159 MVS

December 1999

update

In this issue

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Contributions

If you have anything original to say about MVS, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all MVS users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *MVS Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

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; 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.

Printed in England.

Sorting hexadecimal data

THE PROBLEM

The following edit macro was developed to solve a specific situation. I had a file containing a list of disk models, volume labels, and their addresses, which I wanted to sort by model and by hexadecimal address. The problem was that with a normal sort, characters A-F appear before digits 0-9, and I wanted them to appear after, as in hexadecimal logic.

A SOLUTION

This macro solves the problem by sorting a file as if the EBCIDIC sequence was:

```
X'00'-x'80' , x'f0'-x'ff' , x'81'-x'ef'
```

The numbers are placed just before lowercase ‘a’ (or X'81'). The macro does not check for any valid hexadecimal numbers, it simply sorts data according to the above sequence.

The syntax is somewhat similar to the standard ISPF editor sort, with the following differences:

- Any number of column pairs can be specified.
- After a column pair, specify ‘A’ or ‘D’ for ascending or descending. Ascending is the default, and can be omitted.
- Line labels are not supported. To indicate a range of lines specify the line numbers preceded by a dot (for example .15).
- Bounds setting is honoured as in a standard sort, but I warn the user if the bounds are restrictive, and give him a chance to exit the macro before ruining the file, as happened to me (and to others) some time ago with a regular sort. Bounds are an invisible and transmissible-across-edit-sessions disease (sorry, ‘feature’) that I avoid like the plague.

Some examples of SORTH use:

- Ascending assumed in first pair:

```
sorth 1 5  16 22 d  4Ø 45 d
```

- Same as above, only between lines 35 and 55:

```
sorth 1 5 a 16 22 d 4Ø 45 d .35 .55
```

- Gives a quick on-line description:

```
sorth ?
```

REXX

```
/*== REXX ISPF Editor macro *=======================================*/
/*                                                                  */
/*  SORTH - Sorts data as hexadecimal values                        */
/*                                                                  */
/*==================================================================*/
address ISPEXEC
'ISREDIT MACRO (ARG)'
'ISREDIT RESET'
'ISREDIT (SIZE1) = linenum .zlast'
'ISREDIT (B1,B2) = BOUNDS'
'ISREDIT (LR) = LRECL'
 b1 = strip(b1,"L","Ø")

/*================== Warn if bounds are active ======================*/
if b1 > 1 | b2 <> lr then do
   say "SORTH - WARNING: Bounds are set to" b1 b2
   say "        Hit ENTER to continue,"
   say "        or type anything to cancel SORTH."
   pull resp
   if resp<>"" then exit
   b1a = b1-1
   b2a = lr-b2
   bØ  = b2-b1+1
   bounds = 1
end

/*================== Get and validate parameters ====================*/
upper arg
tag1=""
tag2=""
z = Ø
do alpha = Ø
  select
     when arg="?" | arg="HELP" then signal helpe
     when arg = "" then leave alpha
     when word(arg,1) ="A" |,
```

```
          word(arg,1) ="D" then do
         if z = Ø then signal erro
         sortype.z = word(arg,1)
         arg = subword(arg,2)
      end
      when left(arg,1)="." then do
         number = substr(word(arg,1),2)
         if datatype(number,"W")<>1 then signal erro
         if number > size1 then number = size1
         if tag1="" then tag1 = number
         else tag2 = number
         arg = delword(arg,1,1)
      end
      otherwise do
         z = z+1
         sort1.z = word(arg,1)
         sort2.z = word(arg,2)
         sortype.z = "A"
         if datatype(sort1.z,"W")<>1 |,
            datatype(sort2.z,"W")<>1 |,
            sort1.z > sort2.z  then signal erro
         arg = subword(arg,3)
      end
   end
end

if z = Ø then signal erro
if tag1="" then tag1 = 1
if tag2="" then tag2 = size1
if tag2 < tag1 then signal erro
if bounds = 1 then do z1 = 1 to z
   if sort1.z1 < b1 | sort2.z1 > b2 then do
      say "Sort columns outside bounds. SORTH cancelled"
      exit
   end
end

/*================== Who wins: Ascending or Descending? =============*/
winner = Ø
do z1 = 1 to z
   if sortype.z1 = "A" then winner = winner + 1
   else winner = winner - 1
end
if winner < Ø then winner = "D"
else winner = "A"

/*================== Get lines from file ============================*/
do k = tag1 to tag2
  "ISREDIT (STUPID) = line" k
   if bounds = 1 then do
```

```
      if b1a > Ø then lin_ini.k = left(stupid,b1a)
      if b2a > Ø then lin_end.k = right(stupid,b2a)
      lin.k = substr(stupid,b1,bØ)
   end
   else do
      lin.k = stupid
   end
end

/*================== Sort records ==================================*/

direc = xrange()
rever = reverse(direc)
do k = tag1 to tag2
   linezone.k = ""
   do a1 = 1 to z
      outzone = ""
      zone=substr(lin.k,sort1.a1,sort2.a1-sort1.a1+1)
      do x = 1 to length(zone)
         char = substr(zone,x,1)
         if char > x2c(8Ø) & char < "Ø" then
           outzone=outzone||"Z"substr(zone,x,1)
         else
           outzone=outzone||"$"substr(zone,x,1)
      end
      if sortype.a1 <> winner then do
         outzone = translate(outzone,rever,direc)
      end
      linezone.k = linezone.k || outzone
   end
end

if winner = "A" then do
  do j = tag1 to tag2
    do k = j to tag2
      if linezone.k < linezone.j then do
         temp = linezone.k
         linezone.k = linezone.j
         linezone.j = temp
         temp = lin.k
         lin.k = lin.j
         lin.j = temp
      end
    end
  end
end

else do
  do j = tag1 to tag2
    do k = j to tag2
```

```
      if linezone.k > linezone.j then do
         temp = linezone.k
         linezone.k = linezone.j
         linezone.j = temp
         temp = lin.k
         lin.k = lin.j
         lin.j = temp
      end
    end
  end
end

/*================== Put lines back in file and exit ================*/
do k = tag1 to tag2
   if bounds = 1 then do
      if b1a > 0 then lin.k = lin_ini.k || lin.k
      if b2a > 0 then lin.k = lin.k || lin_end.k
   end
   "ISREDIT LINE " k " = '"lin.k"'"
end
exit

/*================== Help text and error exit =======================*/
erro:
say "Error in parameters"
exit

helpe:
say "SORTH  -  Sorts data as hexadecimal numbers (base 16), by giving"
say "        a lower value to figures 0-9 than to letters a-z or A-Z."
say
say "Format:"
say
say "SORTH c1 c2  A|D   .line1 .line2 "
say "      c1 c2  are column pairs. Any number of column pairs can be"
say "      specified.  A  or  D  indicate  ascending (the default) or"
say "      descending sort, for the column pair;  .line1  and  .line2"
say "      are line numbers to limit the scope of the sort."
say "      Like ISPF sort,  SORTH  respects the  bounds setting,  but"
say "      warns the user about them, if they are restrictive."
say
say "Examples:"
say
say "  SORTH 1 8  D     (sort entire file, descending, columns 1  8 )"
say "  SORTH 20 24  1 6  .100 .189   (sort between lines 100 and 189)"
```

Luis Paulo Ribeiro
Systems Engineer
Edinfor (Portugal)

Copying files between MVS and OpenMVS

INTRODUCTION

We regularly need to exchange files between MVS and OpenMVS. There are a few ways to do this (such as OCOPY, OGET and OPUT), but neither is very user-friendly. We therefore developed a panel-driven utility called OPENCOPY.

It is possible to copy sequential files or members to the HFS or *vice versa*. When copying to the HFS you must set some options such as:

- pathdisp – whether the file should be kept or deleted at normal and abnormal terminations.
- pathmode – determines the file security attributes.
- pathopts – whether the file should be created, replaced, or overwritten.

REXX OPENCOPY

```
/* REXX                                                          */
/* function : Perform between  OMVS - MVS environment            */
/* invocation : OMVS panel                                       */

/* Provide trace possibility */
PARSE UPPER ARG "TRACE("PTRACE")"
IF PTRACE ¬= "" THEN
DO
  TRACE = PTRACE
END
IF TRACE > "Ø" THEN
DO
  TRACE ?r
END

/* This is it ... */
Call Init
Call Main
exit

Init:
x  = MSG('off')
Return
```

```
Main:
/* Show panel */
do forever
 "ISPEXEC CONTROL ERRORS RETURN"
 "ISPEXEC DISPLAY panel(OPENCOPY)"
 if rc >< Ø then
  do
   leave
  end
 "FREE DDNAME(FROMDD)"
 "FREE DDNAME(TODD)"
  x  = MSG('on')
 select
  when ftype = 'MVS' then
    do
      /* Is there a source dataset ? */
      x =  Check_MVS(fromdsn)
      if x > Ø then iterate
      /* yes,  ... continue   */
      "ALLOC DDNAME(FROMDD) DSNAME('"dsn"')  SHR"
      if rc >< Ø then
       do
         iterate
       end
    end
  when ftype = 'UNIX' then
    do
      /* Format  source path */
      fromdsn = strip(fromdsn,L,"'")
      fromdsn = strip(fromdsn,T,"'")
      /* determine Unix Inputfile Disposition */
      path_indisp = ''
      select
       when id = 1 then path_indisp = "KEEP,KEEP"
       /* put in comment in panel; because if alloc      */
       /* output fails, input will be deleted nonetheless*/
       when id = 2 then path_indisp = "DELETE,KEEP"
       otherwise        path_indisp = "KEEP,KEEP"
      end
      if path_indisp >< '' then
       do
         path_indisp = "PATHDISP("path_indisp")"
       end

      /* do the alloc, and see what happens */
      msg. = ''
      x = OUTTRAP('msg.')

      "ALLOC DDNAME(FROMDD) PATH('"fromdsn"') "path_indisp
      if rc >< Ø then
```

```
      do
        zedsmsg = 'OCOPY Allocation problem'
        zedlmsg = ''
        do h = 1 to msg.Ø
           zedlmsg = zedlmsg||msg.h
        end
        "ISPEXEC SETMSG MSG(ISRZØØ1)"
        iterate
      end
      x = OUTTRAP('OFF')
   end
end

select
 when ttype = 'MVS' then
   do
     /* Is there a target dataset? */
     x =  Check_MVS(todsn)
     if x  > Ø then iterate
     /* yes,  ... continue   */
     "ALLOC DDNAME(TODD) DSNAME('"dsn"')   SHR"
     if rc >< Ø then
      do
        iterate
      end
   end
 when ttype = 'UNIX' then
   do
     /* strip off all apostrophes; we want a clean Unix name */
     todsn   = strip(todsn,L,"'")
     todsn   = strip(todsn,T,"'")
     /* determine Unix Output Disposition */
     select
      when od = 1 then path_outdisp = "KEEP,DELETE"
      when od = 2 then path_outdisp = "KEEP,KEEP"
      otherwise        path_outdisp = "KEEP,DELETE"
     end
     /* determine pathopts : accessgroup  */
     select
      when ag = 1 then pathopt1 = "ORDWR"
      when ag = 2 then pathopt1 = "ORDONLY"
      when ag = 3 then pathopt1 = "OWRONLY"
      otherwise        pathopt1 = "ORDWR"
     end
     /* determine pathopts : statusgroup  */
     select
      /* create new, if file already exist do not open           */
      when sg = 1 then pathopt2 = "OCREAT,OEXCL"  /* +- disp=shr */
      /* create new, if file already exist open it (overwrite)   */
      when sg = 2 then pathopt2 = "OCREAT"        /* +_ disp=new */
      when sg = 3 then pathopt2 = "OAPPEND"       /* +- disp=mod */
```

```
        otherwise        pathopt2 = "OCREAT,OEXCL"
       end
       /* compose pathopts */
       pathopts = pathopt1","pathopt2
       /* determine pathmode options        */
       call Check_PathMode

       /* do the alloc, and see what happens */
       msg. = ''
       x = OUTTRAP('msg.')
       "ALLOC DDNAME(TODD) PATH('"todsn"') PATHDISP("path_outdisp")",
       " PATHOPTS("pathopts") "pathmode
       if rc >< Ø then
        do
          zedsmsg = 'OCOPY Allocation problem'
          zedlmsg = ''
          do h = 1 to msg.Ø
             zedlmsg = zedlmsg||msg.h
          end
          "ISPEXEC SETMSG MSG(ISRZØØ1)"
          iterate
        end
       x = OUTTRAP('OFF')
     end
  otherwise nop
 end

/* do the OCOPY    .... */
  msg. = ''
  x = OUTTRAP('msg.')

  "OCOPY INDD(FROMDD) OUTDD(TODD) " TYP "CONVERT("CONV") ",
  "PATHOPTS(USE)"
  if rc >< Ø then
   do
    zedlmsg = ''
    do h = 1 to msg.Ø
       zedlmsg = zedlmsg||msg.h
    end
    zedsmsg = 'OCOPY ended with rc ' rc
    "ISPEXEC SETMSG MSG(ISRZØØ1)"
   end
  else
   do
    zedsmsg = 'OCOPY ended'
    "ISPEXEC SETMSG MSG(ISRZØØ1)"
   end
  x = OUTTRAP('OFF')
end
call Free_All
Return
```

```
Check_MVS: procedure expose dsn
arg dsn
code = Ø
x    = LISTDSI(dsn)
if SYSREASON > Ø then
  do
    zedsmsg = 'Dataset Not Catalogued'
    zedlmsg = dsn       'was not found in the catalog'
    "ISPEXEC SETMSG MSG(ISRZØØ1)"
    code    = 8
  end
 else
  do
    parse var dsn dataset '(' mem ')'
    if mem >< '' then dsn = SYSDSNAME||'('||mem||')'
    else dsn = SYSDSNAME
  end

Return code

Check_PathMode:
j = Ø
Path_Acc. = ''
Pathmode  = ''

/* Access ????  */
if UR = 'Y' | UR = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIRUSR'
 end
if UW = 'Y' | UW = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIWUSR'
 end
if UX = 'Y' | UX = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIXUSR'
 end
if UL = 'Y' | UL = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIRWXU'
 end
if GR = 'Y' | GR = 'y'  then
 do
   j = J + 1;Path_Acc.j = 'SIRGRP'
 end
if GW = 'Y' | GW = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIWGRP'
 end
```

```
if GX = 'Y' | GX = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIXGRP'
 end
if GL = 'Y' | GL = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIRWXG'
 end
if OR = 'Y' | OR = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIROTH'
 end
if OW = 'Y' | OW = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIWOTH'
 end
if OX = 'Y' | OX = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIXOTH'
 end
if OL = 'Y' | OL = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SIRWXO'
 end
if US = 'Y' | US = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SISUID'
 end
if GS = 'Y' | GS = 'y' then
 do
   j = J + 1;Path_Acc.j = 'SISGID'
 end

do k = 1 to j
   if   k > 1 then Path_Acc.k = ","||Path_Acc.k
   Pathmode = PathMode||Path_Acc.k
end
if pathmode >< '' then
 do
   Pathmode = "PATHMODE("pathmode")"
 end

Return Ø

Free_All:
 "FREE DDNAME(FROMDD)"
 "FREE DDNAME(TODD)"
Return
```

