************************************
* INVALID OR NO PARMS PASSED...  *
***********************************************************************
BADPARM DS ØH
WTO 'CLEANUP: INVALID SYSTEM OR MODE PASSED...',ROUTE=11
MVC RETC=F'8'  SET RC=8
B RETURN
***********************************************************************
* BAD RETURN CODE FROM 'UCBSCAN'...  *
***********************************************************************
BADCALL DS ØH
LR R10,R15  SAVE RETC
WTO 'CLEANUP: BAD CALL TO "UCBSCAN"...',ROUTE=11
LR R15,R10  RELOAD RETC
DS F  BANG
***********************************************************************
* BAD RETURN CODE FROM 'UCBLOOK'...  *
***********************************************************************
LOOKERR DS ØH
LR R10,R15  SAVE RETC
LR R8,R0  SAVE REASON
WTO 'CLEANUP: BAD CALL TO "UCBLOOK"...',ROUTE=11
LR R0,R8  RELOAD REASON
LR R15,R10  RELOAD RETC
DS F  BANG
***********************************************************************
* ERROR IN 'CVAFFILT' MACRO...  *
***********************************************************************
CVAFERR DS ØH
LA R1,CVPL+4  TEMP ADDRESSABILITY TO CVAF
USING CVPLMAP,R1
STH R15,HWORD  SAVE RETC
BAL R9,CONVR15  CONVERT RETC TO PRINTABLE HEX
MVC CVAFR15(2),UNPKFLD  MOVE RETC TO DETAIL LINE
MVI HWROD,X'O0'  ZEROISE 1ST BYTE
MVC HWROD+1(1),CVSTAT  SAVE CVSTAT STATUS
BAL R9,CONVR15  CONVERT STAT TO PRINTABLE HEX
MVC CVAFSTAT(2),UNPKFLD  MOVE STAT TO DETAIL LINE
MVC DETACT,CVAFMSG  SHOW CVAFFILT ERROR
BAL R9,PRTLINE  PRINT THE DETAIL LINE
DS F  BANG...
DROP R1
***********************************************************************
* SUBROUTINE: PRINT A DETAIL LINE (IF SYSPRINT OPENED OK)...  *
***********************************************************************
PRTLINE DS ØH
PRTNOP NOP AFTERPRT  MAYBE BRANCH
PUT SYSPRINT,DETAIL  PRINT DETAIL LINE
MVI  DETAIL.C’ ’ CLEAR OUT DETAIL LINE
MVC DETAIL+1(L’DETAIL-1),DETAIL
AFTERPRT DS ØH
BR R9 RETURN FROM ROUTINE
***********************************************************************
* SUBROUTINE: CONVERT RETURN CODES, ETC INTO PRINTABLE HEX...         *
***********************************************************************
CONVR15 DS ØH
UNPK UNPKFLD(3),HWORD+1(2) UNPACK RETC + 1 BYTE
TR UNPKFLD(2),TRTAB-24Ø XLATE TO PRINTABLE HEX
BR R9 RETURN FROM ROUTINE
***********************************************************************
* SUBROUTINE: GENERATE DATE AND TIME STAMPS FOR USE IN 'RENAME'...    *
***********************************************************************
GETTIME DS ØH
STIMER WAIT,BINTVL=BIN1 ENSURE TIMESTAMP ARE DIFFERENT
TIME . GET DATE/TIME
ST R0,TIME SAVE TIME
ST R1,DATE SAVE DATE
UNPK UNPKFLD(7),TIME UNPACK TIME
MVC TIMESTAMP(6),UNPKFLD MOVE HHMMSS TO TIMESTAMP
ZAP PL2X(3),PØ INITIALIZE WORKAREA
MVO PL2,DATE(2) GET YEAR NUMBER
UNPK YEAR,PL2 UNPACK YEAR NUMBER
OI YEAR+2,X’FØ’ SET CORRECT SIGN
DP PL2X(3),P4 CHECK FOR LEAP YEAR
CP PL2+1(1),PØ ZERO REMAINDER? (= LEAP YEAR)
BNE NOLEAP NO...
MVC FEB(4),=F’29’ YES..SET LEAP YEAR DAYS
NOLEAP DS ØH
ZAP DWORD,DATE+2(2) GET DAY IN YEAR (DDDF)
CVB R1,DWORD GET DAYS IN BINARY
STH R1,DATE+2 PUT BACK IN DATE
LA R1,MONTHS POINT TO MONTHS TABLE
XR R7,R7
LA R15,12 LOOP COUNT
MTHLOOP DS ØH
A R7,Ø(R1) ACCUMULATE DAYS SO FAR
CH R7,DATE+2 MORE THAN OUR NO OF DAYS?
BNL MTHOK YES..GOOD
LA R1,8(R1) NO...POINT TO NEXT MONTH
BCT R15,MTHLOOP KEEP LOOKING
WTO5 WTO ’CLEANUP: LOGIC ERROR...',ROUTECD=11
DS F
MTHOK DS ØH
S R7,Ø(R1) GO BACK TO PREV MONTH
LH R8,DATE+2 GET DAYS
SR R8,R7 CALC DAY IN MONTH
CVD R8,DWORD MAKE IT PRINTABLE
UNPK DWORD(4),DWORD+6(2)
OI DWORD+3,X’FØ’ SET CORRECT SIGN
MVC DATDD(2),DWORD+2  MOVE DD
MVC DATMM(2),4(R1)  MOVE MM
MVC DATYY(2),YEAR+1  MOVE YY
BR  R9  RETURN FROM ROUTINE
EJECT

*---------------------------------------------------------------*

LTORG  LITERAL POOL
*
DSNAME  DC  C'PROD.SYSTX.*.**'  EG 'PROD.SYST1.LG2.B0000001'
DSNAMELN DC AL1(*-DSNAME)
SYSTMONE DC CL8'SYSTMONE'
SYSTMTWO DC CL8'SYSTMTWO'
VOLUME  DC CL5'SYXSØ'
PREVVOL  DC CL6'
COUNT  DC PL2'0'
PØ  DC PL1'0'
P1  DC PL1'1'
P4  DC PL1'4'
PL2X  DS P
PL2  DS PL2
YEAR  DS PL3
SWITCH  DC X'00'
NORM  EQU X'10'
DEL  EQU X'20'
NORMAL  DC CL7','NORMAL'
DELETE  DC CL7','DELETE'
LG1  DC CL3'LG1'
LG2  DC CL3'LG2'
LG3  DC CL3'LG3'
UNPKFLD  DS XL7
SPACE  DC C' '
TRTAB  DC C'0123456789ABCDEF'
RETC  DC F'0'
SAVER15  DC F'0'
TIOTADDR  DC F'0'
UCBADDR  DC F'0'
HWORD  DC H'0'
DWORD  DC D'0'
DATE  DC F'0'
TIME  DC F'0'
BIN1  DC F'100'  1 SEC INTERVAL
*
DATETIME  DS ØCL15  DYYMMDD.THHMMSS
DC C'D'
DATSTAMP  DS ØCL6
DATYY  DS CL2
DATMM  DS CL2
DATDD  DS CL2
DC C'.T'
TIMESTAMP  DS CL6
*-----------------------------------------------*
* PARAMETER LISTS FOR 'SCRATCH', 'RENAME', 'UNCATALOG' AND 'CATALOG' *