OPENCOPY PANEL

```
)ATTR
$ TYPE(TEXT)   INTENS(LOW) COLOR(GREEN)
% TYPE(TEXT)   INTENS(HIGH)
+ TYPE(TEXT)   INTENS(LOW)
# TYPE(TEXT)   INTENS(LOW) SKIP(ON)
_ TYPE(INPUT)  INTENS(HIGH) CAPS(OFF)
)BODY
+                            OCOPY Utility
$Command ===>_ZCMD
+
+ Specify "From" Data Set/Path below
+  $Data Set Name   . . ._Fromdsn                                    #
+           $Type   . . ._Z#           %1. MVS    2.UNIX+(Case sensitive!)
+
+ Specify "To"  Data Set/Path below
+  $Data Set Name   . . ._Todsn                                      #
+           $Type   . . ._Z#           %1. MVS    2.UNIX+(Case sensitive!)
+
+ Pathdisp                                    Conversion Options
+$Input                 Output                Convert         Type
+_Z#%1.KEEP,KEEP       _Z#%1.KEEP,DELETE     _Z#%1.yes+      _Z#%1.Text+
+                         %2.KEEP,KEEP          %2.no+          %2.Binary+
+
+ Access Options
+ Pathmode (Y/N)                             +Pathopts
+_Z#%SIRUSR_Z#%SIRGRP_Z#%SIROTH_Z#%SISUID    $Acces group     Status group
+_Z#%SIWUSR_Z#%SIWGRP_Z#%SIWOTH_Z#%SISGID    _Z#%1.ORDWR
_Z#%1.OCREAT,OEXCL
+_Z#%SIXUSR_Z#%SIXGRP_Z#%SIXOTH                %2.ORDONLY      %2.OCREAT
+_Z#%SIRWXU_Z#%SIRWXG_Z#%SIRWXO                %3.OWRONLY      %3.OAPPEND
)INIT
&CV     = '1'
&TYP    = '1'
&PD     = '1'
.ZVARS='(FT,TT,ID,OD,CV,TYP,UR,GR,OR,US,UW,GW,OW,GS,AG,SG,UX,GX,OX,UL,GL,OL)'
.HELP   = OPENHLPO
.CURSOR = FROMDSN
)REINIT
 &CONV   = ''
 &TOTYP  = ''
 &FROMTYP= ''
)PROC
  VER(&FROMDSN,NB)
  VER(&FT,NB,LIST,1,2)
  VER(&TODSN,NB)
  VER(&TT,NB,LIST,1,2)
  VER(&CV,LIST,1,2)
  VER(&TYP,LIST,1,2)
  VER(&ID,LIST,1)
  VER(&OD,LIST,1,2)
```

```
  VER(&UR,LIST,Y,N,y,n)
  VER(&UW,LIST,Y,N,y,n)
  VER(&UX,LIST,Y,N,y,n)
  VER(&UL,LIST,Y,N,y,n)
  VER(&GR,LIST,Y,N,y,n)
  VER(&GW,LIST,Y,N,y,n)
  VER(&GX,LIST,Y,N,y,n)
  VER(&GL,LIST,Y,N,y,n)
  VER(&OR,LIST,Y,N,y,n)
  VER(&OW,LIST,Y,N,y,n)
  VER(&OX,LIST,Y,N,y,n)
  VER(&OL,LIST,Y,N,y,n)
  VER(&US,LIST,Y,N,y,n)
  VER(&GS,LIST,Y,N,y,n)
  VER(&AG,LIST,1,2,3)
  VER(&SG,LIST,1,2)
  IF (&FT = '1')
     .ATTR (FROMDSN) = 'CAPS(OUT)'
  IF (&TT = '1')
     .ATTR (TODSN) = 'CAPS(OUT)'
  &FTYPE= TRANS (&FT 1,MVS 2,UNIX)
  &TTYPE= TRANS (&TT 1,MVS 2,UNIX)
  &CONV = TRANS (&CV  1,YES 2,NO)
  &TYP  = TRANS (&TYP  1,TEXT 2,BINARY)
  &ZSEL = TRANS (TRUNC (&ZCMD,'.')
               *,'?')
)END
```

OPENHLP0 PANEL

```
)ATTR
% TYPE(TEXT)   INTENS(HIGH)
+ TYPE(TEXT)   INTENS(LOW)
_ TYPE(INPUT)  INTENS(HIGH)
)BODY
+
+                        OCOPY Utility - Help
+
+ The OCOPY utility allows you to copy files/members between the
+ Hierarchical File System (HFS) and datasets or libraries.
+
+ The behaviour of the Unix path structure is determined by
+ the PATH, PATHDISP, PATHMODE, and PATHOPTS operands.
+ Further on these are briefly explained but for a discussion
+ in depth refer to:
+
+  %- TSO Command reference
+  %- Unix System Services User's Guide
+  %- Unix System Services Command Reference
+
+
```

```
+
+
+
+
+
+
+                                                          (%Enter+to continue)
)INIT
)PROC
 &ZCONT = OPENHLP1
)END
```

OPENHLP1 PANEL

```
)ATTR
% TYPE(TEXT)   INTENS(HIGH)
+ TYPE(TEXT)   INTENS(LOW)
_ TYPE(INPUT)  INTENS(HIGH)
)BODY
+
+                        OCOPY Utility - Help
+
% PATH+     Identifies an HFS file. A path name is case sensitive.
+
% PATHDISP+ Specifies the disposition of an HFS file upon normal or
+           abnormal TSO session termination.
            KEEP   specifies that the file should be kept.
            DELETE specifies that the file should be deleted.
+
% PATHMODE+ Specifies the file access attributes when the%PATHOPTS+
+           operand specifies OCREAT.
+
%    SIRUSR,SIRGRP,SIROTH+: specifies permission for respectively
+           the file owner, users in the file group class, users in the
+           file other class to read the file.
%    SIWUSR,SIWGRP,SIWOTH+: specifies permission for respectively
+           the file owner, users in the file group class, users in the
+           file other class to write the file.
%    SIXUSR,SIXGRP,SIXOTH+: specifies permission for respectively
+           the file owner, users in the file group class, users in the
+           file other class to search or execute.
+                                                          (%Enter+to continue)
)INIT
)PROC
 &ZCONT = OPENHLP2
)END
```

OPENHLP2 PANEL

```
)ATTR
% TYPE(TEXT)   INTENS(HIGH)
+ TYPE(TEXT)   INTENS(LOW)
_ TYPE(INPUT)  INTENS(HIGH)
)BODY
+
+                       OCOPY Utility - Help
+
%    SIRWXU,SIRWXG,SIRWXO+: specifies permission for respectively
+           the file owner, users in the file group class, users in the file
+           other class to read,write,search or execute.
%    SISUID,SISGID+: system sets respectively the UID or GID to be the same
+           as respectively the user-id, group-id of the file owner when the
+           file runs as a program.
% PATHOPTS+ Specifies the file access and status used when accessing a file
+           ; you can specify up to seven options.
+    ORDWR,ORDONLY,OWRONLY+: Open file for reading, writing or reading/
           writing.
+    OAPPEND +: sets file offset to the end of the file before writing.
+    OCREAT  +: systems creates the file. If the file already exists the
           operation depends of OEXCL.
           If OEXCL is specified the creation fails, else it succeeds and
           the file  is overwritten.
+
+
+
+
+                                                    (%Enter+to continue)
)INIT
)PROC
 &zcont = OPENHLP0
)END
```

Guy Riems (Belgium)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from our Web site. The URL is: www.xephon.com/contnote.html.

A real-time coupling facility monitor

COUPLING FACILITY MONITORING TOOLS

When you decide to implement a Parallel Sysplex, you will need a tool to tune your coupling facility activity. You will have to check your structures allocations in your different coupling facilities, structures response time, structures access rates, etc. One solution is to use RMF to produce reports about coupling facilities and structures. The major drawback of this solution is that it is not a real-time process.

To get real-time information about your coupling facility configuration, IBM provides only a set of MVS commands, which do not provide everything you will need.

IBM COMMANDS ARE NOT USER FRIENDLY

IBM gives you only a set of basic MVS commands to monitor your coupling facility configuration. To obtain the information you need, you should use the different forms of the D XCF,... command.

For example, to get information about your DB2 lock structure, you will use the following command:

```
D XCF,STR,STRNAME=DB2M_LOCK1
```

An example of the output is shown below:

```
IXC360I  15.05.09  DISPLAY XCF 173
STRNAME: DB2M_LOCK1
 STATUS: ALLOCATED
  POLICY SIZE    : 49152 K
  POLICY INITSIZE: 32768 K
  REBUILD PERCENT: 1
  PREFERENCE LIST: ACF3     ACF1     L2A3XCF
  EXCLUSION LIST IS EMPTY

 ACTIVE STRUCTURE
 ————————
  ALLOCATION TIME: 08/16/1999 17:40:51
  CFNAME         : ACF3
  COUPLING FACILITY: 009672.IBM.51.000000069150
                     PARTITION: 1   CPCID: 00
  ACTUAL SIZE    : 32768 K
  STORAGE INCREMENT SIZE: 256 K
```

```
  VERSION        : B11Ø87A3 AEC8A3Ø1
  XCF GRPNAME    : IXCLOØØ5
  DISPOSITION    : KEEP
  ACCESS TIME    : Ø
  MAX CONNECTIONS: 7
  ú CONNECTIONS  : 2

 CONNECTION NAME  ID VERSION  SYSNAME  JOBNAME  ASID STATE
 ———————— — ———— ———— ———— —— ————————
 DXRM$$$$$FRLMØØ2 Ø2 ØØØ2ØØ22 FMVS     DB2FIRLM ØØ9D ACTIVE
 DXRM$$$$$IRLMØØ1 Ø1 ØØØ1ØØ16 PROD     DB2AIRLM Ø191 ACTIVE
```

Or, if you need to get information about the whole coupling facility, you will have to use the following command:

```
D XCF,CF,CFNAME=ACF3
```

An example of the output is shown below:

```
IXC362I  15.Ø6.32  DISPLAY XCF 515
CFNAME: ACF3
COUPLING FACILITY     :   ØØ9672.IBM.51.ØØØØØØØ69150
                            PARTITION: 1   CPCID: ØØ
  POLICY DUMP SPACE SIZE:   1Ø24Ø K
  ACTUAL DUMP SPACE SIZE:   1Ø24Ø K
  STORAGE INCREMENT SIZE:   256 K

  CONNECTED SYSTEMS:
  FMVS     PROD

  STRUCTURES:
  DB2M_GBPØ         DB2M_GBP1         DB2M_GBP1Ø        DB2M_LOCK1
  DB2M_SCA          IEFAUTOS          ISTGENERIC        IXC_PATH3
  SYSTEM_LOGREC     SYSTEM_OPERLOG
```

These commands are not very ‘friendly’ for systems programmers or operators. An additional drawback with these commands is that they do not provide any performance information about coupling facility activity.

Another approach is to use a third-party product (for example Boole & Babbage, Candle, etc). But these products do not always provide all the information you need and want. So this is why I decided to write my own utility to monitor coupling facilities and structures.

MY COUPLING FACILITY MONITOR

My coupling facility monitor is implemented as an ISPF application. The main panel gives you two choices:

```
──────────  Coupling Facility Monitor ──────────────
 OPTION  ===>
                                                    USERID    - I990557
                                                    TIME      - 15:25

    C  Coupling Facility  - Coupling Facilities Display
    S  Structure          - CF Structures Display
```

COUPLING FACILITIES PANELS

The first choice (c) allows you to list coupling facilities connected to your MVS system:

```
─────────── Coupling Facilities info ──── Row 1 to 2 of 2
 COMMAND ===>                                    SCROOL ===> PAGE

 ──────────────────────────────────────-
    CFNAME: ACF1
      Node: 009674.IBM.51.000000068216 PARTITION: 01 CPCID: 00
      CFLEVEL: 4          Storage Increment:     256 k   Volatile: N
      Storage Usage:
        Total:  119808  k
        Dump :   10240  k
        Free :  105472  k
 ──────────────────────────────────────-
    CFNAME: ACF3
      Node: 009672.IBM.51.000000069150 PARTITION: 01 CPCID: 00
      CFLEVEL: 6          Storage Increment:     256 k   Volatile: Y
      Storage Usage:
        Total:  250368  k
        Dump :   10240  k
        Free :  141568  k
```

This panel gives you information on storage usage of your coupling facilities, microcode CFLEVEL, and volatility status. Then, you can select one of the coupling facilities to get the list of its allocated structures.

At that time, you get real-time statistics on structures storage usage. When you hit ‘Enter’, the panel is automatically refreshed. You can also select one structure to get more detailed information:

```
———————————— Structure Detail ————————————
  COMMAND ===>                                          SCROOL ===> PAGE

  CFNAME: ACF3

    STRNAME:       DB2M_LOCK1            Storage:              32768 k

    Type:                          LOCK  Response Time:
                                               SYNC:               47 Ês
    List/Directory Entries:                    ASYNC:             N/A Ês
         Max:
         Used:                             Access Rate:
    List Headers:                              SYNC:            25Ø.6 /sec
    Data Elements:                             ASYNC:             Ø.Ø /sec
         Max:
         Used:
    Lock Entries:
         Max:                  8388608
         Used:                   88149
    Lock/Element Size:               2
```

You obtain response time and access rate information on the selected structure. The figures are also refreshed when you hit ‘Enter’.

Structures panels

The second choice (S) of the main panel allows you to list structures connected to your MVS system:

```
———————————— Structures Info ——————— Row 1 to 3 of 11
    COMMAND ===>                                        SCROOL ===> PAGE

_____________________________________

    STRNAME: DB2M_GBPØ          Status: ALLOCATED
         CF: ACF3          ØØ9672.IBM.51.ØØØØØØØ69150 PARTITION: Ø1 CPCID: ØØ
         Preference List: ACF3     ACF1     L2A3XCF
         Exclusion List: LIST IS EMPTY
         Initsize:   32768 K - Size:   49152 K - Rebuild Pct:       1
_____________________________________
    STRNAME: DB2M_GBP1          Status: ALLOCATED
         CF: ACF3          ØØ9672.IBM.51.ØØØØØØØ69150 PARTITION: Ø1 CPCID: ØØ
         Preference List: ACF3     ACF1     L2A3XCF
         Exclusion List: LIST IS EMPTY
         Initsize:    8192 K - Size:   12288 K - Rebuild Pct:       1
_____________________________________
    STRNAME: DB2M_GBP1Ø         Status: ALLOCATED
         CF: ACF3          ØØ9672.IBM.51.ØØØØØØØ69150 PARTITION: Ø1 CPCID: ØØ
         Preference List: ACF3     ACF1     L2A3XCF
         Exclusion List: LIST IS EMPTY
         Initsize:    2Ø48 K - Size:    4Ø96 K - Rebuild Pct:       1
```

This panel provides structures definition data specified in active CFRM policy (SIZE, INITSIZE, PREFRENCE LIST, etc). You can select one structure to get information about active connections:

```
———————————— Structures Info ———————— Row 1 to 2 of 2
  COMMAND ===>                                          SCROOL ===> PAGE

    STRNAME: DB2M_GBP0          STATUS: ALLOCATED
     CF: ACF3          009672.IBM.51.000000069150 PARTITION: 01 CPCID: 00
     Prefrence List: ACF3     ACF1     L2A3XCF
     Exclusion List: LIST IS EMPTY
     Initsize:   32768 k - Size:   49152 k - Rebuild pct:       1

                                                              Allow
 Connection Name  Sysname  Jobname  Status           Alter    Rebuild
 ————————————————————————————————————-
 DB2_DB2A         PROD     DB2ADBM1 ACTIVE           Y        Y
 DB2_DB2F         FMVS     DB2FDBM1 ACTIVE           Y        Y
 ************************** BOTTOM OF DATA ****************************
```

This panel provides information about the 'ability' to alter and rebuild the structure.

IMPLEMENTING THE COUPLING FACILITY MONITOR

The monitor is designed around two Assembler programs (IXCCFIS and IXCSTIS). These two programs use two Sysplex services macros – IXLMG and IXCQUERY. You can get detailed information about these macros in *OS/390 MVS Programming: Sysplex Services Reference (GC28-1772)*.

IXCCFIS SOURCE

This first program uses the IXLMG macro to retrieve information about coupling facilities connected to your MVS system. The IXLMG macro allows an authorized caller to request measurement data related to the use of a coupling facility. Structures in the IXLYAMDA mapping macro provide the format for the returned data.

- IXLYAMDAREA – maps the header record.
- XLYAMDCF – maps coupling facility information.
- IXLYAMDSTRL – maps coupling facility list structure information.

- IXLYAMDSTRC – maps coupling facility cache structure information.

IXCCFIS

```
IXCCFIS  CSECTIXCCFIS
         AMODE 31IXCCFIS
         RMODE ANY*
         USING IXCCFIS,R15*
         SAVE  (14,12)
         LR    R12,R15
         DROP  R15
*
         LA    R11,2Ø48(R12)
         LA    R11,2Ø48(R11)
         USING IXCCFIS,R12,R11
         GETMAIN R,LV=WORKL
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
         USING DSECT,R13
*
* CREATE ISPF VARIABLES
* =====================
         CALL  ISPLINK,(VDEFINE,FZTDSELS,ZTDSELS,CHAR,L4),VL
         CALL  ISPLINK,(VDEFINE,FSELECT,SELECT,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FCFNAME,CFNAME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FCFNODE,CFNODE,CHAR,L54),VL
         CALL  ISPLINK,(VDEFINE,FTSPACE,TSPACE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FFSPACE,FSPACE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FDSPACE,DSPACE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FCFLEVEL,CFLEVEL,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTINC,STINC,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FVOL,VOL,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FSTRNAME,STRNAME,CHAR,L16),VL
         CALL  ISPLINK,(VDEFINE,FSTSIZE,STSIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTTOT,STTOT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTTYPE,STTYPE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLENT,STLENT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLENTU,STLENTU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDELM,STDELM,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDELMU,STDELMU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLSIZE,STLSIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTLHD,STLHD,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDENT,STDENT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTDENTU,STDENTU,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTSTIME,STSTIME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTATIME,STATIME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTSCNT,STSCNT,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTACNT,STACNT,CHAR,L8),VL
*        CREATE AND SORT ISPF TABLES
```

```
*
* CREATE ISPF TABLE
* ==================
         CALL ISPLINK,(TBCREATE,FSTABLE,,NAMELIST,NOWRITE,REPLACE),VL
         MVC  SORTKEY,FCFNAME                   DEFAULT SORT
         MVC  SORTTYPE,CHARASND
*        CALL ISPLINK,(TBSORT,FSTABLE,SORTPARM),VL
*
* PROGRAM CODE
* =============
         LA    R9,ANSAREA
         USING IXLYAMDAREA,R9
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT...
         LTR   R1Ø,R1Ø
         BZ    RETURN
         USING IXLYAMDCF,R1Ø
*
* ========
* CF PANEL
* ========
*
LOOPCF   EQU   *
*                                              SET DEFAULT VALUES
         MVC   CFNAME,IXLYAMDCF_CFNAME
         MVC   VOL,=CL1'Y'
         TM    IXLYAMDCF_FLAGS,IXLYAMDCF_VOLATILE
         BNZ   FLAGV
         MVC   VOL,=CL1'N'
FLAGV    EQU   *
         LA    R7,IXLYAMDCF_ND
         USING NDE,R7
         MVC   CFNODE(6),NDETYPE
         MVI   CFNODE+6,C'.'
         MVC   CFNODE+7(3),NDEMFG
         MVI   CFNODE+1Ø,C'.'
         MVC   CFNODE+11(2),NDEPLANT
         MVI   CFNODE+13,C'.'
         MVC   CFNODE+14(12),NDESEQUENCE
         MVC   CFNODE+27(1Ø),=C'PARTITION:'
         XR    R8,R8
         IC    R8,NDEPARTITION                * PARTITION
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+38(2),DOUBLE+1
         MVC   CFNODE+41(6),=C'CPCID:'
         XR    R8,R8
         IC    R8,NDECPCID                    * CPCID
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+48(2),DOUBLE+1
```

```
         DROP  R7
         L     R8,IXLYAMDCF_TS                TOTAL SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   TSPACE,MASK
         ED    TSPACE(8),DOUBLE+4
         L     R8,IXLYAMDCF_FS                FREE  SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   FSPACE,MASK
         ED    FSPACE(8),DOUBLE+4
         SR    R8,R8
         LH    R8,IXLYAMDCF_STGI              STORAGE INCREMENT
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   STINC,MASK
         ED    STINC(8),DOUBLE+4
         L     R8,IXLYAMDCF_TDS               DUMP SPACE
         MH    R8,=H'4'
         CVD   R8,DOUBLE
         MVC   DSPACE,MASK
         ED    DSPACE(8),DOUBLE+4
         L     R8,IXLYAMDCF_CFLEVEL           CFLEVEL
         CVD   R8,DOUBLE
         MVC   CFLEVEL,MASK
         ED    CFLEVEL(8),DOUBLE+4
*
* ADD A NEW ROW
* =============
         CALL  ISPLINK,(TBADD,FSTABLE),VL  ADD DATA INTO TABLE
         L     R1Ø,IXLYAMDCF_CFNEXT
         LTR   R1Ø,R1Ø                        LAST CF ENTRY?
         BNZ   LOOPCF
DISPP1   EQU   *
         CALL  ISPLINK,(TBSORT,FSTABLE,SORTPARM),VL  SORT TABLE
         CALL  ISPLINK,(TBTOP,FSTABLE),VL  POINT TO TOP OF TABLE
         CALL  ISPLINK,(TBDISPL,FSTABLE,FSPANEL),VL  DISPLAY TABLE
         C     R15,=F'8'               HAS PF3   BEEN HIT (R15 = 8)?
         BE    RETURN                  NO, TERMINATE PROGRAM
         CLC   SELECT,=CL1'S'
         BNE   NOSELECT
* ===============
* CF / STR PANEL
* ===============
FLAG1Ø   EQU   *
         CALL ISPLINK,(TBCREATE,FSTABLES,,NAMELISS,NOWRITE,REPLACE),VL
         MVC  SORTKEYS,FSTRNAME                   DEFAULT SORT
         MVC  SORTTYPS,CHARASND
*        CALL ISPLINK,(TBSORT,FSTABLE,SORTPARS),VL
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT...
         USING IXLYAMDCF,R1Ø
```

```
LOOPCFS  EQU   *
         CLC   CFNAME,IXLYAMDCF_CFNAME
         BE    GOTCF
         L     R1Ø,IXLYAMDCF_CFNEXT
         B     LOOPCFS
GOTCF    EQU   *
         SR    R5,R5                       TOTAL STRUCTURES SIZE
         L     R4,IXLYAMDCF_STR...
         LTR   R4,R4
         BZ    FLAGØ2
LOOPSTR  EQU   *
         BAL   R6,GETSTR
         USING IXLYAMDSTRL,R4
         CALL  ISPLINK,(TBADD,FSTABLES),VL  ADD DATA INTO TABLE
         L     R4,IXLYAMDSTRL_STRNEXT
         LTR   R4,R4                       LAST ENTRY?
         BNZ   LOOPSTR
FLAGØ2   EQU   *
         CVD   R5,DOUBLE
         MVC   STTOT,MASK
         ED    STTOT(8),DOUBLE+4
         CALL  ISPLINK,(TBSORT,FSTABLES,SORTPARS),VL  SORT TABLE
         CALL  ISPLINK,(TBTOP,FSTABLES),VL  POINT TO TOP OF TABLE
REDISPP2 CALL  ISPLINK,(TBDISPL,FSTABLES,FSPANELS),VL  DISPLAY TABLE
         C     R15,=F'8'                  HAS PF3   BEEN HIT (R15 = 8)?
         BE    DISPP1                     NO, TERMINATE PROGRAM
         MVC   OTIME,=2D'Ø'
         MVC   OSTIMEC,=F'Ø'
         MVC   OSTIMES,=D'Ø'
         MVC   OATIMEC,=F'Ø'
         MVC   OATIMES,=D'Ø'
         MVC   STSTIME,=CL8'N/A'
         MVC   STATIME,=CL8'N/A'
         MVC   STSCNT,=CL8'N/A'
         MVC   STACNT,=CL8'N/A'
         CLC   SELECT,=CL1'S'
         BE    REDISPP3
         CALL  ISPLINK,(TBCLOSE,FSTABLES),VL  CLOSE TABLE
         B     FLAG1Ø
* ================
* STR DETAIL PANEL
* ================
REDISPP3 EQU   *
         STCKSYNC TOD=NTIME              GET REQUEST TIME
         BAL   R6,IXLMG
         L     R1Ø,IXLYAMDAREA_CFENT...
         USING IXLYAMDCF,R1Ø
LOOPCFSD EQU   *
         CLC   CFNAME,IXLYAMDCF_CFNAME
         BE    GOTCFD
         L     R1Ø,IXLYAMDCF_CFNEXT
         B     LOOPCFSD
GOTCFD   EQU   *
```

```
         L     R4,IXLYAMDCF_STR...
         LTR   R4,R4
         BZ    FLAGØ2
LOOPSTRD EQU   *
         USING IXLYAMDSTRL,R4
         CLI   IXLYAMDSTRL_TYPE,X'21'
         BNE   STCACHED                          * CACHE STRUCTURE?
         CLC   STRNAME,IXLYAMDSTRL_STRNAME
         BE    GOTSTD
         B     GETSTDE
STCACHED EQU   *
         USING IXLYAMDSTRC,R4
CLC   STRNAME,IXLYAMDSTRC_STRNAME
         BE    GOTSTD
GETSTDE  EQU   *
         L     R4,IXLYAMDSTRC_STRNEXT
         LTR   R4,R4                         LAST ENTRY?
         BNZ   LOOPSTRD
GOTSTD   EQU   *
         BAL   R6,GETSTR
         CALL  ISPLINK,(DISPLAY,STPANEL),VL
         C     R15,=F'8'               HAS PF3   BEEN HIT (R15 = 8)?
         BE    REDISPP2
         B     REDISPP3
NOSELECT EQU   *
         B     DISPP1
ERROR1   EQU   *
         CALL  ISPLINK,(SETMSG,MSG1),VL
         B     RETURN
ERROR2   EQU   *
         CALL  ISPLINK,(SETMSG,MSG2),VL
         B     RETURN
*
* RETURN
* ======
RETURN   EQU   *
*        CALL  ISPLINK,(TBCLOSE,FSTABLE),VL  CLOSE TABLE
         L     R13,4(R13)
         L     R1,8(R13)
         FREEMAIN R,LV=WORKL,A=(R1)
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
IXLMG    EQU   *
         AUTHON                                 AUTH SVC
         MODESET KEY=ZERO
         IXLMG DATAAREA=ANSAREA,                                       X
               DATALEN=ANSLEN,                                         X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         LTR   R15,R15
         BNZ   ERROR2
         MODESET KEY=NZERO
```

```
         AUTHOFF                                  RESET AUTH
         BR    R6
GETSTR   EQU   *
         MVC   STLENT,=CL8' '
         MVC   STLENTU,=CL8' '
         MVC   STDELM,=CL8' '
         MVC   STDELMU,=CL8' '
         MVC   STLSIZE,=CL8' '
         MVC   STLHD,=CL8' '
         MVC   STDENT,=CL8' '
         MVC   STDENTU,=CL8' '
         USING IXLYAMDSTRL,R4
         CLI   IXLYAMDSTRL_TYPE,X'21'
         BNE   STCACHE                          * CACHE STRUCTURE?
         MVC   STRNAME,IXLYAMDSTRL_STRNAME
*                                                RESPONSE TIME
*                                                SYNC
         MVC   NSTIMEC,IXLYAMDSTRL_SYNCTIMECOUNT
         MVC   NSTIMES,IXLYAMDSTRL_SYNCSUMTIME
         MVC   NATIMEC,IXLYAMDSTRL_ASYNCTIMECOUNT
         MVC   NATIMES,IXLYAMDSTRL_ASYNCSUMTIME
         CLC   OSTIMEC,=F'Ø'                    FIRST TIME?
         BE    ACCESSL
         L     R3,NSTIMES+4
         L     R2,OSTIMES+4
         SR    R3,R2
         BNH   BADS
         XR    R2,R2
         L     RØ,NSTIMEC
         S     RØ,OSTIMEC
         BNH   BADS
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STSTIME,MASK
         ED    STSTIME(8),DOUBLE+4
BADS     EQU   *
         L     R3,NATIMES+4
         L     R2,OATIMES+4
         SR    R3,R2
         BNH   BADA
         XR    R2,R2
         L     RØ,NATIMEC
         S     RØ,OATIMEC
         BNH   BADA
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STATIME,MASK
         ED    STATIME(8),DOUBLE+4
BADA     EQU   *
ACCESSL  EQU   *
*                                             ACCESS RATE
         STCKCONV STCKVAL=OTIME,CONVVAL=TWORK,TIMETYPE=BIN
         L     R2,TWORK
```

```
         STCKCONV STCKVAL=NTIME,CONVVAL=TWORK,TIMETYPE=BIN
         L     RØ,TWORK
         SR    RØ,R2
         BNH   BADDELTA
         CLC   OSTIMEC,=F'Ø'                 FIRST TIME?
         BE    BADDELTA
         L     R3,NSTIMEC
         S     R3,OSTIMEC
         L     R1,=F'1ØØØ'     XXX.X REQ/SEC
         SR    R2,R2
         MR    R2,R1