*--------------------------------------------------------------------*

UNCATLG  CAMLST UNCAT,OLDNAME
CATALOG  CAMLST CAT,NEWNAME,,VOLLIST
SCRLST   CAMLST SCRATCH,OLDNAME,,VOLLIST
RENMALST CAMLST RENAME,OLDNAME,NEWNAME,,VOLLIST

VOLLIST  DC  H'1'                     ONE VOLUME
DC  X'3030200F'  NOTE: 339Ø DEVICE CODE
CAMVOL   DC  CL6' '                   VOLID
CAMSTAT  DC  H'0'                     RENAME STATUS CODE
OLDNAME  DC  CL44' '                  OLDNAME
NEWNAME  DC  CL44' '                  NEWNAME

*--------------------------------------------------------------------*

* START COMMAND (ISSUED BY SVC 34)...                                 *

CMDSTART DC  Y(CMDLEN),Y(Ø)
CMDDSN  DC  CL3Ø' '                  CMMDSN
DC  C''',VOLID=''             VOLUME
CMDVOL   DC  CL6' '                  VOLUME
DC  C'',TIM=T'               TIME
CMDTIM   DC  CL6' '                  TIME
CMDLEN   EQU  *-CMDSTART

*--------------------------------------------------------------------*

* SYSPRINT DCB...                                                     *

SYSPRINT DCB   DDNAME=SYSPRINT,                                        X
DSORG=PS,                                               X
MACRF=PM,                                               X
LRECL=121,                                              X
BLKSIZE=121,                                            X
RECFM=FBA

*--------------------------------------------------------------------*

* FIELDS FOR DATE/TIME CALCULATIONS...                                *

MONTHS   DC    F'31',CL2'01'
FEB      DC    F'28',CL2'02'
DC    F'31',CL2'03'
DC    F'30',CL2'04'
DC    F'31',CL2'05'
DC    F'30',CL2'06'
DC    F'31',CL2'07'
DC    F'31',CL2'08'
DC    F'30',CL2'09'
DC    F'31',CL2'10'
DC    F'30',CL2'11'
DC    F'31',CL2'12'

*--------------------------------------------------------------------*

* PRINT LINES...                                                      *

HEAD1    DS  ØCL121
DC CL50'1MODE=' SEQUENTIAL ' 
DC CL71'FILE CLEANUP PROGRAM SYSTEM='
* 
HEAD2 DS 0CL121 
DC CL50'------------------------------' 
DC CL76'------------------------------' 
* 
HEAD3 DS 0CL121 
DC CL50'-VOLUME-' DATASET--------' 
DC CL50'------------------------' 
DC CL21'----------' 
* 
DETAIL DS 0CL121 
DC CL4' ' 
DETVOL DC CL6' ' 
DC CL5' ' 
DETDSN DC CL44' ' 
DC CL4' ' 
DETACT DC CL58' ' 
* 
DELETEDC DC CL58'DELETED (EMPTY)' 
DELETEDM DC CL58'DELETED (DELETE MODE)' 
NOTDEL DC CL58'NOT DELETED - RC = X"XX" STATUS = X"XX"' 
NOTUNCAT DC CL58'NOT DELETED BUT NOT UNCATALOGED - RC = X"XX"' 
RENAME DC CL58'RENAMED TO ' 
NOTRENAME DC CL58'NOT RENAMED - RC = X"XX" STATUS = X"XX"' 
NONEFND DC CL58'*** NO DATASETS LOCATED' 
CVAFMSG DS 0CL58 
DC CL35'ERROR IN "CVAFFILT" MACRO - RC = X'' 
CVAFR15 DC CL2' ' 
DC CL13'' CVSTAT = X'' 
CVAFSTAT DC CL2' ' 
DC CL6'' 
UNCATBAD DC CL58'UNCATALOG FAILED (RC = X"XX") FOR ' 
CATLGBAD DC CL58'CATALOG FAILED (RC = X"XX") FOR ' 
CMDMSG DC CL58'COMMAND ISSUED: ' 
UNKNMSG DC CL58'DATASET TYPE NOT LG1/LG2/LG3...IGNORED' 
INUSE DC CL8'(IN USE)' 
*--------------------------------------------------------------------* 
* UCB SCAN ROUTINE PARAMETERS...                                      * 
*--------------------------------------------------------------------* 
DS 0F 
UCBAREA DS XL48 HOLDS UCB COMMON & DEV SEGS 
UCBWORK DS XL100 UCBSCAN WORKAREA 
*--------------------------------------------------------------------* 
* SPACE ALLOCATION FOR CVPL, FCL, BFL AND DSCB BUFFERS...             * 
*--------------------------------------------------------------------* 
FCLDEF DS (FCLHDLEN+FCLDSNEL)X FCL HEADER AND 1 FCL ELEMENT 
FCLSIZE EQU *-FCLDEF 
*-------------DEFINE A CVAF BUFFER LIST WITH 'N' BUFFER LIST ELEMENTS------ 
BFLHDEF DS (BFLHLN)X BUFFER LIST HEADER (BFLH) 

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
BFLEDEF DS (BUFFNUM*BFLELN)X 'N' BUFFER LIST ELEMENTS (BFLE'S)
BFLSIZE EQU *-BFLHDEF
*----------------------DEFINE 'N' FULL DSCB BUFFERS-------------------------------
DSCBDEF DS (BUFFNUM*DSCBSIZE)X
*--------------------------------------------------------------------------------
* REGISTERS EQUATES, ETC...
*--------------------------------------------------------------------------------
BUFFNUM EQU 4Ø 4Ø BUFFER LIST ELEMENTS AND BUFFERS
DASDTYPE EQU X'2Ø'
ONLINE EQU X'8Ø'
RØ EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R1Ø EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
CVPLMAP ICVAFPL CVPLFSA=YES
FCLMAP ICVFCL
BFLMAP ICVAFBFL
IEFUCBOB , PRINT NOGEN
CVT DSECT=YES
DSCBMAP DSECT
IECSDSL1 (1) FORMAT1 DSCB MAPPING TO BEF BUFFSIZE
DSCBSIZE EQU *-IECSDSL1
*
END , END OF PROGRAM

Grant Carson
Systems Programmer (UK) © Xephon 2001
INTRODUCTION
There are many reasons why one might want to view a map of the active MVS virtual storage layout – for example, to check on the size of the private user region, to confirm the size of the LPA (Link Pack Area) before applying maintenance, or to tune the sizes of (E)CSA ((Extended) Common Service Area) and (E)SQA ((Extended) System Queue Area), etc. Presently OS/390 offers no easy way to do this.

VSMAP, the ISPF dialog described in this article, is written in REXX and uses a single ISPF panel named VSMWIN. The REXX can run in ISPF, native TSO, batch TSO, and batch REXX. The ISPF dialog is executed by the single command ‘VSMAP’. In ISPF, the result is a display similar to the one shown in Figure 1.

<table>
<thead>
<tr>
<th>MVS1 Virtual Storage Map - 10 Oct 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Address +------------------------+ 2G</td>
</tr>
<tr>
<td>06400000</td>
</tr>
<tr>
<td>0553D000</td>
</tr>
<tr>
<td>0225E000</td>
</tr>
<tr>
<td>01864000</td>
</tr>
<tr>
<td>01000000</td>
</tr>
<tr>
<td>00FCC000</td>
</tr>
<tr>
<td>00EC4000</td>
</tr>
<tr>
<td>00CCD000</td>
</tr>
<tr>
<td>00800000</td>
</tr>
<tr>
<td>00050000</td>
</tr>
<tr>
<td>+------------------------+ 16M + 0</td>
</tr>
</tbody>
</table>

*Figure 1: ISPF panel display - VSMWIN*
By default under ISPF, the panel will be displayed in a pop-up window, unless the screen size is too small (24 lines or fewer) in which case the map will be displayed in the full ISPF screen.