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STSCNT,MASKRATE
         ED    STSCNT(8),DOUBLE+5
         L     R3,NATIMEC
         S     R3,OATIMEC
         L     R1,=F'1ØØØ'     XXX.X REQ/SEC
         SR    R2,R2
         MR    R2,R1
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STACNT,MASKRATE
         ED    STACNT(8),DOUBLE+5
BADDELTA EQU   *
         MVC   OTIME,NTIME
         MVC   OSTIMEC,NSTIMEC
         MVC   OSTIMES,NSTIMES
         MVC   OATIMEC,NATIMEC
         MVC   OATIMES,NATIMES
         CLI   IXLYAMDSTRL_TTY,IXLYAMDA_LIST    * LIST STRUCTURE?
         BE    STLIST
         CLI   IXLYAMDSTRL_TTY,IXLYAMDA_LOCK    * LOCK STRUCTURE?
         BE    STLOCK
STLIST   EQU   *
         MVC   STTYPE,=CL8'LIST'
*                                            DIRECTORY ENTRIES
         L     R8,IXLYAMDSTRL_MLSEC
         CVD   R8,DOUBLE
         MVC   STDENT,MASK
         ED    STDENT(8),DOUBLE+4
         L     R8,IXLYAMDSTRL_LSEC
         CVD   R8,DOUBLE
         MVC   STDENTU,MASK
         ED    STDENTU(8),DOUBLE+4
*                                             LIST ENTRIES
         L     R8,IXLYAMDSTRL_MLSELC
         CVD   R8,DOUBLE
         MVC   STDELM,MASK
         ED    STDELM(8),DOUBLE+4
         L     R8,IXLYAMDSTRL_LSELC
         CVD   R8,DOUBLE
         MVC   STDELMU,MASK
```

```
         ED    STDELMU(8),DOUBLE+4
*                                                LIST HEADERS
         L     R8,IXLYAMDSTRL_LC
         CVD   R8,DOUBLE
         MVC   STLHD,MASK
         ED    STLHD(8),DOUBLE+4
*                                                LIST ELM SIZE
*                                                256*(2**LELX)
         SR    R8,R8
         IC    R8,IXLYAMDSTRL_LELX
         LA    R3,1
         LTR   R8,R8
         BZ    MUT1
EXP1     EQU   *
         SLL   R3,1
         BCT   R8,EXP1
MUT1     EQU   *
         MH    R3,=H'256'
         CVD   R3,DOUBLE
         MVC   STLSIZE,MASK
         ED    STLSIZE(8),DOUBLE+4
         B     FLAGØ1
STLOCK   EQU   *
         MVC   STTYPE,=CL8'LOCK'
         L     R8,IXLYAMDSTRL_NLE                 LOCK ENTRIES
         CVD   R8,DOUBLE
         MVC   STLENT,MASK
         ED    STLENT(8),DOUBLE+4
         L     R8,IXLYAMDSTRL_NLTEC
         CVD   R8,DOUBLE
         MVC   STLENTU,MASK
         ED    STLENTU(8),DOUBLE+4
*                                                LOCK SIZE
*                                                (2**LTECH)
         SR    R8,R8
         IC    R8,IXLYAMDSTRL_LTECH
         LA    R3,1
         LTR   R8,R8
         BZ    MUT2
EXP2     EQU   *
         SLL   R3,1
         BCT   R8,EXP2
MUT2     EQU   *
         CVD   R3,DOUBLE
         MVC   STLSIZE,MASK
         ED    STLSIZE(8),DOUBLE+4
FLAGØ1   EQU   *
*===
*        MVC   WTO(WTOL),WTOC
*        MVC   WTO+Ø4(16),STRNAME
*        MVC   WTO+25(4),IXLYAMDSTRL_STRNEXT
*        WTO   MF=(E,WTO)
*===
```

```
*
         L     R8,IXLYAMDSTRL_SS              STRUCTURE SIZE
         MH    R8,=H'4'
         AR    R5,R8
         CVD   R8,DOUBLE
         MVC   STSIZE,MASK
         ED    STSIZE(8),DOUBLE+4
         B     GETSTEND
STCACHE  EQU   *
         USING IXLYAMDSTRC,R4
         MVC   STRNAME,IXLYAMDSTRC_STRNAME
         MVC   STTYPE,=CL8'CACHE'
         MVC   NSTIMEC,IXLYAMDSTRC_SYNCTIMECOUNT
         MVC   NSTIMES,IXLYAMDSTRC_SYNCSUMTIME
         MVC   NATIMEC,IXLYAMDSTRC_ASYNCTIMECOUNT
         MVC   NATIMES,IXLYAMDSTRC_ASYNCSUMTIME
         CLC   OSTIMEC,=F'Ø'                  FIRST TIME?
         BE    ACCESSC
         L     R3,NSTIMES+4
         L     R2,OSTIMES+4
         SR    R3,R2
         BNH   BADSC
         XR    R2,R2
         L     RØ,NSTIMEC
         S     RØ,OSTIMEC
         BNH   BADSC
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STSTIME,MASK
         ED    STSTIME(8),DOUBLE+4
BADSC    EQU   *
         L     R3,NATIMES+4
         L     R2,OATIMES+4
         SR    R3,R2
         BNH   BADAC
         XR    R2,R2
         L     RØ,NATIMEC
         S     RØ,OATIMEC
         BNH   BADAC
         DR    R2,RØ
         CVD   R3,DOUBLE
         MVC   STATIME,MASK
         ED    STATIME(8),DOUBLE+4
BADAC    EQU   *
ACCESSC  EQU   *
*                                           ACCESS RATE
         STCKCONV STCKVAL=OTIME,CONVVAL=TWORK,TIMETYPE=BIN
         L     R2,TWORK
         STCKCONV STCKVAL=NTIME,CONVVAL=TWORK,TIMETYPE=BIN
         L     RØ,TWORK
         SR    RØ,R2
         BNH   BADDELT
         CLC   OSTIMEC,=F'Ø'                  FIRST TIME?
```

```
         BE    BADDELT
         L     R3,NSTIMEC
         S     R3,OSTIMEC
         L     R1,=F'1000'     XXX.X REQ/SEC
         SR    R2,R2
         MR    R2,R1
         DR    R2,R0
         CVD   R3,DOUBLE
         MVC   STSCNT,MASKRATE
         ED    STSCNT(8),DOUBLE+5
         L     R3,NATIMEC
         S     R3,OATIMEC
         L     R1,=F'1000'     XXX.X REQ/SEC
         SR    R2,R2
         MR    R2,R1
         DR    R2,R0
         CVD   R3,DOUBLE
         MVC   STACNT,MASKRATE
         ED    STACNT(8),DOUBLE+5
BADDELT  EQU   *
         MVC   OTIME,NTIME
         MVC   OSTIMEC,NSTIMEC
         MVC   OSTIMES,NSTIMES
         MVC   OATIMEC,NATIMEC
         MVC   OATIMES,NATIMES
*                                          STRUCTURE SIZE
         L     R8,IXLYAMDSTRC_SS
         MH    R8,=H'4'
         AR    R5,R8
         CVD   R8,DOUBLE
         MVC   STSIZE,MASK
         ED    STSIZE(8),DOUBLE+4
*                                          DIRECTORY ENTRIES
         L     R8,IXLYAMDSTRC_TDEC
         CVD   R8,DOUBLE
         MVC   STDENT,MASK
         ED    STDENT(8),DOUBLE+4
         L     R8,IXLYAMDSTRC_TSCC
         CVD   R8,DOUBLE
         MVC   STDENTU,MASK
         ED    STDENTU(8),DOUBLE+4
*                                          DATA ELEMENT ENTRIES
         L     R8,IXLYAMDSTRC_TDAEC
         CVD   R8,DOUBLE
         MVC   STDELM,MASK
         ED    STDELM(8),DOUBLE+4
         L     R8,IXLYAMDSTRC_TCDEC
         CVD   R8,DOUBLE
         MVC   STDELMU,MASK
         ED    STDELMU(8),DOUBLE+4
*                                          DATA ELM SIZE
*                                          256*(2**LTECH)
         SR    R8,R8
```

```
         IC    R8,IXLYAMDSTRC_DAEX
         LA    R3,1
         LTR   R8,R8
         BZ    MUT3
EXP3     EQU   *
         SLL   R3,1
         BCT   R8,EXP3
MUT3     EQU   *
         MH    R3,=H'256'
         CVD   R3,DOUBLE
         MVC   STLSIZE,MASK
         ED    STLSIZE(8),DOUBLE+4
GETSTEND EQU   *
         BR    R6
*
* ISPF MESSAGES
* =============
MSG1     DC    CL8'IXCØØ1E'
MSG2     DC    CL8'IXCØØ2E'
*
* ISPF OBJECTS (PANELS, SKELETONS...)
* ===================================
FSPANEL  DC    CL8'IXCCF'                    ISPF PANEL NAME
FSPANELS DC    CL8'IXCCFST'                  ISPF PANEL NAME
FSSKEL   DC    CL8'IXCCFISS'                 ISPF SKELETON
FSMOUT   DC    CL8'IXCCFISO'                 FT OUTPUT MEMBER
FSTABLE  DC    CL8'FSTABLE'                  TABLE NAME
FSTABLES DC    CL8'FSTABLES'                 TABLE NAME
STPANEL  DC    CL8'IXCCFSTD'                 ISPF PANEL NAME
*
* ISPF VARIABLES
* ==============
FZTDSELS DC    CL8'ZTDSELS'
ZTDSELS  DS    CL4
FSELECT  DC    CL8'SELECT'
SELECT   DS    CL1
FCFNAME  DC    CL8'CFNAME'
CFNAME   DS    CL8
FCFNODE  DC    CL8'CFNODE'
CFNODE   DS    CL54
FTSPACE  DC    CL8'TSPACE'
TSPACE   DS    CL8
FFSPACE  DC    CL8'FSPACE'
FSPACE   DS    CL8
FDSPACE  DC    CL8'DSPACE'
DSPACE   DS    CL8
FCFLEVEL DC    CL8'CFLEVEL'
CFLEVEL  DS    CL8
FSTINC   DC    CL8'STINC'
STINC    DS    CL8
FVOL     DC    CL8'VOL'
VOL      DS    CL1
FSTRNAME DC    CL8'STRNAME'
```

```
STRNAME  DS    CL16
FSTSIZE  DC    CL8'STSIZE'
STSIZE   DS    CL8
FSTTOT   DC    CL8'STTOT'
STTOT    DS    CL8
FSTTYPE  DC    CL8'STTYPE'
STTYPE   DS    CL8
FSTLENT  DC    CL8'STLENT'
STLENT   DS    CL8
FSTLENTU DC    CL8'STLENTU'
STLENTU  DS    CL8
FSTDELM  DC    CL8'STDELM'
STDELM   DS    CL8
FSTDELMU DC    CL8'STDELMU'
STDELMU  DS    CL8
FSTLSIZE DC    CL8'STLSIZE'
STLSIZE  DS    CL8
FSTLHD   DC    CL8'STLHD'
STLHD    DS    CL8
FSTDENT  DC    CL8'STDENT'
STDENT   DS    CL8
FSTDENTU DC    CL8'STDENTU'
STDENTU  DS    CL8
FSTSTIME DC    CL8'STSTIME'
STSTIME  DS    CL8
FSTATIME DC    CL8'STATIME'
STATIME  DS    CL8
FSTSCNT  DC    CL8'STSCNT'
STSCNT   DS    CL8
FSTACNT  DC    CL8'STACNT'
STACNT   DS    CL8
*                                          VARIABLES LIST
NAMELIST DC    CLØ7Ø'(CFNAME CFNODE TSPACE DSPACE DSPACE CFLEVEL STINC X
               FSPACE VOL)'
NAMELISS DC    CL2ØØ'(STRNAME STSIZE STTYPE STLENT STDELM STLSIZE STLHDX
                STDENT STDELM STDENTU STDELMU STLENTU STSCNT STACNT)'
*                                          SORT PARMS
SORTPARM DS    ØCL12
         DC    CL1'('
SORTKEY  DS    CL8
         DC    CL1','
SORTTYPE DS    CL3
         DC    CL1')'
SORTPARS DS    ØCL12
         DC    CL1'('
SORTKEYS DS    CL8
         DC    CL1','
SORTTYPS DS    CL3
         DC    CL1')'
*
* ISPF CONSTANTS
* ==============
DISPLAY  DC    CL7'DISPLAY'
```

```
VDEFINE  DC    CL7'VDEFINE'
TBADD    DC    CL5'TBADD'
TBCLOSE  DC    CL7'TBCLOSE'
TBCREATE DC    CL8'TBCREATE'
TBDISPL  DC    CL7'TBDISPL'
TBSORT   DC    CL6'TBSORT'
TBTOP    DC    CL5'TBTOP'
ORDER    DC    CL5'ORDER'
NOWRITE  DC    CL7'NOWRITE'
REPLACE  DC    CL7'REPLACE'
SETMSG   DC    CL6'SETMSG'
FTOPEN   DC    CL6'FTOPEN'
FTINCL   DC    CL6'FTINCL'
FTCLOSE  DC    CL7'FTCLOSE'
SAVE     DC    CL4'SAVE'
RESTORE  DC    CL7'RESTORE'
CHARASND DC    CL3'C,A'
NUMRDSND DC    CL3'N,D'
*                                          TYPES
*                                          ——
CHAR     DC    CL4'CHAR'
*                                          LENGTH
*                                          ———
L1       DC    F'1'
L4       DC    F'4'
L8       DC    F'8'
L16      DC    F'16'
L54      DC    F'54'
*                                          FIELDS
* WTO TO DEBUG
WTOC     WTO   '                                                  X
                                               ',MF=L,ROUTCDE=(11)
WTOL     EQU   *-WTOC                       LENGTH OF MACRO EXPANSION
WTO      DS    CL(WTOL)
         LTORG
*
* PROGRAM DATA AREAS
* =================
MASK     DC    X'4020202020202120'
MASKRATE DC    X'4040202021204B20'
NTIME    DC    D'0'
OTIME    DC    D'0'
TWORK    DC    2D'0'
NSTIMEC  DC    F'0'
NSTIMES  DC    D'0'
OSTIMEC  DC    F'0'
OSTIMES  DC    D'0'
NATIMEC  DC    F'0'
NATIMES  DC    D'0'
OATIMEC  DC    F'0'
OATIMES  DC    D'0'
ANSLEN   DC    F'40960'                      10*4096
ANSAREA  DS    10CL4096
```

```
DSECT    DSECT
SAVEAREA DS    18F                              SAVEAREA
RETCODE  DS    F
RSNCODE  DS    F
DOUBLE   DS    D
LGDSECT  EQU   *-DSECT
WORKL    EQU   LGDSECT                          LENGTH OF WORAREA
         IXLYAMDA
         IXLYNDE
         REGISTER
         END
```