Although the VSMAP command is designed to run under ISPF and display an ISPF panel, it has been written to allow it to run under native TSO and in batch as well. In these cases the virtual storage layout will be presented in a simple line display, as shown in Figure 2. This is automatic depending on the environment in which the VSMAP REXX EXEC is executed but the line display can be requested under ISPF by using an optional parameter (see below).

The line display is also used if the display of the panel should fail for any reason. The VSMAP command does not use LIBDEF, but assumes that the panel is available in the active ISPPLIB concatenation.

![Figure 2: TSO line display](image)

### PARAMETERS

No parameters are required to run the VSMAP command, but there are several optional parameters that are used to indicate the type of display required.

- (none) – display is determined by active environment
• POPUP – to display in pop-up window
• FULL – to display in full-screen panel
• TSO – to show TSO line display, even under ISPF.

The parameters are mutually exclusive and if multiple parameters are entered, the first will be honoured (if the environment permits) while the rest are ignored. The parameters can be abbreviated to a single character and an invalid parameter will be ignored.

ABOUT THE CODE
The VSMAP REXX EXEC uses the REXX STORAGE function to retrieve information from the CVT (Communications Vector Table) and GDA (Global Data Area). The GDA control block is used by VSM (Virtual Storage Manager) to contain information about system-related virtual storage and to anchor SQA and CSA queues. The CVT provides the address of the GDA and together these control blocks provide all the data required to build the map. They include the start address and size of most of the virtual storage areas and these figures are used to calculate the addresses and sizes of the remaining areas.

The REXX STORAGE function is described as a TSO/E external function but, unlike some other TSO/E external functions, you can use the STORAGE function in REXX EXECs that run in any address space, TSO/E and non-TSO/E.

REXX does not perform hexadecimal arithmetic and so each value must be converted from hex into decimal to perform the calculation, and then back into hex again. This includes control block offsets.

The code also demonstrates the use of internal functions for repetitive actions. In VSMAP there are three – KB, MB, and THOU. KB – and MB are used for converting byte values into kilobytes and megabytes respectively, while THOU is used to insert a comma separator in large values to improve readability. Note that these calculations use the ‘traditional’ multiplier of 1024 and not the 1000 that is commonly used when describing DASD storage space.

The VSMWIN panel uses dashes and vertical bars to create a grid effect. The use of these special characters is a common cause of
translation problems when moving between the PC ASCII environment and the mainframe and EBCDIC, but any problems should be easy to recognize and remedy. The specification of panel attribute characters is also an area prone to translation problems. The VSMWIN panel uses four attribute characters that are defined in the panel so, when downloading the dialog, be prepared to review the panel before running for the first time.

The plus character (+) is also displayed in the VSMWIN panel. As this is one of the ISPF default panel attribute characters, the DEFAULT keyword of the ‘)ATTR’ header statement is used in VSMWIN to redefine the default attribute characters without the plus sign.

FUTURE ENHANCEMENTS

The VSMAP dialog uses meaningful variable names and has a simple structure, which should make it easy to modify.

It could be enhanced to provide further information, which is available in system control blocks. For example, the LPA and extended LPA sizes could be subdivided into MLPA, FLPA, and PLPA (modified, fixed, and pageable LPA) with data from the CVT. Again using information from the CVT, the nucleus areas too can be split to show the read-only and read-write nucleus areas. And the display could incorporate the 4K PSA (Prefixed Save Area) and 16K system region.

REFERENCE

Details of the structure of the CVT and GDA and descriptions and offsets of the fields used by VSMAP can be found in manual OS/390 MVS Data Areas, Volume 1 (ABEP - DALT) and Volume 2 (DCCB – ITTCTE). A virtual storage overview can be found in the OS/390 MVS Initialization and Tuning Guide describing each of the different areas and their function. Information about REXX functions can be found in the OS/390 TSO/E REXX Reference manual.
VSMAPREXX EXEC

/* +----------------------------- REXX ----------------------------+ */
/* | DOC: VSMAP - REXX command to display Virtual Storage Map      | */
/* |                                                               | */
/* |      The type of display is determined by an optional         | */
/* |      parameter as follows:                                    | */
/* |         (none)  to display in ISPF panel                      | */
/* |         POPUP   to display in pop-up window                   | */
/* |         FULL    to display in full-screen panel               | */
/* |         TSO     to show TSO line display, even under ISPF,    | */
/* |                 also used if the panel or window fails         | */
/* |                                                               | */
/* |      This dialog uses panel VSMWIN from active ISPPLIB.       | */
/* |                                                               | */
/* +---------------------------------------------------------------+ */

parse upper arg prm1 rest
if prm1 ¬= '' then call VERPRM prm1        /* verify parameters */
if rest ¬= '' then say rest 'ignored'

numeric digits(15)                         /* digits to 15 */
sysname = mvsvar('SYSNAME')
today = date()

gda = d2x(c2d(storage(16,4))+560)          /* address gda */
gda = d2x(c2d(storage(gda,4)))

csastart = d2x(x2d(gda)+108)               /* csa start address */
csastart = c2d(storage(csastart,4))
csasize = d2x(x2d(gda)+112)                /* csa size */
csasize = c2d(storage(csasize,4))

ecsastrt = d2x(x2d(gda)+124)               /* ecsa start address */
cecsastrt = c2d(storage(ecsastrt,4))
cecsasize = d2x(x2d(gda)+128)              /* ecsa size */
cecsasize = c2d(storage(eccsasize,4))

sqastart = d2x(x2d(gda)+144)               /* sqa start address */
sqastart = c2d(storage(sqastart,4))
sqasize = d2x(x2d(gda)+148)                /* sqa size */
sqasize = c2d(storage(sqasize,4))

esqastart = d2x(x2d(gda)+152)              /* esqa start address */
esqastart = c2d(storage(esqastart,4))
esqasize = d2x(x2d(gda)+156)               /* esqa size */
esqasize = c2d(storage(esqasize,4))

pvtstart = d2x(x2d(gda)+160)               /* pvt start address */
pvtstart = c2d(storage(pvtstart,4))
pvtsize = d2x(x2d(gda)+164)                /* pvt size */
\begin{verbatim}

pvtsize = c2d(storage(pvtsize,4))  /* inc 4k PSA + 16K syste*/

epvtstart = d2x(x2d(gda)+168)  /* epvt start address */
epvtstart = c2d(storage(epvtstart,4))
epvtsize = d2x(x2d(gda)+172)  /* epvt size */
epvtsize = c2d(storage(epvtsize,4))

lpastart = d2x(c2d(storage(l6,4))+361)  /* lpa start address */
lpastart = c2d(storage(lpastart,4))

pstart = (4+16)*1024  /* private start address */
lpasize = (sqastart - lpastart)
psize = (csastart - pstart)

nucstart = sqastart + sqasize

M16 = x2d('FFFFFF')  /* 16 meg line */
nsize = (M16 - nucstart)

elpastrt = (esqastrt + esqasize)
elpasize = (ecsastrt - elpastrt)

enucstrt = x2d(1000000)  /* enucleus start address*/
ensize = (esqastrt - enucstrt)

epstart = (ecsastrt + ecsasize)

top = x2d('7FFFFFFF')
epsize = (top - epstart)