IXCSTIS SOURCE

This second program uses the IXCQUERY macro to retrieve information about structures allocated in coupling facilities.

The IXCQUERY macro allows any authorized caller to request information about the resources the cross-system coupling facility (XCF) manages. The REQINFO parameter determines whether the information is about XCF groups, systems in the Sysplex, the Sysplex itself, coupling facility resources, or information related to the automatic restart manager.

When using the REQINFO=STR_ALLDATA parameter, IXCQUERY returns information about all coupling facility structures.

You need to use the ANSAREA parameter to tell XCF where to return the information, and ANSLEN to tell XCF the length of the answer area. Sections in the IXCYQUAA mapping macro provide the format for the returned data. QUAHDR maps the offset and length of the other record types.

QUACFSTR maps information about coupling facility structures allocated in a coupling facility. QUASTR maps the coupling facility structure record.

```
IXCSTIS  CSECTIXCSTIS
         AMODE 31IXCSTIS
         RMODE ANY*
         SAVE  (14,12)
         BALR  R12,Ø
         USING *,R12
         GETMAIN R,LV=WORKL
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
```

```
         USING DSECT,R13
*        CREATE ISPF VARIABLES
         CALL  ISPLINK,(VDEFINE,FSELECT,SELECT,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FSKEY,SKEY,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSTRNAME,STRNAME,CHAR,L16),VL
         CALL  ISPLINK,(VDEFINE,FALLOC,ALLOC,CHAR,L13),VL
         CALL  ISPLINK,(VDEFINE,FPENDING,PENDING,CHAR,L21),VL
         CALL  ISPLINK,(VDEFINE,FCFNAME,CFNAME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FUM,UM,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FCFNODE,CFNODE,CHAR,L54),VL
         CALL  ISPLINK,(VDEFINE,FPLCF,PLCF,CHAR,L71),VL
         CALL  ISPLINK,(VDEFINE,FXLCF,XLCF,CHAR,L67),VL
         CALL  ISPLINK,(VDEFINE,FINITSIZ,INITSIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSIZE,SIZE,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FREBUILD,REBUILDP,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FSYSNAME,SYSNAME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FJOBNAME,JOBNAME,CHAR,L8),VL
         CALL  ISPLINK,(VDEFINE,FCONNAME,CONNAME,CHAR,L16),VL
         CALL  ISPLINK,(VDEFINE,FCSTATUS,CSTATUS,CHAR,L16),VL
         CALL  ISPLINK,(VDEFINE,FALLOWA,ALLOWA,CHAR,L1),VL
         CALL  ISPLINK,(VDEFINE,FALLOWR,ALLOWR,CHAR,L1),VL
REDISPP1 EQU  *
*        CREATE AND SORT ISPF TABLE
         CALL ISPLINK,(TBCREATE,FSTABLE,,NAMELIST,NOWRITE,REPLACE),VL
         MVC  SORTKEY,FSTRNAME        SPECIFY DEFAULT SORT FIELD
         MVC  SORTTYPE,CHARASND       SPECIFY SORT DIRECTION
         LA    R2,ANSAREA
         USING QUAHDR,R2
         BAL   R6,IXCQUERY
         L     R3,QUAHSGOF
         LR    R4,R2
         AR    R4,R3
         USING QUASTR,R4
         USING QUASTRCF,R5
LOOPSTR  EQU   *
*                                              SET DEFAULT VALUES
         MVC   CFNAME,=CL8'N/A'
         MVC   ALLOC,=CL13'NOT ALLOCATED'
         MVC   PENDING,=CL21' '
         MVC   PLCF,=CL71' '
         MVC   XLCF,=CL67'LIST IS EMPTY'
         MVC   CFNODE,=CL54' '
         MVC   REBUILDP,=CL84'N/A'
         MVC   UM,=CL1' '
         MVC   STRNAME,QUASTRNAME
         L     R7,QUASTRINITSIZE              INIT SIZE
         MH    R7,=H'4'
         CVD   R7,DOUBLE
         MVC   INITSIZE,MASK
         ED    INITSIZE(8),DOUBLE+4
         L     R7,QUASTRSIZE                  SIZE
         MH    R7,=H'4'
```

```
         CVD   R7,DOUBLE
         MVC   SIZE,MASK
         ED    SIZE(8),DOUBLE+4
         SR    R7,R7
         IC    R7,QUASTRREBUILDPERCENT         REBUILD PERCENT
         LTR   R7,R7
         BZ    FLAGØ6
         CVD   R7,DOUBLE
         MVC   REBUILDP,MASK
         ED    REBUILDP(8),DOUBLE+4
FLAGØ6   EQU   *
         TM    QUASTRINHDW,QUASTRINHDWON       STRUCTURE ALLOCATED?
         BNO   FLAGØ2
FLAGØ1   EQU   *
         MVC   ALLOC,=CL13'ALLOCATED'
         LR    R5,R4
         L     R6,QUASTRCFO
         AR    R5,R6
         MVC   CFNAME,QUASTRCFNAME
         LA    R7,QUASTRCFND
         USING NDE,R7
         MVC   CFNODE(6),NDETYPE
         MVI   CFNODE+6,C'.'
         MVC   CFNODE+7(3),NDEMFG
         MVI   CFNODE+1Ø,C'.'
         MVC   CFNODE+11(2),NDEPLANT
         MVI   CFNODE+13,C'.'
         MVC   CFNODE+14(12),NDESEQUENCE
         MVC   CFNODE+27(1Ø),=C'PARTITION:'
         XR    R8,R8
         IC    R8,NDEPARTITION               * PARTITION
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+38(2),DOUBLE+1
         MVC   CFNODE+41(6),=C'CPCID:'
         XR    R8,R8
         IC    R8,NDECPCID                   * CPCID
         CVD   R8,DOUBLE
         UNPK  DOUBLE(3),DOUBLE+6(2)
         OI    DOUBLE+2,X'FØ'
         MVC   CFNODE+48(2),DOUBLE+1
         DROP  R
         TM    QUASTRSTATE1,QUASTRSTDPEND      POLICY CHANGE PENDING?
         BNO   FLAGØ2
FLAGØ3   EQU   *
         MVC   PENDING,=CL21'POLICY CHANGE PENDING'
FLAGØ2   EQU   *
*                                              PREFERENCE LIST
         L     R7,QUASTRPL$
         LTR   R7,R7                          ENTRIES?
         BZ    FLAGØ4                         NO
         LR    R7,R4
```

```
         L     R6,QUASTRPLO
         AR    R7,R6
         USING QUASTRPL,R7
         LA    R11,PLCF
LOOPPL   EQU   *
         MVC   Ø(8,R11),QUASTRPLNAME
         LA    R1Ø,9
         AR    R11,R1Ø
         TM    QUASTRPLTYP,QUATYPSTRPL_LAST     LAST PL ENTRY?
         BO    FLAGØ4
         AH    R7,QUASTRPLLEN
         B     LOOPPL
FLAGØ4   EQU   *
         CLC   ALLOC,=CL13'ALLOCATED'          STRUCTURE ALLOCATED?
         BNE   FLAGØ7
         CLC   PLCF(8),CFNAME
         BE    FLAGØ7
         MVC   UM,=CL1'*'
*                                               EXCLUSION  LIST
FLAGØ7   EQU   *
         L     R7,QUASTRXL$
         LTR   R7,R7                           ENTRIES?
         BZ    FLAGØ5                          NO
         LR    R7,R4
         L     R6,QUASTRXLO
         AR    R7,R6
         USING QUASTRXL,R7
         LA    R11,XLCF
LOOPXL   EQU   *
         MVC   Ø(16,R11),QUASTRXLNAME
*        MVC   Ø(5Ø,R11),QUASTRXLTYP
         LA    R1Ø,17
         AR    R11,R1Ø
         TM    QUASTRXLTYP,QUATYPSTRXL_LAST     LAST PL ENTRY?
         BO    FLAGØ5
         AH    R7,QUASTRXLLEN
         B     LOOPXL
FLAGØ5   EQU   *
*        ADD A NEW ROW
         CALL  ISPLINK,(TBADD,FSTABLE,,ORDER),VL  ADD DATA INTO TABLE
         TM    QUASTRTYP,QUATYPSTR_LAST         LAST STRUCTURE?
         BO    STLAST
         AH    R4,QUASTRLEN
         B     LOOPSTR
STLAST   EQU   *
         CALL  ISPLINK,(TBSORT,FSTABLE,SORTPARM),VL  SORT TABLE
DISPLAY  CALL  ISPLINK,(TBTOP,FSTABLE),VL  POINT TO TOP OF TABLE
REDISP   CALL  ISPLINK,(TBDISPL,FSTABLE,FSPANEL),VL  DISPLAY TABLE
         C     R15,=F'8'               PF3?
         BE    RETURN
         CLC   SELECT,=CL1'S'
         BE    REDISPP2
```

```
         CALL  ISPLINK,(TBCLOSE,FSTABLE),VL  CLOSE TABLE
         B     REDISPP1
REDISPP2 EQU   *
         CALL ISPLINK,(TBCREATE,FSTABLES,,NAMELISS,NOWRITE,REPLACE),VL
         MVC  SORTKES,FCONNAME        SPECIFY DEFAULT SORT FIELD
         MVC  SORTTYPS,CHARASND       SPECIFY SORT DIRECTION
         LA    R2,ANSAREA
         USING QUAHDR,R2
         BAL   R6,IXCQUERY
         L     R3,QUAHSGOF
         LR    R4,R2
         AR    R4,R3
         USING QUASTR,R4
         CLC   ALLOC,=CL13'ALLOCATED'          STRUCTURE ALLOCATED?
         BE    SSTR
DISPP3   EQU   *
         CALL  ISPLINK,(TBSORT,FSTABLES,SORTPARS),VL  SORT TABLE
         CALL  ISPLINK,(TBTOP,FSTABLES),VL  POINT TO TOP OF TABLE
         CALL  ISPLINK,(TBDISPL,FSTABLES,FSPANELS),VL  DISPLAY TABLE
         C     R15,=F'8'               PF3 ?
         BE    REDISP
         CALL  ISPLINK,(TBCLOSE,FSTABLES),VL  CLOSE TABLE
         B     REDISPP2
SSTR     EQU   *
         CLC   STRNAME,QUASTRNAME
         BNE   SSTRN
         L     R7,QUASTRUSERO
         AR    R7,R4
         USING QUASTRUSER,R7
LCONN    EQU   *
         MVC   CONNAME,QUASTRUSERCNAME
         MVC   SYSNAME,QUASTRUSERSYS
         MVC   JOBNAME,QUASTRUSERJOB
*                                              ALLOW REBUILD
         MVC   ALLOWR,=C'Y'
         TM    QUASTRUSERFLG1,QUASTRUSERALLOWREBLD
         BO    FLAG09
         MVC   ALLOWR,=C'N'
FLAG09   EQU   *
*                                              ALLOW ALTER
         MVC   ALLOWA,=C'Y'
         TM    QUASTRUSERALTERFLG,QUASTRUSERALTERALLOWED
         BO    FLAG08
         MVC   ALLOWA,=C'N'
FLAG08   EQU   *
         TM    QUASTRUSERFLG1,QUASTRUSERACT
         BO    FLAG10A
         TM    QUASTRUSERFLG1,QUASTRUSERFAIL
         BO    FLAG10B
         MVC   CSTATUS,=CL16'UNKNOWN'
         B     FLAG10
FLAG10A  EQU   *
```

```
         MVC   CSTATUS,=CL16'ACTIVE'
         B     FLAG1Ø
FLAG1ØB  EQU   *
         MVC   CSTATUS,=CL16'FAILED PERSISTENT'
         B     FLAG1Ø
FLAG1Ø   EQU   *
         CALL  ISPLINK,(TBADD,FSTABLES,,ORDER),VL  ADD DATA INTO TABLE
         CLI   QUASTRUSERTYP,X'A4'       LAST CONNECTIONS ?
         BE    DISPP3
         SR    R4,R4
         LH    R4,QUASTRUSERLEN
         AR    R7,R4
         B     LCONN
SSTRN    EQU   *
         AH    R4,QUASTRLEN
         B     SSTR
ERROR1   EQU   *
         CALL  ISPLINK,(SETMSG,MSG1),VL
         B     RETURN
RETURN   EQU   *
         L     R13,4(R13)
         L     R1,8(R13)
         FREEMAIN R,LV=WORKL,A=(R1)
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
IXCQUERY EQU   *
         AUTHON                                  AUTH SVC
         MODESET KEY=ZERO
         IXCQUERY REQINFO=STR_ALLDATA,                                 X
               ANSAREA=ANSAREA,                                        X
               ANSLEN=ANSLEN,                                          X
               RETCODE=RETCODE,                                        X
               RSNCODE=RSNCODE
         LTR   R15,R15
         BNZ   ERROR1
         MODESET KEY=NZERO
         AUTHOFF                                 RESET AUTH
         BR    R6
ANSLEN   DC    F'4Ø96Ø'
MASK     DC    X'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø'
*                                              MESSAGE
*                                              ———-
MSG1     DC    CL8'IXCØØ1E'
FSPANEL  DC    CL8'IXCST'            <===      ISPF PANEL NAME
FSPANELS DC    CL8'IXCSTST'          <===      ISPF PANEL NAME
*                                              FIELDS
FSKEY    DC    CL8'SKEY'
SKEY     DS    CL8
FSELECT  DC    CL8'SELECT'
SELECT   DS    CL1
FSTRNAME DC    CL8'STRNAME'
STRNAME  DS    CL16
```

```
FALLOC   DC    CL8'ALLOC'
ALLOC    DS    CL13
FCFNAME  DC    CL8'CFNAME'
CFNAME   DS    CL8
FPENDING DC    CL8'PENDING'
PENDING  DS    CL21
FPLCF    DC    CL8'PLCF'
PLCF     DS    CL71
FXLCF    DC    CL8'XLCF'
XLCF     DS    CL67
FINITSIZ DC    CL8'INITSIZE'
INITSIZE DS    CL8
FSIZE    DC    CL8'SIZE'
SIZE     DS    CL8
FREBUILD DC    CL8'REBUILDP'
REBUILDP DS    CL8
FCFNODE  DC    CL8'CFNODE'
CFNODE   DS    CL54
FUM      DC    CL8'UM'
UM       DS    CL1
FSYSNAME DC    CL8'SYSNAME'
SYSNAME  DS    CL8
FCONNAME DC    CL8'CONNAME'
CONNAME  DS    CL16
FJOBNAME DC    CL8'JOBNAME'
JOBNAME  DS    CL8
FCSTATUS DC    CL8'CSTATUS'
CSTATUS  DS    CL16
FALLOWA  DC    CL8'ALLOWA'
ALLOWA   DS    CL1
FALLOWR  DC    CL8'ALLOWR'
ALLOWR   DS    CL1
NAMELIST DC    CL150'(STRNAME ALLOC PENDING CFNAME CFNODE PLCF XLCF    X
               INITSIZE SIZE REBUILDP UM)'
NAMELISS DC    CL150'(CONNAME SYSNAME JOBNAME CSTATUS ALLOWA ALLOWR)'
FSTABLE  DC    CL8'FSTABLE'                    TABLE NAME
FSTABLES DC    CL8'FSTABLES'                   TABLE NAME
*                                              SORT PARMS
*                                              _____
CHARASND DC    CL3'C,A'
NUMRDSND DC    CL3'N,D'
SORTPARM DS    ØCL12
         DC    CL1'('
SORTKEY  DS    CL8
         DC    CL1','
SORTTYPE DS    CL3
         DC    CL1')'
SORTPARS DS    ØCL12
         DC    CL1'('
SORTKES  DS    CL8
         DC    CL1','
SORTTYPS DS    CL3
         DC    CL1')'
```

```
*                                                    ISPF FUNCTIONS
*                                                    ──────
VDEFINE  DC    CL7'VDEFINE'
TBADD    DC    CL5'TBADD'
TBCLOSE  DC    CL7'TBCLOSE'
TBCREATE DC    CL8'TBCREATE'
TBDISPL  DC    CL7'TBDISPL'
TBSORT   DC    CL6'TBSORT'
TBTOP    DC    CL5'TBTOP'
TRKS     DC    CL8'TRKS'
ORDER    DC    CL5'ORDER'
NOWRITE  DC    CL7'NOWRITE'
REPLACE  DC    CL7'REPLACE'
SETMSG   DC    CL6'SETMSG'
*                                                    TYPES
*                                                    ──
CHAR     DC    CL4'CHAR'
*
*                                                    LENGTH
*                                                    ──────
L1       DC    F'1'
L4       DC    F'4'
L8       DC    F'8'
L13      DC    F'13'
L16      DC    F'16'
L21      DC    F'21'
L54      DC    F'54'
L67      DC    F'67'
L71      DC    F'71'
*                                                    FIELDS
* WTO TO DEBUG
WTOC     WTO   '                                                X
                                                        ',MF=L,ROUTCDE=(11)
WTOL     EQU   *-WTOC                                 LENGTH OF MACRO EXPANSION
WTO      DS    CL(WTOL)
DSECT    DSECT
SAVEAREA DS    18F                                    SAVEAREA
RETCODE  DS    F
RSNCODE  DS    F
DOUBLE   DS    D
ANSAREA  DS    10CL4096
LGDSECT  EQU   *-DSECT
WORKL    EQU   LGDSECT                                LENGTH OF WORAREA
         IXCYQUAA
         IXLYNDE
         REGISTER
         END
```

ISPF PANELS

You need to install several ISPF panels in a library included in your

ISPPLIB concatenation.

Panel IXC

```
%------------  Coupling Facility Monitor ----------------
%OPTION  ===>_ZCMD
+
%                                                    +USERID   - &ZUSER
%                                                    +TIME     - &ZTIME
%
%   C +Coupling Facility  - Coupling Facilities Display
%   S +Structure          - CF Structures Display
%
)INIT
)PROC
&ZQ = &Z
  IF (&ZCMD ¬= ' ')
    &ZQ = TRUNC(&ZCMD,'.')
    IF (&ZQ = ' ')
      .MSG = ISRUØØØ
  &ZSEL = TRANS( &ZQ
                 C,'PGM(IXCCFIS)'
                 S,'PGM(IXCSTIS)'
               ' ',' '
                 X,'EXIT'
                 *,'?' )
  &ZTRAIL = .TRAIL
)END
```

Panel IXCCF

```
)ATTR
   _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) COLOR(RED)
   - TYPE(OUTPUT) INTENS(LOW) JUST(RIGHT)
   ı TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ! TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   } TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) COLOR(RED)
   ? TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   { TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
)BODY EXPAND(......)
+...-...?Coupling Facilities info+...-...+
${COMMAND%===>_ZCMD                                %SCROOL ===>_SAMT+
$
)MODEL CLEAR(SELECT)
+-----------------------------------------
_z% CFNAME:!z      %
     %Node:ız                                                  %
     %CFLEVEL:ız      %   %Storage Increment:-z      %k   Volatile:-z%
```

```
      Storage Usage:
       %Total:-z      % k
       %Dump :-z      % k
       %Free :-z      % k
)INIT
 .ZVARS = '(SELECT CFNAME CFNODE CFLEVEL STINC +
            vol +
            TSPACE DSPACE FSPACE)'
 &ZTDMARK = '*************** BOTTOM OF DATA +******************'
)PROC
  IF (.RESP = ENTER)
   VER (&SKEY,LIST,CFNAME)
)END
```

Panel IXCCFST

```
)ATTR
   _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) COLOR(RED)
   - TYPE(OUTPUT) INTENS(LOW) JUST(RIGHT)
   ı TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ! TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   } TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) COLOR(RED)
   ? TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   { TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
)BODY EXPAND(......)
+...-...?Coupling Facility Detail+...-...+
${COMMAND%===>_ZCMD                                          %SCROOL
===>_SAMT+
$
 %CFNAME:!CFNAME %
     %Dump:-dspace %k Structures:-sttot  %k Free:-fspace %k Total:-tspace %k

    STRNAME          Storage Type    Lst/Dir Lst     Data    Lock    Lock/Elm
                       (k)           Entries Headers Element Entries Size (b)
                                     Tot/Use         Tot/Use
)MODEL CLEAR(SELECT)
 _ZıZ               -Z      ıZ      -Z      -Z      -Z       -Z      -Z
%
                                    -Z      %       -Z       -Z      %
)INIT
 .ZVARS = '(SELECT STRNAME STSIZE STTYPE STDENT STLHD STDELM STLENT STLSIZE +
            STDENTU STDELMU STLENTU)'
 &ZTDMARK = '****************************** BOTTOM OF DATA +
*******************************'
)PROC
  IF (.RESP = ENTER)
   VER (&SKEY,LIST,CFNAME)
)END
```

Panel IXCCFSTD

```
)ATTR
   _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) COLOR(RED)
   - TYPE(OUTPUT) INTENS(LOW) JUST(RIGHT)
   ı TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ! TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   } TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) COLOR(RED)
   ? TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   { TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
)BODY EXPAND(......)
+...-...?Structure Detail+...-...+
${COMMAND%===>_ZCMD                                        %SCROOL
===>_SAMT+
$
 %CFNAME:ıCFNAME %

    STRNAME:      !Z              %     Storage:           -z      %k

    Type:                    -z      %   Response Time:
                                              SYNC:         -Z      %Ês
    List/Directory Entries:                   ASYNC:        -Z      %Ês
         Max:                -z      %
         Used:               -z      %   Access Rate:
    List Headers:            -z      %        SYNC:         -Z      %/sec
    Data Elements:                            ASYNC:        -Z      %/sec
         Max:                -z      %
         Used:               -z      %
    Lock Entries:
         Max:                -z      %
         Used:               -z      %
    Lock/Element Size:       -z      %

)INIT
 .ZVARS = '(STRNAME +
            STSIZE +
            STTYPE +
            STSTIME +
            STATIME +
            STDENT STDENTU +
            STLHD +
            STscnt +
            stacnt +
            STDELM +
            STDELMU +
            STLENT STLENTU +
            STLSIZE)'

 &ZTDMARK = '****************************** BOTTOM OF DATA +
```

```
*********************************'
)PROC
  IF (.RESP = ENTER)
   VER (&SKEY,LIST,CFNAME)
)END
```