/* addresses in hex */
epstart = right(d2x(epstart),8,0)
ecsastrt = right(d2x(ecsastrt),8,0)
elpastrt = right(d2x(elpastrt),8,0)
esqastrt = right(d2x(esqastrt),8,0)
enucstrt = right(d2x(enucstrt),8,0)
nucstart = right(d2x(nucstart),8,0)
sqastrt = right(d2x(sqastart),8,0)
lpastart = right(d2x(lpastart),8,0)

csastart = right(d2x(csastart),8,0)
pstart = right(d2x(pstart),8,0)

/* end-of-calculations */

csaKB = KB(csasize)
lpaKB = KB(lpasize)
sqaKB = KB(sqasize)

/* convert sizes to KB */
\end{verbatim}

pKB = KB(psize)
nKB = KB(nsize)

esqaKB = KB(esqasize)
elpaKB = KB(elpasize)
ecsaKB = KB(ecsasize)
enKB = KB(ensize)

/* convert sizes to MB */
csaMB = MB(csasize)
lpaMB = MB(lpasize)
sqaMB = MB(sqasize)
pMB = MB(psize)
nMB = MB(nsize)

esqaMB = MB(esqasize)
elpaMB = MB(elpasize)
ecsaMB = MB(ecsasize)
enMB = MB(ensize)
epMB = MB(epsize)  /* E-private always MB */
epsize = MB(epsize)  /* E-private always MB */

if sysvar(sysispf) = 'ACTIVE' & tso = 'Y' then
    call POPUP
else call DISPLAY
exit

/* +---------------------------------------------------------------+ */
/* |  Sub-routines                                                 | */
/* +---------------------------------------------------------------+ */
VERPRM:                                    /* verify parameters */
parse arg parm
select
    when parm = substr('POPUP',1,length(parm)) then POP = 'Y'
    when parm = substr('FULL',1,length(parm)) then FULL = 'Y'
    when parm = substr('TSO',1,length(parm)) then TSO = 'Y'
otherwise say parm 'invalid option - specify POPUP, FULL or TSO'
end
return

/* +---------------------------------------------------------------+ */
/* | ISP F pop-up window                                           | */
/* +---------------------------------------------------------------+ */
POPUP:
"ISPEXEC CONTROL ERRORS RETURN"
if FULL ≠ 'Y' then
  do
    if (POP = 'Y') | (sysvar('SYSLTERM') > 24),
      then "ISPEXEC ADDPOP"
  end
"ISPEXEC DISPLAY PANEL(VSMWIN)"
if rc > 8 then call DISPLAY
"ISPEXEC REMPOP"
return

MB: procedure          /* convert bytes to MB */
arg bytes
  x = format(bytes/1024/1024,,2)
  x = x 'M'
return x

KB: procedure          /* convert bytes to KB */
arg bytes
  x = format(bytes/1024,,0)
  if length(strip(x)) > 3 then x = thou(x)   /* ,000 separator needed */
  x = x 'K'
return x

THOU: procedure        /* insert commas in num */
arg bignum
  bignum = strip(bignum)
  cpos = 3         /* position for comma */
  do until length(bignum)-cpos <= 0
    if length(bignum)-cpos <= 0 then return bignum
    bignum = insert(',',bignum,length(bignum)-cpos)
    cpos = cpos + 4                        /* position of next comma*/
  end
return bignum

DISPLAY:
say sysname 'Virtual Storage Layout' || today
/* 'Start Address 2G' */
  line = ''
say line
  line = insert('Extended Private',line,5)
  line = insert(epsize,line,24)
  line = insert(epMB,line,35)
say line
/* epstart */
  line = ''
  line = insert('Extended CSA',line,5)
  line = insert(right(ecsaKB,9),line,24)
  line = insert(right(ecsaMB,9),line,35)
say line
/* ecsastrt */
  line = ''
  line = insert('Extended LPA',line,5)
  line = insert(right(elpaKB,9),line,24)
  line = insert(right(elpaMB,9),line,35)
say line
/* elpastrt */
  line = ''
  line = insert('Extended SQA',line,5)
  line = insert(right(esqaKB,9),line,24)
  line = insert(right(esqaMB,9),line,35)
say line
/* esqastrt */
  line = ''
  line = insert('Extended Nucleus',line,5)
  line = insert(right(enKB,9),line,24)
  line = insert(right(enMB,9),line,35)
say line
  line = ''
  line = insert(right('-',41,'-'),line,4)
say line '16M'
/* enucstrt '16M' */
  line = ''
  line = insert('Nucleus',line,5)
  line = insert(right(nKB,9),line,24)
  line = insert(right(nMB,9),line,35)
say line
/* nucstart */
  line = ''
  line = insert('SQA',line,5)
  line = insert(right(sqaKB,9),line,24)
  line = insert(right(sqaMB,9),line,35)
say line
/* sqastart */
  line = ''
  line = insert('LPA',line,5)
  line = insert(right(lpaKB,9),line,24)
  line = insert(right(lpaMB,9),line,35)
say line
/* lpastart */
  line = ''
  line = insert('CSA',line,5)
  line = insert(right(csaKB,9),line,24)
  line = insert(right(csaMB,9),line,35)
say line
/* csastart */
  line = ''
  line = insert('Private',line,5)
  line = insert(right(pKB,9),line,24)
  line = insert(right(pMB,9),line,35)
say line
/* pstart 'ØM' */
return

VSMWIN ISPF PANEL

)`ATTR DEFAULT(]%e_)
@ TYPE(TEXT) COLOR(TURQ) CAPS(OFF)
# TYPE(OUTPUT) COLOR(YELLOW) JUST(RIGHT)
! TYPE(OUTPUT) COLOR(WHITE) JUST(RIGHT)
$ TYPE(OUTPUT) COLOR(TURQ) CAPS(OFF) JUST(RIGHT)
)BODY WINDOW(66,21)
@ Start Address +-------------------------------------------+ 2G
@ $epstart @ Y Extended Private !epsize @ $epMB @ Y
@ @ Y-------------------------------------------Y-
@ $ecsastrt@ Y Extended CSA !ecsaKB @ $ecsaMB @ Y
@ @ Y-------------------------------------------Y-
@ $elpastrt@ Y Extended LPA !elpaKB @ $elpaMB @ Y
@ @ Y-------------------------------------------Y-
@ $esqastrt@ Y Extended SQA !esqaKB @ $esqaMB @ Y
@ @ Y-------------------------------------------Y-
@ $enucstrt@ Y Extended Nucleus !enKB @ $enMB @ Y
@ @ Y-------------------------------------------Y 16M
@ $nucstart@ Y Extended Nucleus !nKB @ $nMB @ Y
@ @ Y-------------------------------------------Y-
@ $sqastrt@ Y Extended SQA !sqaKB @ $sqaMB @ Y
@ @ Y-------------------------------------------Y-
@ $lpastart@ Y Extended LPA !lpakB @ $lpAM @ Y
@ @ Y-------------------------------------------Y-
@ $csastart@ Y Extended CSA !csaKB @ $csaMB @ Y
@ @ Y-------------------------------------------Y-
@ $pstart @ Y Extended Private !pKB @ $pMB @ Y
@ @ Y-------------------------------------------Y 0
)INIT
&ZWINTTL = '&sysname Virtual Storage Map - &today'
)PROC
)END
/* VSMWIN panel - used by VSMAP REXX EXEC */

Moira Hunter
Systems Programmer (UK) © Xephon 2001
INTRODUCTION

In general, MVS sites use VOLSER naming conventions to manage their disk farms. But the administrators have to manually maintain procedures to manage on-line/off-line status of their disks. These procedures are based on unit address rather than VOLSER.