Panel IXCST

```
)ATTR
   _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) COLOR(RED)
   - TYPE(OUTPUT) INTENS(LOW) JUST(RIGHT)
   ı TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ! TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   } TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) COLOR(RED)
   ? TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   { TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
)BODY EXPAND(......)
+...-...?Structures Info+...-...+
${COMMAND%===>_ZCMD                                          %SCROOL
===>_SAMT+
$
)MODEL clear(select)
+-----------------------------------------
_Z% STRNAME:!Z               % Status:ıZ            % ıZ                    +
        %CF:ıZ      %}Z% ıZ
        %Prefrence List:ız                                                  +
        %Exclusion List:ız                                                  +
        %Initsize:-Z      %K - Size:-Z      %K - Rebuild Pct:-Z      %
)INIT
 .ZVARS = '(SELECT STRNAME ALLOC +
            PENDING CFNAME UM CFNODE PLCF XLCF INITSIZE +
            SIZE REBUILDP)'
 &ZTDMARK = '****************************** BOTTOM OF DATA +
*********************************'
)PROC
  IF (.RESP = ENTER)
)END
```

Panel IXCSTST

```
)ATTR
   _ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) COLOR(RED)
   - TYPE(OUTPUT) INTENS(LOW) JUST(RIGHT)
   ı TYPE(OUTPUT) INTENS(LOW) JUST(LEFT)
   ! TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT)
   } TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) COLOR(RED)
   ? TYPE(TEXT) COLOR(RED) INTENS(HIGH)
   { TYPE(TEXT) COLOR(YELLOW) INTENS(HIGH)
   % TYPE(TEXT) COLOR(GREEN) INTENS(HIGH)
   $ TYPE(TEXT) SKIP(ON) INTENS(LOW)
```

```
)BODY EXPAND(......)
+...-...?Structures Info+...-...+
${COMMAND%===>_ZCMD                                          %SCROOL
===>_SAMT+
$
    STRNAME:!Z               % STATUS:ıZ            % ıZ                      +
        %CF:ıZ      %}Z% ıZ
        %Preference List:ız
+
        %Exclusion List:ız                                                    +
        %Initsize:-z      %k - Size:-z      %k - Rebuild pct:-z      %

                                                          Allow
 Connection Name  Sysname  Jobname  Status           Alter    Rebuild
+-----------------------------------------
)MODEL
ıZ               ıZ       ıZ       ıZ               ıZ%      ız%
)INIT
 .ZVARS = '(STRNAME ALLOC +
            PENDING CFNAME UM CFNODE PLCF XLCF INITSIZE +
            SIZE REBUILDP +
            CONNAme +
            SYSNAME +
            jobname +
            cstatus +
            allowa  +
            allowr)'
 &ZTDMARK = '******************************** BOTTOM OF DATA +
********************************'
)PROC
  IF (.RESP = ENTER)
)END
```

ISPF MESSAGES

You also need to install the following ISPF messages member in a library included in your ISPMLIB concatenation.

Message IXC

```
IXC001E 'ERROR...' .ALARM=YES
'ERROR DURING IXCQUERY MACRO...'
IXC002E 'ERROR...' .ALARM=YES
'ERROR DURING IXLMG MACRO...'
```


Mapping cross memory connections to an address space

INTRODUCTION

In APAR II08563, IBM describes the control block chains to follow in order to map the cross memory connections established between address spaces.

The APAR describes how to extract this information from stand-alone dumps or SVC dumps with ASID 2 (PCAUTH) local storage included, but I thought a program that could be executed on a running system would be a handy tool to have.

Since the process is explained in detail in the APAR, I will only mention that the key is the XMSE and SETC control blocks, which reside in the PCAUTH address space, so access to PCAUTH's extended private area is required. I used an access register to achieve this.

The XMSE includes the job name and ASID of its creator in OS/390 Release 1 Version 3 and above, and also points to the SETC for that address space. The SETC points to the XMSEs of all the address spaces that have connections to the address which 'owns' this SETC, so mapping the connections becomes very straightforward.

Here is a sample of the output from my program MAPXM. In this instance it is analysing the connections for a CICS address space that has connections to Landmark's product 'The Monitor for CICS/ESA, IMS/DBCTL and MQSeries':

```
CICS0001 XMS CONNECTIVITY               19991031    155059

ASSB 01AA8A80  XMSE 7F773C50    TO 00000008  FROM 00000000

TYPE --XMSE--  --JOBN--  --ASID--

 TO  FF786000  TCE1LFS   00000036
 TO  FF779830  IMS1DLI   0000006C
 TO  FF779A08  IMS1CTRL  0000006B
 TO  FF786420  IMS1DBM1  00000066
 TO  FF779BE0  IMS1DIST  00000067
 TO  FF77FC50  IMS1MSTR  00000064
```

```
    TO  FFFD9710  MQS1MSTR  00000037
    TO  FF783E28  TCE1CSM   00000043
```

Two special cases of connectivity are catered for. The first is an address space that is connected to a system LX, in which case it effectively has cross memory connections to all other address spaces. PCAUTH itself is such an address space:

```
   PCAUTH   XMS CONNECTIVITY              19991031    154518

   ASSB 02B81E00  XMSE 7FFD8E28    TO 00000000  FROM 00000000

   TYPE --XMSE--  --JOBN--  --ASID--

   SYSTEM LX BIT IS ON, XMS TO ALL ASIDS IN THE SYSTEM
```

The second case is for an address space that has no connections: in this case the address space's own XMSE is zero. JES is an example of this:

```
   JES2     XMS CONNECTIVITY              19991031    154518

   XMSE IS ZERO
```

The program is run as a simple batch job, with a single input parameter which is the job name to be analysed. This must be specified as an 8-byte field, so trailing blanks must be included (ie 'PCAUTH'). The program should be assembled with high-level Assembler, or the long names used will have to be modified for Assmbler H. It must be linked with AC=1 into an APF authorized library.

MAPXMS

```
*--------------------------------------------------------------------
*        MAP CROSS MEMORY CONNECTIONS
*        SEE IBM APAR II08563 FOR DETAILS
*--------------------------------------------------------------------
*
         LCLC  &MODULE
&MODULE  SETC  'MAPXMS'
&MODULE  CSECT
&MODULE  AMODE 31
&MODULE  RMODE 24
         YREGS
         SAVE  (14,12)
         USING MAPXMS,R12
         LR    R12,R15
```

```
         LR    R14,R13
         LA    R13,SAVE
         ST    R13,8(,R14)
         ST    R14,4(,R13)
READ_PARAMETERS DS ØH
         LR    R11,R1                    PARAMETER POINTER
         L     R1Ø,Ø(R11)
         MVC   JOBNAME(8),2(R1Ø)
FIND_ASID_OF_JOBNAME DS ØH
         L     R11,CVTPTR
         L     R11,CVTASVT-CVTMAP(R11)
         USING ASVT,R11
         LA    R1Ø,ASVTENTY
         L     R9,ASVTMAXU
ASVT_LOOP_ROUTINE DS ØH
         TM    Ø(R1Ø),ASVTAVAL           IS THE SLOT OCCUPIED?
         BO    TRY_NEXT_ASCB             NO, THEN BYPASS
         L     R8,Ø(R1Ø)                 GET POINTER TO ASCB
         USING ASCB,R8                   ASCB ADDRESSABILITY
         L     R1,ASCBJBNI               GET JOBNAME POINTER
         LTR   R1,R1                     JOBNAME?
         BZ    TRY_STC_FOR_JOBNAME       NO, STC MAYBE
         CLC   JOBNAME(8),Ø(R1)
         BE    FOUND_REGN
TRY_STC_FOR_JOBNAME DS ØH
         L     R1,ASCBJBNS               START/MOUNT/LOGON NAME?
         LTR   R1,R1                     IS IT?
         BZ    TRY_NEXT_ASCB             NO, JUST CONTINUE
         CLC   JOBNAME(8),Ø(R1)
         BE    FOUND_REGN
TRY_NEXT_ASCB  DS ØH
         LA    R1Ø,4(R1Ø)                POINT TO NEXT ASCB
         BCT   R9,ASVT_LOOP_ROUTINE      CONTINUE...
REGN_NOT_RUNNING DS ØH
         OPEN  (SYSPRINT,OUTPUT)
         MVC   OUTREC+2(7),=CL7'JOBNAME'
         MVC   OUTREC+9(8),JOBNAME
         MVC   OUTREC+18(1Ø),=CL1Ø' NOT FOUND'
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         CLOSE SYSPRINT
         B     END_OF_PROGRAM
FOUND_REGN DS  ØH
         MVC   JOBNAME_ASID,ASCBASID
         ST    R8,ADDRESS_OF_ASCB
         DROP  R8
GETMAIN_AREA_TO_STORE_INFO_R7_FOR_TABLE_POINTER DS ØH
         GETMAIN RU,LV=65536
         LR    R7,R1
         ST    R7,TABSTART
INTO_ACCESS_MODE_R6_FOR_JOBN_ADDRESS_SPACE DS ØH
```

```
         LH    R4,JOBNAME_ASID
         BAL   R9,INTO_ACCESS_MODE
FIND_ASSB DS   ØH
         L     R6,ADDRESS_OF_ASCB
         LA    R6,336(,R6)
         L     R6,Ø(,R6)
         ST    R6,ADDRESS_OF_ASSB
FIND_XMSE DS   ØH
         LA    R6,72(,R6)
         L     R6,Ø(,R6)
         ST    R6,ADDRESS_OF_XMSE
         BAL   R9,OUT_OF_ACCESS_MODE
CHECK_XMSE_ZERO DS   ØH
         LTR   R6,R6
         BNZ   INTO_ACCESS_MODE_R6_FOR_PCAUTH_ADDRESS_SPACE
         MVC   XMSE_ZERO,=CL1'Y'
         B     WRITE_FINDINGS
INTO_ACCESS_MODE_R6_FOR_PCAUTH_ADDRESS_SPACE DS ØH
         LH    R4,PCAUTH_ASID
         BAL   R9,INTO_ACCESS_MODE
FIND_SETC DS   ØH
         L     R6,ADDRESS_OF_XMSE
         LA    R6,4(,R6)
         L     R6,Ø(,R6)
         ST    R6,ADDRESS_OF_SETC
PROC_SETC DS   ØH
         LA    R6,6(,R6)
         TM    Ø(R6),X'8Ø'
         BNO   SYSTEM_LX_BIT_OFF
         MVC   SYSTEM_LX_BIT_ON,=CL1'Y'
         BAL   R9,OUT_OF_ACCESS_MODE
         B     WRITE_FINDINGS
SYSTEM_LX_BIT_OFF DS ØH
         L     R6,ADDRESS_OF_SETC
         LA    R6,2Ø(,R6)
         MVC   XMS_TO_COUNT,Ø(R6)
         LA    R6,2(,R6)
         MVC   XMS_FROM_COUNT,Ø(R6)
         L     R6,ADDRESS_OF_SETC
         LA    R6,28(,R6)
         ST    R6,ADDRESS_IN_ARRAY
         LH    R3,XMS_TO_COUNT
         LH    R6,XMS_FROM_COUNT
         AR    R3,R6
         LTR   R3,R3
         BZ    END_OF_XMSE
PROCESS_XMSE_COUNT DS ØH
         BAL   R8,PROCESS_XMSE_ARRAY
         BCT   R3,PROCESS_XMSE_COUNT
END_OF_XMSE DS ØH
         ST    R7,TABEND
         BAL   R9,OUT_OF_ACCESS_MODE
```

```
WRITE_FINDINGS DS ØH
         L     R7,TABSTART
         L     R6,TABEND
         OPEN  (SYSPRINT,OUTPUT)
WRITE_HEADER DS ØH
         MVC   OUTREC+2(8),JOBNAME
         MVC   OUTREC+11(16),=CL16'XMS CONNECTIVITY'
         TIME  DEC,TIMEDATE,LINKAGE=SYSTEM,DATETYPE=YYYYMMDD
         L     R5,TIMEDATE+8
         REGDISP R5,OUTREC+42(8)
         L     R5,TIMEDATE
         REGDISP R5,OUTREC+54(6)
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
WRITE_IF_ZERO DS ØH
         CLC   XMSE_ZERO,=CL1'Y'
         BNE   WRITE_ASSB
         MVC   OUTREC+2(12),=CL12'XMSE IS ZERO'
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         B     END_OF_DETAIL
WRITE_ASSB DS ØH
         MVC   OUTREC+2(4),=CL4'ASSB'
         L     R5,ADDRESS_OF_ASSB
         REGDISP R5,OUTREC+7(8)
         MVC   OUTREC+17(4),=CL4'XMSE'
         L     R5,ADDRESS_OF_XMSE
         REGDISP R5,OUTREC+22(8)
         MVC   OUTREC+32(4),=CL4'  TO'
         LH    R5,XMS_TO_COUNT
         REGDISP R5,OUTREC+37(8)
         MVC   OUTREC+47(4),=CL4'FROM'
         LH    R5,XMS_FROM_COUNT
         REGDISP R5,OUTREC+52(8)
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         MVC   OUTREC+2(33),=CL33'TYPE --XMSE--  --JOBN--  --ASID--'
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
WRITE_SYSTEM_LX_BIT_ON DS ØH
         CLC   SYSTEM_LX_BIT_ON,=CL1'Y'
         BNE   WRITE_DETAIL
         MVC   OUTREC+2(31),=CL31'SYSTEM LX BIT IS ON, XMS TO ALL'
         MVC   OUTREC+33(2Ø),=CL2Ø' ASIDS IN THE SYSTEM'
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         B     END_OF_DETAIL
WRITE_DETAIL DS ØH
         LH    R3,XMS_TO_COUNT
         LH    R6,XMS_FROM_COUNT
         AR    R3,R6
         LTR   R3,R3
```

```
         BZ    END_OF_DETAIL
WRITE_XMSE_DETAIL DS ØH
         BAL   R8,WRITE_XMSE
         BCT   R3,WRITE_XMSE_DETAIL
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
END_OF_DETAIL DS ØH
         CLOSE SYSPRINT
         L     R7,TABSTART
         FREEMAIN RU,LV=65536,A=(7)
END_OF_PROGRAM DS ØH
         L     R13,SAVE+4
         RETURN (14,12),RC=Ø
*
WRITE_XMSE DS ØH
         TM    Ø(R7),X'8Ø'
         BNO   WRITE_XMSE_Ø1
         MVC   OUTREC+2(4),=CL4' TO '
         B     WRITE_XMSE_Ø2
WRITE_XMSE_Ø1  DS ØH
         MVC   OUTREC+2(4),=CL4'FROM'
WRITE_XMSE_Ø2  DS ØH
         L     R5,Ø(R7)
         REGDISP R5,OUTREC+7(8)
         MVC   OUTREC+17(8),4(R7)
         LH    R5,12(R7)
         REGDISP R5,OUTREC+27(8)
         CLC   OUTREC+17(8),=CL8'NOT USED'
         BNE   WRITE_XMSE_Ø3
         MVC   OUTREC+2(4),=CL4' '
         MVC   OUTREC+27(8),=CL8' '
WRITE_XMSE_Ø3  DS ØH
         BAL   R9,WRITE_RECORD_TO_SYSPRINT
         LA    R7,14(,R7)
         BR    R8                  RETURN TO CALLER
*
PROCESS_XMSE_ARRAY DS ØH
         L     R6,ADDRESS_IN_ARRAY
         LA    R6,4(,R6)
         MVC   Ø(4,R7),Ø(R6)
         LA    R7,4(,R7)
         ST    R6,ADDRESS_IN_ARRAY
         TM    3(R6),X'Ø1'
         BO    ENTRY_NOT_USED
         L     R6,Ø(,R6)
         BAL   R9,EXTRACT_JOBNASID_FROM_XMSE
         LA    R7,1Ø(,R7)
         BR    R8                  RETURN TO CALLER
ENTRY_NOT_USED DS ØH
         MVC   Ø(8,R7),=CL8'NOT USED'
         LA    R7,1Ø(,R7)
         BR    R8                  RETURN TO CALLER
*
```

```
EXTRACT_JOBNASID_FROM_XMSE DS ØH
         LA    R6,28(,R6)
         MVC   Ø(8,R7),Ø(R6)        XMSE JOBNAME
         LA    R6,8(,R6)
         MVC   8(2,R7),Ø(R6)        XMSE ASID
         BR    R9                   RETURN TO CALLER
*
WRITE_RECORD_TO_SYSPRINT DS ØH
         PUT   SYSPRINT,OUTCARD
         MVI   OUTREC,C' '
         MVC   OUTREC+1(132),OUTREC
         BR    R9                   RETURN TO CALLER
*
INTO_ACCESS_MODE DS ØH
         MODESET MODE=SUP,KEY=ZERO
         AXSET AX=AX1
         SSAR  R4
         SAC   512
         LAM   R6,R6,=F'1'
         BR    R9                   RETURN TO CALLER
*
OUT_OF_ACCESS_MODE DS ØH
         EPAR  R2
         SSAR  R2
         SAC   Ø
         AXSET AX=AXØ
         MODESET MODE=PROB,KEY=NZERO
         BR    R9                   RETURN TO CALLER
*-------------------------------------------------------------------
*        WORKING STORAGE
*-------------------------------------------------------------------
                DS ØD
SAVE            DS 18F
AXØ             DC H'Ø'
AX1             DC H'1'
JOBNAME         DC CL8'JOBNAME '
TABSTART        DS F
TABEND          DS F
TIMEDATE        DS ØCL16         TIME AND DATE RETURNED
                DC XL16'ØØ'
ADDRESS_OF_ASCB  DS F
ADDRESS_OF_ASSB  DS F
ADDRESS_OF_XMSE  DS F
ADDRESS_OF_SETC  DS F
ADDRESS_IN_ARRAY DS F
SYSTEM_LX_BIT_ON DS C
XMSE_ZERO        DS C
                 DS ØH
XMS_TO_COUNT     DS H
XMS_FROM_COUNT   DS H
JOBNAME_ASID     DS H
PCAUTH_ASID      DC XL2'ØØØ2'
```

```
OUTCARD          DC AL2(137),AL2(Ø)
OUTREC           DC CL133' '
         ORG     OUTREC+133
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,                      X
               LRECL=137,BLKSIZE=137Ø,RECFM=VB
         IHAASVT
         IHAASCB
         CVT   DSECT=YES
         END
```

REGDISP MACRO

```
*********************************************************************
**  Convert the contents of a passed register to an 8-character      **
**  display field.                                                   **
*********************************************************************
         MACRO
&LABEL   REGDISP &HEX,&DSP
&LABEL   STM   Ø,15,SAVE&SYSNDX
         ST    &HEX,WHEX&SYSNDX
         UNPK  WDSP&SYSNDX.(9),WHEX&SYSNDX.(5)
         NC    WDSP&SYSNDX.(8),MASK&SYSNDX
         TR    WDSP&SYSNDX.(8),HXTB&SYSNDX
         MVC   &DSP,WDSP&SYSNDX
         LM    Ø,15,SAVE&SYSNDX
         B     END&SYSNDX
SAVE&SYSNDX DS 16F
MASK&SYSNDX DC XL8'ØFØFØFØFØFØFØFØF'
HXTB&SYSNDX DC CL16'Ø123456789ABCDEF'
WHEX&SYSNDX DS F
            DS C
WDSP&SYSNDX DS CL8'********'
            DC CL1'.'
END&SYSNDX  DS ØH
         MEND
```

SAMPLE JCL TO EXECUTE MAPXMS

```
//MAPXMS   JOB ....
//MAPXMS   EXEC PGM=MAPXMS,PARM='PCAUTH  '
//STEPLIB  DD DSN=YOUR.APF.LOADLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Patrick Mullen
System Software Consultant (Canada)

Utility to identify and eject tapes with errors

THE PROBLEM

The following utility is designed to run against the TCDB VOLCAT on OS/390 with ATL attached device. The standard IBM-supplied utility runs well, but, at our site, which has about 7000 cartridges, it takes over two hours to check the tapes. We have to undertake this validity scan upon the catalog on a daily basis. Although it does not CPU or storage, the long elapsed time can sometimes impact on our maintenance window and we have to wait until final completion before tapes with errors are ejected from the Automated Tape Libraries.

A SOLUTION

The following REXX runs against the whole catalog; however, it has an elapsed time of only 1-2 minutes! This includes the time it takes to eject the tape with errors from the library and create a small report for the operators with the error on the specific tapes and the location of every tapes with errors. You can edit and add the error codes, and also can add explanation to your operators of what should be done for any type of error. The JCL is shown below:

```
//SS38EJC JOB (SS38,B1,3),TAPES-IN-ERROR,NOTIFY=SØ38,
//   MSGCLASS=T,REGION=5ØØØK
//*
//* PRINT TCDB TO FILE
//*
//S1   EXEC PGM=IDCAMS
//SYSPRINT DD DSN=SØ38.TEXT3,DISP=SHR
//SYSIN    DD *
  LISTC CAT(TCDB.VOLCAT.VGENERAL) ALL
/*
//*
//* RUN ISPFBATCHREXX:
//* 1. EJECT COMMANDS WILL BE AT:
//*    'SØ38.LIB.CNTL(EJERROR)' AND WILL BE
//*    PROCCESSED AT STEP #3
//* 2. DETAILED REPORT OF TAPES IN ERROR WILL BE AT:
//*    BE AT: 'SØ38.TEXT31'
//*
//S2      EXEC ISPBATCH
//SYSPROC  DD DSN=SØ38.LIB.CNTL,DISP=SHR
```

```
//         DD DSN=SYS2.CLIST,DISP=SHR
//SYSTSIN  DD *
  PROFILE NOPREFIX
  ISPSTART CMD(TAPERR) BDISPMAX(99999999)
/*
//*
//* EJECT ERROR TAPES FROM LIBRARY WITH
//* THE STANDARD IBM EJECT UTILITY
//*
//S3   EXEC PGM=LCSUTIL1,PARM='EJECT'
//DI    DD  DDNAME=SYSIN
//PRINT DD  SYSOUT=*
//SYSIN DD DSN=SØ38.LIB.CNTL(EJERROR),DISP=SHR
//*
```

TAPERR

```
/* REXX - CHECK FOR ERROR-TAPES IN ATL         */
TRACE N
/* *———————————————————— */
/*                                             */
/* Step #1: ALLOC VOLCAT printings & edit them */
/*                                             */
/* Step #2: WRITE ERROR TAPES                  */
/*                                             */
/* Step #3: WRITE EJECT Commands               */
/*                                             */
/* *———————————————————— */
ADDRESS ISPEXEC "EDIT DATASET('SØ38.TEXT3') MACRO(TAPERRE)"
ADDRESS TSO 'FREE FILE(IN) DA(SØ38.TEXT3)'
ADDRESS TSO 'ALLOC FILE(IN) DA(SØ38.TEXT3) SHR'
ADDRESS TSO 'ALLOC FILE(OUT) DA(SØ38.TEXT31) SHR'
ADDRESS TSO 'ALLOC FILE(OUT1) DA(SØ38.LIB.CNTL(EJERROR)) SHR'
I=Ø
LOOP1:
"EXECIO 1 DISKR IN "
IF RC > Ø THEN SIGNAL OUT
  PULL ENTRY
   TEMP = ENTRY
   PARSE VAR TEMP A1 .
         IF SUBSTR(A1,1,4) = 'ØVOL' THEN DO
                   VOLE=SUBSTR(ENTRY,19,6)
                   SIGNAL LOOP1
         END
       IF SUBSTR(A1,1,4) = 'LIBR' THEN DO
                 PARSE VAR ENTRY LIB REC ERR CRE .
                 LIB=SUBSTR(LIB,17,8)
                 REC=SUBSTR(REC,18,8)
                 CRE=SUBSTR(CRE,15,1Ø)
       END
    IFERR=SUBSTR(ERR,18,7)
    IF IFERR = 'NOERROR' THEN SIGNAL LOOP1
                 SAY VOLE LIB ERR CRE
```

```
WRITERR:
IF I > Ø THEN SIGNAL WRITETP
OUT. =
OUT.1 = ' VOLSER '  ' POSITION '  ' ERROR CODE ' ,
        '             CREATION DATE '
OUT.2 = '————'  '——————'  '———————————' ,
        '———————'
  "EXECIO * DISKW OUT (STEM OUT."
I=I+1
WRITETP:
OUT. =
OUT.1 =  VOLE || '    ' || LIB || '   ' || ERR || ' ' CRE
  "EXECIO * DISKW OUT (STEM OUT."
  IF RC > Ø THEN DO
            SAY "ERROR WRITING ON DATASET :"
            SIGNAL OUT
            END
IF LIB = '—SHELF' THEN SIGNAL LOOP1
WRITEJ:
OUT1. =
OUT1.1 = VOLE
  "EXECIO * DISKW OUT1 (STEM OUT1."
  IF RC > Ø THEN DO
            SAY "ERROR WRITING ON DATASET :"
            SIGNAL OUT
            END
SIGNAL LOOP1
OUT:
  "EXECIO Ø DISKR IN (FINIS"
  ADDRESS TSO "FREE F(IN)"
  "EXECIO Ø DISKW OUT (FINIS"
  ADDRESS TSO "FREE F(OUT)"
  "EXECIO Ø DISKW OUT1 (FINIS"
  ADDRESS TSO "FREE F(OUT1)"
  EXIT
```

TAPERRE EDIT MACRO

```
/* REXX */
/* DELETE all lines except VOLSER & Error-Status */
ADDRESS  ISREDIT "MACRO"
ADDRESS  ISREDIT "X 'VOLUME-ENTRY' ALL"
ADDRESS  ISREDIT "X 'ERROR-STATUS' ALL"
ADDRESS  ISREDIT "DEL NX ALL"
ADDRESS  ISREDIT "RESET"
ADDRESS  ISREDIT "SAVE"
ADDRESS  ISREDIT "END"
ADDRESS  ISREDIT "MEND"
RETURN
```

Oded Tager
Senior Systems Programmer (Israel)

Cursor-sensitive ISPF (part 2)

This month we continue our look at the cursor-sensitive 'DS' command.

```
  if (&zalspc = CYLINDER)
   &SPCUC0 = 'cylinders :'
   &SPCUC1 = 'cylinders  . . :'
   &SPCUC2 = 'cylinders:'
   &SPCUC3 = 'cylinders :'
  if (&zalspc = TRACK)
   &SPCUC0 = 'tracks  . :'
   &SPCUC1 = 'tracks . . . . :'
   &SPCUC2 = 'tracks . :. '
   &SPCUC3 = 'tracks  . :'
  if (&zalspc = BLOCK)
   &SPCUC0 = 'blocks  . :'
   &SPCUC1 = 'blocks . . . . :'
   &SPCUC2 = 'blocks . :'
   &SPCUC3 = 'blocks  . :'
  if (&zalspc = MEGABYTE)
   &SPCUC0 = 'megabytes :'
   &SPCUC1 = 'megabytes  . . :'
   &SPCUC2 = 'megabytes:'
   &SPCUC3 = 'megabytes :'
  if (&zalspc = KILOBYTE)
   &SPCUC0 = 'kilobytes :'
   &SPCUC1 = 'kilobytes  . . :'
   &SPCUC2 = 'kilobytes:'
   &SPCUC3 = 'kilobytes :'
  if (&zalspc = BYTE)
   &SPCUC0 = 'bytes . . :'
   &SPCUC1 = 'bytes  . . . . :'
   &SPCUC2 = 'bytes  . :'
   &SPCUC3 = 'bytes . . :'
)PROC
)END
```

DSIPO PANEL

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------------*/
/* Display dataset information for DS command (EXEC)   */
/*  derived from IBM panel ISRUAIPO                    */
/*-------------------------------------------------------------*/
)ATTR DEFAULT(%+_)
  ¬ TYPE(PT)
  ~ TYPE(FP)
  # TYPE(VOI) PADC(USER)
  * TYPE(CH)
```

```
  ı TYPE(NEF) CAPS(ON) PADC(USER)
  ! AREA(SCRL) EXTEND(ON)
)BODY  CMD(ZCMD)
+                            ¬Dataset Information+
+
~DS Action ===>ıZ
+
!SAREA39
!
!
!
)AREA SAREA39
~Data Set Name  . . . :ıZ                                              +

*General Data*                          *Current Allocation*
 ~Volume serial . . . :ıZ     +          ~Allocated &SPCUCØ . :#Z
+
 ~Device type . . . . :#Z       +        ~Allocated extents . :#Z
+
 ~Organization  . . . :#Z       +        ~Maximum dir. blocks :#Z
+
 ~Record format . . . :#Z     +
 ~Record length . . . :#Z      +
 ~Block size  . . . . :#Z       +      *Current Utilization*
 ~1st extent &SPCUC2  :#Z             + ~Used &SPCUC1  . . . :#Z
+
 ~Secondary &SPCUC3 . :#Z             + ~Used dir. blocks  . :#Z
+
                                        ~Number of members . :#Z
+
 ~Creation date . . . :#Z         +
~ Last Reference date :#Z         +
)INIT
.ZVARS = '(ZCMD DSN  ZALVOL TOTA DEVT EXTA DSORG ZALDIR ZALRF ZALLREC
ZALBLK +
           ZAL1EX TOTU ZAL2EX DIRU NRMEM CRDT REFDATE)'
.HELP = DSIHELP
&ZCMD = ' '
  if (&zalspc = CYLINDER)
   &SPCUCØ = 'cylinders :'
   &SPCUC1 = 'cylinders  . . :'
   &SPCUC2 = 'cylinders:'
   &SPCUC3 = 'cylinders :'
  if (&zalspc = TRACK)
   &SPCUCØ = 'tracks  . :'
   &SPCUC1 = 'tracks . . . . :'
   &SPCUC2 = 'tracks . :. '
   &SPCUC3 = 'tracks  . :'
  if (&zalspc = BLOCK)
   &SPCUCØ = 'blocks  . :'
   &SPCUC1 = 'blocks . . . . :'
```

```
   &SPCUC2 = 'blocks . :'
   &SPCUC3 = 'blocks  . :'
  if (&zalspc = MEGABYTE)
   &SPCUCØ = 'megabytes :'
   &SPCUC1 = 'megabytes  . . :'
   &SPCUC2 = 'megabytes:'
   &SPCUC3 = 'megabytes :'
  if (&zalspc = KILOBYTE)
   &SPCUCØ = 'kilobytes :'
   &SPCUC1 = 'kilobytes  . . :'
   &SPCUC2 = 'kilobytes:'
   &SPCUC3 = 'kilobytes :'
  if (&zalspc = BYTE)
   &SPCUCØ = 'bytes . . :'
   &SPCUC1 = 'bytes  . . . . :'
   &SPCUC2 = 'bytes  . :'
   &SPCUC3 = 'bytes . . :'
)PROC
)END
```

DSIHELP PANEL

```
)ATTR DEFAULT(%+_)
/*———————————————————————————————————————————————————————————*/
/* HELP for "Data Set Information" panels from DS command (EXEC) */
/*———————————————————————————————————————————————————————————*/
  ¬ TYPE(PT)
  $ TYPE(NT)
  ı TYPE(ET)
  # TYPE(CT)
)BODY
#HELP                        ¬Dataset Information#
HELP#
$
$This panel differs from the equivalent IBM panels in the following ways:
$
%o$The information shown on this panel comes from LISTDSI, so it does not
$  have exactly the same details (eg only the first volume serial is shown).
$
%o$PressıENTER$to get fresh information, or pressıPF3$orıPF12$to EXIT.
$
%o$TheıDataset Name$field can be overtyped, then ENTER pressed to show
$  information about another dataset.
$
%o$TheıVolume Serial$field can be overtyped, then ENTER pressed, to show
$  information about a dataset with the same name on a different volume.
$
%o$Instead of aıCommand$field there is aıDS Action$field for entering any
$  action to take, including the following:
```

```
$    #A, B, C, CO, D, DI, E, F, I, L, LC, M, MO, R, RS, S, U, V, VOL, Z.
$
%o$Action#H$or#?$will display the general HELP information for DS Command.
$
%o$You can use combinations, eg. from display of a PDS you could type action#B
$  and add#(LNK*)$at end of dsname to BROWSE all members starting with 'LNK'.
$
)INIT
)PROC
&HELP = YES                            /* used in DSHELP panel */
&ZCONT = DSHELP
)END
$    #A, B, C, CO, D, DI, E, F, I, L, LC, M, MO, Q, R, RS, S, ST, U, V,
VOL, Z.
```

LVSAM AND LCAT

Note that these two EXECs can be used without the DS command. For example, from an ISPF option 3.4 dataset list you can enter ‘LVSAM’ or ‘LCAT’ beside a dataset name.