The VARYDISK program is used to vary DASD devices off-line at IPL time by VOLSER pattern rather than normal unit address specification. This drastically reduces the possible errors that could result from out-of-date PARMLIB member administration.

When started, the program will read SYSIN cards that specify the VOLSER patterns of volumes that are to remain online. The program then scans through the on-line DASD units and attempts to match the VOLSER from a unit to any of the specified patterns. If a match is found, a message is issued stating that the volume is to remain ONLINE. If no match is found, a message is issued stating that the volume is to be varied off-line and its unit address is varied off-line after all DASD units have been processed. The following implementation issues apply.

VARYDISK STC

The VARYDISK started task JCL is as follows:

```
//VARYDISK PROC
//VARYDISK EXEC PGM=VARYDISK,PARM='MODE=VARY,WTO=ALL,PRINT=Y'
//STEPLIB DD DISP=SHR,DSN=LINK.SYSTEM.APF
//SYSPRINT DD DISP=(,CATLG),
//             DSN=SYS1.VARYDISK.SYSPRINT(+1),
//             SPACE=(TRK,(1,1))
//SYSIN DD DISP=SHR,DSN=SYS1.VARYDISK.SYSIN
```

SYSIN FILE

The SYSIN cards must be 80-byte images with the VOLSER pattern placed in columns 1-6.
A single character wildcard of ‘%’ can be used to signify any character. All six character must be specified (e.g. if you wish to keep all volumes on-line starting ‘ST’, you must use ‘ST%%%’ and not ‘ST*’ or ‘ST%’. System symbols are supported as SYSIN cards, but the trailing dot must not be specified and the symbol name must be exactly six characters long. Comments are indicated by any of the following:

- Blank in column 1
- ‘/*’ in columns 1-2.

An example of a valid SYSIN file is shown below:

```
/=========================*/n
/* VARYDISK PARAMETER FILE */
/=========================*/

/* SHARED DISK */
%%%S%%
/* PRODUCTION DISKS */
%%%P%%
```

**VARYDISK RUN-TIME PARAMETERS**

The program accepts three optional run-time parameters:

- **MODE**
- **WTO**
- **PRINT**

The **TYPE** keyword indicates the run-time mode as follows:

- **MODE=TEST** – the program issues messages indicating the resultant status of the device. No action is taken against any of the DASD units. This mode allows the user to verify the SYSIN cards.
- **MODE=VARY** – the program varies the required devices off-line to the system.
- **WTO=NON** – no WTOs indicating device status are issued by this program.
- **WTO=ONL** – a WTO is issued for each device that is to remain ONLINE.
• WTO=OFF – a WTO is issued for each device that is to be varied OFFLINE.
• WTO=ALL – a WTO is issued for each device that is to be varied OFFLINE and also for devices that remain ONLINE.
• PRINT=Y – a WTO copy is written in SYSPRINT dataset.
• PRINT=N – no WTO copy is written to SYSPRINT dataset.

The default run-time options are:

**MODE=TEST, WTO=ALL, PRINT=Y**

```assembly
VARYDISK
VARYDISK CSECT
VARYDISK AMODE 31
VARYDISK RMODE 24
*
PATTERN_NUM EQU 000003             MAX NUMBER OF VOLSER PATTERNS
DISK_NUM     EQU 005000             MAX NUMBER OF DISK ENTRIES
*
*
* REGISTER USAGE:
*
*  R2  - POINTS TO PARAMETERS
*  R3  - INDEX ON PATTERN TABLE
*  R4  - WORK REGISTER
*  R5  - WORK REGISTER
*  R6  - WORK REGISTER
*  R7  - UCB
*  R8  - USED BY WRITE ROUTINE: POINTS TO WTO MSG
*  R9  - USED BY BRANCH AND LINK
*  R10 - PATTERN NUMBER
*  R11 - TABLE OF DISK
*  R12 - BASE REGISTER
*  R13 - WORKAREA
*

SAVE (14,12)
BASR R12,Ø
USING *,R12               R12 = BASE REGISTER
*
L    R2,Ø(R1)             PARAMETER ADDRESS
LH   R4,Ø(R2)             PARAMETER LENGTH
*
GETMAIN R,LV=WORKL
*
ST   R1,8(R13)
ST   R13,4(R1)
LR    R13,R1
USING WORK,R13
```

© 2001. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
C     R4,=F'Ø'             PARM LENGTH = Ø ?
BE    BADPARM

*      * COPY PARMS *
MVI   XPARM,C' "          CLEAR RECEIVING AREA
MVC   XPARM+1(L'XPARM),XPARM

*      COPY PARM LENGTH FOR BCT
LR    R5,R4                PARML (PARM LOOP)

*      SUBTRACT 1 FROM LENGTH FOR MOVE
BCTR  R4,Ø                 MOVE COMMAND FROM PARAMETER LIST
EX    R4,MOVE

*      COPY PARM LENGTH FOR BCT
MVC   MPARMT+3Ø(3Ø),XPARM
WTO   TEXT=MPARM

SET RUNTIME OPTIONS

*  OPTIONS,OTEST+OONLINE+OOFFLINE+OPRINT

*  DEFAULT OPTIONS:
*  *  MODE=TEST
*  *  WTO=ALL (ONLINE+OFFLINE)
*  PRINT=Y

PARML   EQU   *
CLC   Ø(Ø9,R2),=CLØ9"MODE=VARY'   SETVARY
BE
CLC   Ø(Ø9,R2),=CLØ9"MODE=TEST'
BE   SETTEST

*  SETWTON

CLC   Ø(Ø7,R2),=CLØ7"WTO=NON'
BE

*  SETWT00

CLC   Ø(Ø7,R2),=CLØ7"WTO=OFF'
BE   SETWTOF

*  SETWT0A

CLC   Ø(Ø7,R2),=CLØ7"WTO=ALL'
BE

*  SETPRY

CLC   Ø(Ø7,R2),=CLØ7"PRINT=Y'
BE

*  SETPRN

PARMN   EQU   *
LA   R2,1(R2)             NEXT PARAMETER CHAR
BCT R5,PARML  END OF PARAMETERS
B CONTØ1

* SETVARY EQU *
NI OPTIONS,255-OTEST  RESET FLAG
OI OPTIONS,OVARY
B PARMN

* SETTEST EQU *
NI OPTIONS,255-OVARY  RESET FLAG
OI OPTIONS,OTEST
B PARMN

* SETWTON EQU *
NI OPTIONS,255-OONLINE  RESET FLAG
NI OPTIONS,255-OOFFLINE RESET FLAG
B PARMN

* SETWTOO EQU *
NI OPTIONS,255-OOFFLINE RESET FLAG
OI OPTIONS,OONLINE
B PARMN

* SETWTOF EQU *
NI OPTIONS,255-OONLINE  RESET FLAG
OI OPTIONS,OOFFLINE
B PARMN

* SETWTOA EQU *
OI OPTIONS,OONLINE
OI OPTIONS,OOFFLINE
B PARMN

* SETPRY EQU *
NI OPTIONS,255-OPRINT  RESET FLAG
OI OPTIONS,OPRINT
B PARMN

* SETPRN EQU *
NI OPTIONS,255-OPRINT  RESET FLAG
B PARMN

* CONTØ1 EQU *
* INITIALIZE TABLE FOR VOLSER PATTERNS
* GETMAIN R,LV=6*PATTERN_NUM
ST R1,PATT_T
*
LR R3,R1
SR R1Ø,R1Ø  PATTERN COUNT
LA R4,6 PATTERN LENGTH
*
OPEN (SYSIN,(INPUT))
OPEN (SYSPRINT,(OUTPUT))
*
READ EQU *
GET SYSIN,INREC
CLC INREC(2),=CL2/*' COMMENT ?
BE READ YES - IGNORE THIS LINE
MVC Ø(6,R3),INREC COPY PATTERN
MVC MPATTERN+25(Ø6),Ø(R3)
LA R8,MPATTERN
BAL R9,WRITE
LA R10,1(R10) PATTERN COUNT + 1
L R5,=A(PATTERN_NUM)
CR R10,R5
BH MAXPATT MAX PATTERN NUMBER REACHED
AR R3,R4
B READ READ NEXT LINE
*
ENDSYSIN EQU *
*
ST R10,PATT_N SAVE PATTERN COUNT
*
GETMAIN R,LV=4*DISK_NUM
ST R1,DISK_T
LR R11,R1
SR R2,R2 OFFLINE DISK COUNT
*
XC UCBWORK,UCBWORK CLEAR WORKAREA
*
MODESET KEY=ZERO,MODE=SUP PATTERNS
*
GETUCB EQU *
*
UCBSCAN ADDRESS, X
WORKAREA=UCBWORK, X
DEVCLASS=DASD, X
UCBPTR=UCBADDR, X
DYNAMIC=YES, X
NOPIN, X
LOC=ANY, X
RANGE=ALL X
*
LTR R15,R15 LAST UCB ?
BNZ CONTØ2 YES
*
L R7,UCBADDR USING UCBOB,R7
*
TM UCBSTAT,UCBONLI IS THE DEVICE ONLINE ?
BZ GETUCB NO, GET NEXT UCB

* UCBDEVN DEVN=UNITADDR GET EBCDIC DEVICE NUMBER

* MVC VOLSER,UCBVOLI GET VOLSER
MVC DEVNBIN,UCBCHAN GET DEVN BINARY

* MVC MDEBUGT(4),UCBCHAN
* WTO TEXT=MDEBUG

  L R3,PATT_T POINTS TO FIRST PATTERN ENTRY
  L R6,PATT_N PATTERN NUMBER

PATL EQU *
  LR R7,R3
  LA R4,6
  LA R5,VOLSER

CHARL EQU *
  CLI Ø(R7),C'%' CHARN
  CLC Ø(1,R5),Ø(R7) BNE NEXTPAT

CHARN EQU *
  LA R7,1(R7)
  LA R5,1(R5)
  BCT R4,CHARL
  B VOLONL

* NEXTPAT EQU *
  LA R3,6(R3)
  BCT R6,PATL

* VOLOFF EQU *

  TM OPTIONS,OFFLINE ISSUE WTO ?
  BNO VOLOFF1

* MVC VOLOFFT+18(Ø6),VOLSER
MVC VOLOFFT+32(Ø4),UNITADDR
  LA R8,VOLOFF

* BAL R9,WRITE

* VOLOFF1 EQU *

  MVC Ø(2,R11),DEVNBIN ADD TO OFFLINE DISK TABLE
  LA R11,4(R11) POINT TO NEXT ENTRY
  LA R2,1(R2)
  L R5.=A(DISK_NUM)
  CR R2,R5
  BH MAXDISK MAX DISK NUMBER REACHED

  B CONTØ3
* VOLONL EQU *
  * TM OPTIONS,OONLINE ISSUE WTO ?
  BNO VOLONL1 NO *
  * MVC MVOLONLT+18(Ø6),VOLSER
  MVC MVOLONLT+32(Ø4),UNITADDR
  LA R8,MVOLONL *
  * BAL R9,WRITE *
  VOLONL1 EQU *
  * CONTØ3 EQU *
  * B GETUCB *
  CONTØ2 EQU *
  * TM OPTIONS,OTEST JUST TEST ?
  BO RETURN NO *
  * ST R2,MY_NUM
  L R11,DISK_T *
  * LA R2,2 *
  * ST R2,MY_NUM *
  * LA R11,MYUCB *
  * LA R4,MY_VDEV
  USING VDEV,R4 *
  * MVC MY_CALL,=CL8"VARYDISK'
  MVI VDEV_VERSION,VDEV_VERN
  MVC VDEV_ID,=CL4"VDEV'
  * INDICATE TO TURN IT OFF
  OI VDEV_KEYWORDS1,VDEV_OFFLINE *
  * IEEVARYD OPERATION=MY_VDEV,
    DEVICES=(R11),
    NUMDEV=MY_NUM,
    CALLERID=MY_CALL,
    RETCODE=MY_RET_C,
    RSNCODE=MY_RSN_C *
  * DC X'ØØ'
  * MVC MDEBUGT(4),MY_RET_C
  * MVC MDEBUGT+10(4),MY_RSN_C
  * WTO TEXT=MDEBUG
RETURN EQU *
 *
 * MODESET KEY=NZERO,MODE=PROB
 * 
 CLOSE (SYSIN)
 CLOSE (SYSPRINT)
 *
 * RELEASE STORAGE AREAS
 L R1,PATT_T
 FREEMAIN R,LV=6*PATTERN_NUM,A=(R1)
 *
 L R1,DISK_T
 FREEMAIN R,LV=4*DISK_NUM,A=(R1)
 *
 L R3,RET_CODE
 RESTORE RETURN CODE
 *
 L R13,4(R13)
 RESTORE R13
 L R1,8(R13)
 FREEMAIN R,LV=WORKL,A=(R1)
 LR R15,R3
 L R14,12(R13)
 LM R0,R12,2Ø(R13)
 BSM Ø,R14
 RETURN TO MVS AND USE RC=R15
 *
 BADPARM EQU *
 WTO TEXT=MBADPARM
 B RC_Ø8
 *
 MAXPATT EQU *
 WTO TEXT=MMAXPATT
 B RC_Ø8
 *
 MAXDISK EQU *
 WTO TEXT=MMAXDISK
 B RC_Ø8
 *
 RC_Ø8 EQU *
 LA R15,8
 ST R15,RET_CODE
 B RETURN
 *
 WRITE EQU *
 WTO TEXT=(R8)
 TM OPTIONS,OPRINT PRINT TO SYSPRINT ?
 BNO WRITE_R
 NO
 MVC OUTREC,2(R8) COPY MESSAGE
 PUT SYSPRINT,OUTREC
 WRITE_R EQU *
 BR R9
 RETURN
 *
 MBADPARM DS ØF
 DC H'8Ø'
VARIDISK COMPIATION AND LINK-EDIT
You should assemble and link-edit VARYDISK into an APF authorized load library. Use the following linkedit attributes:

- **AMODE(31)**
- **RMODE(24)**
- **AC(1).**
Re-entrant programming

INTRODUCTION
There are many applications where re-entrant programs are necessary or appropriate. Unfortunately, re-entrant programming has gained a reputation of being difficult. This article explains the principles involved and provides illustrative sample coding that show simple solutions for the associated ‘problems’.

WHAT IS RE-ENTRANT PROGRAMMING?
A single copy of a re-entrant program can be used in parallel by more than one user. Obviously, at any one time only one program is actually running. However, a re-entrant program may be exited by one user before it completes (eg to wait for an I/O event to complete) and be re-entered by another user at some other point before the first user returns. In a large system, there may well be very many users; one of the best examples is an airline reservation system where, more or less concurrently, many users will be requesting flight availability information, etc. This immediately makes evident the properties of a re-entrant program:

• Every user must have their own data area. The calling program normally provides a unique data area for each instance of the called re-entrant program. An alternative is for the called program to allocate its own data area.

• The program module cannot contain any variable data, ie data areas that are not read-only. If this was not the case, the same data could be changed by more than user. A possible relaxation of this requirement is global data, such as for a first time switch, etc. In this case, it is conceivable that such data is contained in the program. However, this conflicts with the requirement for true re-enterability, and it is better to keep such global data centrally outside the program. Furthermore, it may be more efficient to have the program code loaded into a re-entrant memory pool.
REUSABILITY

Reusability is a related, but much simpler, concept. A reusable program can have only one user before it finishes processing (exits). It can then have a different user. Every call to the program must perform any required reinitialization on entry to the program. A re-entrant program is also reusable.

REQUIREMENTS FOR RE-ENTRANT PROGRAMMING

Each instance must initialize its data area. Because many IBM macros require a parameter list, such macros are provided in three forms:

- Standard form
- Executable form
- List form.

The standard form is used for non-reentrant programming. The executable and list forms are used for re-entrant programming. The MF (Macro Form) keyword specifies the macro form (E = executable form, L = list form). Whereas the list form builds the static parameter list, the executable form adds specific (dynamic) parameters to the associated list. Although in some simple cases, eg the CALL macro, the list form represents only placeholders, in most cases it also contains flags, etc. This means that such list form macro expansions must also be included in the initialization. Unless the programmer knows the form of the macro concerned, he should include it in the initialization.

Example for the use of the execute and list forms of a macro:

```
macname  dynparms,MF=(E,listname)
listname  macname  statparms,MF=L
```

where:

- `macname` – representative macro name.
- `dynparms` – representative dynamic parameters.
- `listname` – representative list name; the name of the associated list-form macro.
- `statparms` – representative static parameters.
ADVANTAGES OF RE-ENTRANT PROGRAMMING
Reentrant programming has the advantage that only one copy of the program needs to be present irrespective of the number of users. Frequently used components can be placed in the LPA (Link Pack Area).

DISADVANTAGES OF RE-ENTRANT PROGRAMMING
The disadvantage of re-entrant programs lie in their increased complexity and additional processing:

- Variable data must be stored in a work area passed to the program (or allocated by the program itself). Additional areas can be allocated dynamically (GETMAIN) with the addresses and lengths being stored in this work area.

- The initialization must be performed dynamically.

The techniques described in this article reduce these disadvantages to a minimum.

Note: Although for traditional reasons GETMAIN is mentioned, STORAGE OBTAIN will normally be used because of its better performance.

SAMPLE RE-ENTRANT PROGRAM

RENTPGM CSECT
RENTPGM AMODE 31
RENTPGM RMODE ANY
  BAKR 14,Ø save registers and return address
  BASR 12,Ø set base register
  USING *,12 assign base register

SPACE
  * allocate work area
  STORAGE OBTAIN,LENGTH=LUA
  * register 1: address of allocated dynamic area
    LR 9,1 address of dynamic area
  * initialise work area
  DYN USING UA,9 explicit use of the dynamic area
    LA 0,DYN.UADATA address of dynamic area
    LA 1,L'UADATA length of dynamic area
    LA 14,UADATA address of the definitions
    LR 15,1 length
MVCL 0,14 initialise dynamic area
USING UA,9 use variables in the dynamic area
MVC MSGTEXT,=CL8'TESTMSG'
LA 2,MSGLEN
WTO TEXT=(2),MF=(E,WTLIST)
SPACE
EOP LA 15,0 set return code
PR , program return
SPACE
LTORG , literal origin (must precede user-area)
SPACE
* start of definition of the user-area
UA DS ØD user-area, align on double-word
MSGLEN DC AL2(MSGTEXTE-MSG)
MSG DC C'HDR:'
MSGTEXT DS CL8
MSGTEXTE EQU *
SPACE
WTLIST WTO TEXT=,ROUTCDE=11,MF=L
SPACE
LUA EQU *-UA length of user-area
ORG UA reset origin
UADATA DS CL(LUA) redefine field with implicit length (LUA)
* end of definition of the user-area
END

Although this sample program is very simple, it demonstrates the features of reentrant programming.

This sample reentrant program allocates its own work area. This has the advantage that it allocates the space it needs, but at the cost of performing a (relatively time-intensive) GETMAIN. If a reentrant (sub)program is called very frequently, it may be better for the calling program to perform its own storage management and pass an appropriate work area to the called program. This has the disadvantage that the calling program needs to know the size of such an area.

The key to this program is the use of the labelled USING to simplify the initialization of the work area. To simplify the explanation, MVC rather than MVCL is used in the following code (the restriction to 256 moving maximum characters does not matter in this case).

In the program DYN.UADATA specifies that the USING with the name DYN is to be used as base address (namely register 9). Register 9 addresses to the dynamic area.

Also UADATA without a prefix specifies that the current base register
(namely register 12) is to be used for addressing, ie the static data area defined in the program code.

The subsequent USING UA,9 without label allows field names to be used from the dynamic area without requiring the prefix.

<table>
<thead>
<tr>
<th>DYN</th>
<th>USING UA,9</th>
<th>explicit use of the dynamic area</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVC</td>
<td>DYN.UADATA,UADATA</td>
<td>move to dynamic area</td>
</tr>
</tbody>
</table>

**Need help with an MVS problem or project?**

Maybe we can help:

- If it is on a topic of interest to other subscribers, we will commission an article on the subject, which we will publish in *MVS Update*, and which we will pay for – it will not cost you anything.

- If it is a more specialized, or more complex, problem, you can advertise your requirements (including one-off projects, freelance contracts, permanent jobs, etc) to the hundreds of MVS professionals who visit *MVS Update*’s home page every month. This service is also free of charge.

Visit the *MVS Update* Website, http://www.xephon.com/mvsupdate.html, and follow the link to Suggest a topic or Opportunities for MVS specialists.
INTRODUCTION
The GDG transfer program was first published in *MVS Update* Issue 107, August 1995, and is available to download off the Xephon Web site.

The GDG Transfer Program reads the GDG name with a relative generation number, extracting an absolute generation number and a version number (generation data set in full), extracts VOLSER numbers, reading the JCL from the input DD statement, replacing the `##VOLSERS` statement with the `// VOL=SER=(volsers list)` statements, replacing the `##DSN` statement with a `// DSN=datasetname` statement and submitting (if necessary) the modified JCL (via internal reader).

The modified JCL can be transferred and executed in another MVS environment (via ‘/*XEQ nodename’ JCL JES2 statement).

The restriction included in the source code is related to the number of tapes extracted for a GDG. In the 1995 version, the upper limit is 20. The current modification removes this restriction.

It is done by implementing the Catalog Search Interface, program name IGGCSI00. This technology is documented in *DFSMS/MVS Version 1 Release 5, Managing Catalogs* SC26-4914-04, Appendix D.

As the major part of the source code published in 1995 remains unchanged, only the necessary changes are supplied with this update. The source code should be changed as follows:

1. Find the instruction: `MVC DSNAME(44),GDGBASE`. Remove the instructions: `LA R1,0` and `ST R1,VOLCOUNT`
2. Find the line: `OKLOCATE EQU *`
   
   Delete the instructions between `OKLOCATE EQU *` and `OPEN (SUCKFDCB,(INPUT))` (do not delete the two above instructions).
3 Delete the instructions between MVC INJCLREC(12),=C'// VOL=SER=(' and B NOSPIT (before NOVOLSER EQU *)
(do not delete the two above instructions).

4 Between MVC INJCLREC(12),=C'// VOL=SER=(' and B NOSPIT (before NOVOLSER EQU *) include the following
string of instructions:

*==== begin of include 1  ====================================
MVC CSIFILTK(44),DSNAME
LA R1,PARMLIST
CALL IGGCSIØØ
LTR R15,R15
BZ OKRC
MVC PRINT(133),BLANK
MVC PRINT+1(16),=C'IGGCSIØØ RC NE Ø'
PUT PRINTDCB,PRINT
B FREEM

OKRC EQU *
USING DATADSEC,R5
LA R5,WORKAREA
L R1,CSICRETN
LTR R1,R1 TEST RC
BZ DATARCOK
MVC PRINT(133),BLANK
MVC PRINT+1(16),=C'DATA RC NE Ø'
PUT PRINTDCB,PRINT
B FREEM

DATARCOK EQU *
LA R4,DSECTEND
USING MAPENTRY,R4
LA R9,VOLSENTR
MVC PRINT(133),BLANK
MVC PRINT+1(44),CSIENAME MOVE DATA SET NAME
PUT PRINTDCB,PRINT PUT
XR R3,R3
LH R3,EFLD2LN DATA LENGTH
LA R5,INJCLREC+12 (R5) = ADDRESS OF TAPE VOLUME

LOOPVOLS EQU *
MVC Ø(6,R5),Ø(R9)
MVI INJCLREC+18,C','
LA R1,6
CR R3,R1
BNE OKCOMMA
MVC INJCLREC+18(2),=C'),,'

OKCOMMA EQU *
MVC JCLREC(ØØ),BLANK
MVC JCLREC(ØØ),INJCLREC
PUT SPITFDCB,JCLREC

MVC PRINT,BLANK
MVC PRINT+1(8),JCLREC
PUT PRINTDCB,PRINT
MVC INJCLREC+3(9),BLANK
A R9,=F'6'
S R3,=F'5'
BCT R3,LOOPVOLS

*----- END of include 1 ---------------------------------------------

5. Between NEXTEXTE DS CL251
   and
   PRINTDCB DCB MACRF=PT,RECFM=FBA,LRECL=133,BLKSIZE=133,DSORG=PS,
   include the following string of instructions:

   *=====BEGIN 2 ==================================================

   * PARAMETER LIST
   PARMLIST DS ØD
   PARMREAS DC A(REASAREA) 1ST WORD
   DC A(SELECRIT) 2ND WORD
   DC A(WORKAREA) 3RD WORD

   * ID,REASON CODE,RC
   REASAREA DS ØF
   MODULEID DC XL2'ØØØØ' MODULE IDENTIFICATION
   REASCODE DC XL1'ØØ' REASON CODE
   RETUCODE DC XL1'ØØ' RETURN CODE

   * SELECTION CRITERIA FIELDS
   SELECRIT DS ØF
   CSIFILTK DC CL44' ' DSNAME
   CSICATNM DC CL44' ' CSICATNM DC CL44' '
   CSISTYPS DS ØCL16
   CSIOPTS  DS ØCL4
   CSIRESRV DC XL1'ØØ'
   CSINUMEN DC H'1'
   CSIENTS  DS ØCL8
   CSIFLDNM DC CL8'VOLSER ' FIELD NAME

   *=====END 2 ==================================================

6. Delete lines between LTORG and END
7. Between LTORG and END include the following string of instructions:

   *=====BEGIN 3 ==================================================

   WORKAREA DS ØF
   DC F'32000' CSIU SRLN
   DS XL32000

   DUMMYREC DSECT
   GDGBASE DS CL44 GDGBASE FIELD FROM THE EXEC STMT
**Free weekly news by e-mail**

Four weekly news services are available free of charge from Xephon, covering the following subject areas: data centre, distributed systems, networks, and software.

Each week, subscribers receive, by e-mail, a short news bulletin consisting of a list of items; each item has a link to the page on our Web site that contains the corresponding article. Each news bulletin also carries links to the main industry news stories of the week. You can subscribe to one or more of these news services, or review recent articles at the following URL: http://www.xephon.com.
IBM has previewed its NUMA-Q Enabled For S/390 (EFS), which essentially runs S/390 operating systems, including OS/390, VM/ESA, and VSE/ESA, on the Intel-based NUMA-Q 2000 servers. No changes are required to run the operating systems and associated programs and applications. It does this through Fundamental Software’s FLEX-ES product. It is being positioned as an entry-level S/390 server option for technology upgrades or incremental performance increases that require less capacity. Multiprise 3000 Model H30. But it maintains its ability to run applications in DYNIX/ptx, Linux, and Windows 2000. The S/390 layer follows the S/390 design rules for the maximum amount of storage available for use as central storage and expanded storage. The S/390 processors are mapped to the physical NUMA-Q 2000 Intel processors on a 1:1 basis with the exception of the first processor, which is used for DYNIX/ptx resource management and is not enabled for S/390.

Optional S/390 features are available via the licensed product features and include Parallel Channel Adapter (PCA) in either a one- or three-channel version and the Integrated Communications Adapter (ICA). The S/390 Enablement does not support all functions. Among these are z/OS and OS/390 Version 2 Release 10, over 18 TB/image of disk storage for S/390, Parallel Sysplex, ESCON, DASD on PCA, SNA over Ethernet, VM Dataspace support, Data Compression, and IEEE floating point.

Contact your local IBM representative for further information.

http://www.numaq.com/hardware

---

IBM has announced Version 2.6 of its Tivoli NetView Performance Monitor, which monitors, records, and reports network communication, performance, and utilization through both Java-based GUI and traditional 3270 SNA displays.

Enhancements include claimed better monitoring of SNMP router data by adding threshold relations, recovery of the SNMP router collection on TCP/IP failure, additional Cisco router support, improved 3270 panel display support, tighter integration with Tivoli NetView, and recovery of GUI Interface Connection on TCP/IP failure. Now integrated with NetView for OS/390, Version 2.6 runs on OS/390 Version 2 Release 5 or later, and z/OS Version 1 Release 1.

The company also announced Version 1.2 of NPM for TCP/IP, for managing TCP/IP OS/390 systems and the connected TCP/IP network. It can identify and reduce network resource congestion and can be installed, configured, and run in most environments in less than a day.

For further information contact:

Tivoli Systems, 9442 Capital of Texas Highway, North Austin, TX 78759, USA.
Tel: 512 436 8000
Fax: 512 794 0623

Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK.
Tel: 01753 896 896
Fax: 01753 896 899

http://www.tivoli.com

---

Contact your local IBM representative for further information.

http://www.numaq.com/hardware

* * *