LVSAM EXEC

```
/*============================== REXX ================================*/
/* LVSAM - VSAM LISTCAT OUTPUT FORMATTED IN AN ISPF PANEL             */
/*                                                                    */
/*  Invoked : 'LVSAM datasetname'                                     */
/*            a) in ISPF 3.4 dataset list - enter LVSAM beside the    */
/*               dataset name                                         */
/*            b) LVSAM is used by the DS EXEC                         */
/*                                                                    */
/*  External: LCAT   - EXEC invoked only if the dataset is not VSAM   */
/*            LVSAMP - panel displaying dataset information           */
/*            LVSAMH - panel for HELP                                 */
/*====================================================================*/
 IF SYSVAR(SYSISPF) = 'NOT ACTIVE' THEN DO
    SAY 'ISPF MUST BE ACTIVE TO USE THIS FUNCTION'
    EXIT
    END

 ZERRHM = 'LVSAMH'                  /* HELP panel for LVSAMP panel    */
 ZERRALRM = 'YES'                   /* sound alarm with any message   */

 ARG DSNAME
 IF DSNAME = ' ' THEN DO
    SAY "PLEASE ENTER VSAM DATASET NAME"
    PULL DSNAME
    END
 TDSN = STRIP(DSNAME,,"'")
```

```
 STATUS = SYSDSN("'"TDSN"'")
 IF STATUS <> 'OK' THEN DO                  /* show error message     */
    ZERRSM = 'Dataset not found'
    ZERRLM = 'Dataset' TDSN 'was not found by SYSDSN, Status =' STATUS
    "ISPEXEC SETMSG MSG(ISRZØØ2)"           /* standard IBM message   */
    EXIT
    END

GETINFO:
 X = OUTTRAP('OUT.',15ØØ)                   /* trap up to 15ØØ lines  */
 "LISTC ENT('"TDSN"') ALL"                  /* show the LISTC output  */
 X = OUTTRAP('OFF')

 IF POS('NONVSAM',OUT.1) <> Ø THEN DO
    ZERRSM = 'Dataset not VSAM'
    ZERRLM = 'Dataset' TDSN 'was catalogued but it is NOT VSAM'
    "ISPEXEC SETMSG MSG(ISRZØØ2)"           /* show error message     */
    "%LCAT '"TDSN"'"                        /* show the LISTC output  */
    EXIT
    END
 IF POS('ALIAS',OUT.1) <> Ø THEN DO
    ZERRSM = 'Dataset is ALIAS'
    ZERRLM = TDSN 'was catalogued but it is an ALIAS'
    "ISPEXEC SETMSG MSG(ISRZØØ2)"           /* show error message     */
    "%LCAT '"TDSN"'"                        /* show the LISTC output  */
    EXIT
    END
 IF POS('INDEX',OUT.1) <> Ø THEN INDEX = 'Y'
 VSCOMP = WORD(OUT.1,1)                     /* CLUSTER, DATA or INDEX */
 CATLG = SUBSTR(OUT.2,17,44)                /* catalog it is in       */
 CDATE = SUBSTR(OUT.4,53,8)                 /* creation date yyyy.ddd */
 EDATE = SUBSTR(OUT.5,53,8)                 /* expiry date   yyyy.ddd */
 DO I = 7 TO OUT.Ø
    IF POS('SMSDATA',OUT.I) = 6 THEN DO         /****** SMSDATA *******/
       I = I + 1
       SCLASS = STRIP(SUBSTR(OUT.I,24,8),,'-')  /* SMS Storage Class  */
       MCLASS = STRIP(SUBSTR(OUT.I,53,8),,'-')  /* SMS Managmnt Class */
       I = I + 1
       DCLASS = STRIP(SUBSTR(OUT.I,24,8),,'-')  /* SMS Data Class     */
       END
    IF POS('ATTRIBUTES',OUT.I) = 6 THEN DO      /***** ATTRIBUTES *****/
       I = I + 1
       IF POS('NOUPGRADE',OUT.I) = 8 ! POS('UPGRADE',OUT.I) = 8 THEN DO
          PARSE VAR OUT.I OPTF
          ITERATE
          END
       KLEN = STRIP(SUBSTR(OUT.I,25,8),,'-')    /* key length         */
       ALEN = STRIP(SUBSTR(OUT.I,54,7),,'-')    /* av'ge record length*/
       CISZ = STRIP(SUBSTR(OUT.I,114,5),,'-')   /* CI size            */
       I = I + 1
```

```
      RKP  = STRIP(SUBSTR(OUT.I,25,8),,'-')    /* relative key pos'n */
      MLEN = STRIP(SUBSTR(OUT.I,54,7),,'-')    /* max record length  */
      CICA = STRIP(SUBSTR(OUT.I,114,5),,'-')   /* number of CI/CA    */
      I = I + 1
      IF POS('RECORDS/CI',OUT.I) <> Ø THEN I = I + 1
      IF POS('AXRKP',OUT.I) <> Ø THEN I = I + 1
      TYPE = 'INDEXED   '
      IF POS('NONINDEX',OUT.I) <> Ø THEN TYPE = 'NONINDEX  '
      IF POS('NUMBERED',OUT.I) <> Ø THEN TYPE = 'NUMBERED  '
      IF POS(' IMBED',OUT.I) <> Ø & INDEX = 'Y' THEN SEQ = 'Y'
      PARSE VAR OUT.I OPT1 OPT2 OPT3 OPT4 OPT5 OPT6 OPT7 OPT8
      I = I + 1
      IF POS('NONINDEX',OUT.I) <> Ø THEN TYPE = 'NONINDEX  '
      IF POS('NUMBERED',OUT.I) <> Ø THEN TYPE = 'NUMBERED  '
      IF POS(' IMBED',OUT.I) <> Ø & INDEX = 'Y' THEN SEQ = 'Y'
      PARSE VAR OUT.I OPTA OPTB OPTC OPTD OPTE
      IF OPTF <> 'OPTF' THEN OPTE = OPTF
      LEAVE
      END
   END
DO I=I TO OUT.Ø
   IF POS('STATISTICS',OUT.I) = 6 THEN DO      /***** STATISTICS *****/
      I = I + 1
      RTOT = STRIP(SUBSTR(OUT.I,21,11),,'-')   /* total records      */
      CISP = STRIP(SUBSTR(OUT.I,54,7),,'-')    /* CI splits          */
      I = I + 1
      RDEL = STRIP(SUBSTR(OUT.I,21,11),,'-')   /* records deleted    */
      CASP = STRIP(SUBSTR(OUT.I,54,7),,'-')    /* CA splits          */
      EXTS = STRIP(SUBSTR(OUT.I,87,3),,'-')    /* extents            */
      I = I + 1
      RINS = STRIP(SUBSTR(OUT.I,21,11),,'-')   /* records inserted   */
      CIFR = STRIP(SUBSTR(OUT.I,58,3),,'-')    /* CI freespace       */
      I = I + 1
      RUPD = STRIP(SUBSTR(OUT.I,21,11),,'-')   /* records updated    */
      CAFR = STRIP(SUBSTR(OUT.I,58,3),,'-')    /* CA freespace       */
      I = I + 1
      RRET = STRIP(SUBSTR(OUT.I,21,11),,'-')   /* records retrieved  */
      LEAVE
      END
   END
DO I=I TO OUT.Ø
   IF POS('ALLOCATION',OUT.I) = 6 THEN DO      /***** ALLOCATION *****/
      I = I + 1
      ATYP = 'cylinders:'
      IF SUBSTR(OUT.I,29,3) = 'ACK' THEN
      ATYP = 'tracks:'
      IF SUBSTR(OUT.I,29,3) = 'ORD' THEN
      ATYP = 'records:'
      BTYP = ATYP
      ARBA = STRIP(SUBSTR(OUT.I,5Ø,11),,'-')   /* high alloc RBA     */
```

```
      I = I + 1
      SPRI = STRIP(SUBSTR(OUT.I,24,8),,'-')    /* primary space      */
      URBA = STRIP(SUBSTR(OUT.I,5Ø,11),,'-')   /* high used RBA      */
      I = I + 1
      SSEC = STRIP(SUBSTR(OUT.I,24,8),,'-')    /* secondary space    */
      LEAVE
      END
   END
A = 1
DO I = I TO OUT.Ø
   IF POS('INDEX',OUT.I) = 4 THEN LEAVE
   IF POS('VOLSER',OUT.I) = 8 THEN DO
      VSER.A = STRIP(SUBSTR(OUT.I,26,6),,'-')
      IF SEQ = 'Y' & A = 1 THEN DO
         ARBA = STRIP(SUBSTR(OUT.I,8Ø,1Ø),,'-')
         EXTS = STRIP(SUBSTR(OUT.I,116,3),,'-')
         END
      A = A +1
      IF A = 2 THEN DO
         I = I + 1
         IF SEQ = 'Y' THEN DO
            URBA = STRIP(SUBSTR(OUT.I,8Ø,1Ø),,'-')
            I = I + 1
            TRKS = STRIP(SUBSTR(OUT.I,59,2),,'-')
            LEAVE
            END
         I = I + 1
         TRKS = STRIP(SUBSTR(OUT.I,59,2),,'-')
         END
      END
   END
VOL1 = VSER.1
IF VSER.2 <> 'VSER.2' THEN VOL2 = VSER.2
IF VSER.3 <> 'VSER.3' THEN VOL3 = VSER.3
ATRK = ARBA/CISZ/CICA*TRKS         /* allocated tracks               */
UTRK = URBA/CISZ/CICA*TRKS         /* used tracks                    */
UPER = UTRK/ATRK*1ØØ               /* used percentage                */
ATRK = FORMAT(ATRK,,Ø)
UTRK = FORMAT(UTRK,,Ø)
UPER = FORMAT(UPER,,Ø)

"ISPEXEC DISPLAY PANEL(LVSAMP)"    /* display the information        */
IF RC > Ø THEN RETURN              /* EXIT if PF3 or PF12            */

ZERRSM = 'Updated'
ZERRLM = 'The VSAM information was updated from new LISTCAT output'
"ISPEXEC SETMSG MSG(ISRZØØ2)"      /* standard IBM message           */

SIGNAL GETINFO             /* go back to display updated information */
```

LCAT EXEC

```
/*=======================>> REXX     <<==============================*/
/*  LCAT:  Browse output from 'LISTCAT ENT(dsn) ALL' command          */
/*=========================================================================*/
   Parse Upper ARG dsn parm
   x = OutTrap('LISTC_Output.',1500)     /* up to 1500 lines */
   If parm = '' Then parm = 'ALL'
   Address TSO "LISTC ENT("DSN")" parm
   listc_rc = rc
   x = OutTrap('Off')
   outds = "'"Userid()".TSO.LISTC'" /* output dataset name */
   If Sysdsn(outds) = "OK"
      Then alloc_info = "SHR REUSE"
      Else alloc_info = "NEW UNIT(3390) SPACE(1 1) TRACKS",
                        "RECFM(F B) LRECL(121) DSORG(PS)",
                        "REUSE"
   "ALLOC F(TSOLISTC) DATASET("outds")" alloc_info
   "EXECIO * DISKW TSOLISTC (STEM LISTC_Output. OPEN FINIS "
   Address ISPEXEC "ISPEXEC BROWSE DATASET("outds")"
   Return listc_rc
```

LVSAMP PANEL

```
)ATTR DEFAULT(%+_)
/*————————————————————*/
/* panel for LVSAM EXEC */
/*————————————————————*/
  ¬ TYPE(PT)
  $ TYPE(NT)
  # TYPE(VOI)
  ı TYPE(VOI) JUST(RIGHT)
  ? AREA(SCRL)   EXTEND(ON)
)BODY EXPAND(//)
+-/-/-%VSAM DATASET INFORMATION+-/-/-
$Command ===>_ZCMD
$
?INFO
?
)AREA INFO
$&VSCOMP NAME: #TDSN                                        $
$  Catalog:    #CATLG                                  $
$
¬GENERAL DATA:                              CURRENT ALLOCATION:
$  Management class:      #MCLASS    $        Allocated trks:    ıATRK
$
$  Storage class:         #SCLASS    $        Allocated extents: ıEXTS
$
$   Volume serial(s):     #VOL1  #VOL2  #VOL3  #VOL4  #VOL5  #VOL6
$  Data class:            #DCLASS    $    ¬CURRENT UTILIZATION:
```

```
$   1st extent &ATYP     $#SPRI    $          Used trks:         ¬UTRK
$
$   Secondary &BTYP     $ #SSEC    $          Used percent:      ¬UPER
$
$   Data set name type:   #TYPE      $
$                                          ¬USAGE DATA:
$
$   Ave record length:    #ALEN      $          Total records:  ¬RTOT
$
$   Max record length:    #MLEN      $          Records deleted:¬RDEL
$
$   Key length:           #KLEN   $             Records inserted:¬RINS
$
$   Relative key position:#RKP     $           Records updated:¬RUPD
$
$   CI size:              #CISZ      $          Records retrieved:¬RRET
$
$   CI freespace:         #CIFR      $          CI splits:      ¬CISP
$
$   CA freespace:         #CAFR      $         $CA splits:      ¬CASP
$
$   Creation date:        #CDATE     $

#OPTC            #OPTD        #OPTE
$
#OPT1            #OPT2        #OPT3            #OPT4        #OPT5
#OPT6            #OPT7        #OPT8            #OPTA        #OPTB
)INIT
  &ZCMD = &Z
  .HELP = LVSAMH
)PROC
  IF (&ZCMD = CAN,CANCEL,EXIT)
      .RESP = END
)END
```

LVSAMH PANEL

```
)PANEL KEYLIST(ISRHLP2,ISR)
)ATTR DEFAULT(%+_)
/*———————————————————————————————————————————————————————————————————————*/
/* HELP for "VSAM Dataset Information" panel (LVSAMP) from LVSAM EXEC */
/*———————————————————————————————————————————————————————————————————————*/
  ¬ TYPE(PT)
  $ TYPE(NT)
  ¬ TYPE(ET)
)BODY WINDOW(7Ø,Ø9) CMD()
$
$   %LVSAM$displays information about a VSAM dataset by issuing a
$    a TSO LISTCAT command, then formatting the output for display
$    on an ISPF panel. LVSAM is only valid for a VSAM dataset.
$
$    When¬ENTER$is pressed the information is updated
```

```
$        ¬PF3/PF12$will EXIT from the display
$
)INIT
 &ZWINTTL = 'VSAM DATASET INFORMATION'
)PROC
)END
```

ISRUDLP AND ISRDULS0

The following two panels are not complete, but they show the changes required to the standard IBM panels. All the changes have comments to identify them. This will be obvious if you compare these with the original panels. Note that the real panels have (unprintable) attribute bytes for fields and I have changed them to blanks in these fragments of panels. You should leave all such attribute bytes exactly as they are in the original panels (and you do not have to add any).

Make the modifications as shown to (copies of) your standard panels. You may notice that I have added '&ZUSER on &ZSYSID' at the top of each panel. This is done not only because I find it useful information, but also because it makes it immediately obvious that it is a modified panel.

The original panels I have used are from Version 4.5 of ISPF. These two panels have not been changing much through the various ISPF Version 4 releases, thus the same changes can be applied to new panel versions when you upgrade your ISPF.

Note that the changes do not affect the functionality of these panels except when they are used by the DS EXEC, hence there is no problem in having them permanently allocated to DDname SPPLIB.

ISRUDLP PANEL

```
 )PANEL KEYLIST(ISRSAB,ISR)
/*-------------------------------------------------------------------------*/
/* &ZUSER on &ZSYSID added, so this is obviously not a standard panel */
/*-------------------------------------------------------------------------*/
/* if there is an option from the DS EXEC - press ENTER                    */
/* .. when returning to this panel - reset everything and press END        */
/* .. this process stops the user seeing this panel                        */
/* Note: the panel works normally if not invoked via the DS EXEC           */
/*-------------------------------------------------------------------------*/
```

```
)ATTR DEFAULT(   ) FORMAT(MIX)                                        /*  */
 ØB TYPE(AB)
 ØD TYPE(PS)
    ..
    ..
    ..
PDC DESC('Appendices') MNEM(1) ACTION RUN(TUTOR) PARM('ISRØØØØ4')
PDC DESC('Index') MNEM(2) ACTION RUN(TUTOR) PARM('ISR91ØØØ')
)ABCINIT
.ZVARS=EDMHELP
/*-------------------------------------------------------------------------*/
/* &ZUSER on &ZSYSID added in the first line of )BODY section              */
/*-------------------------------------------------------------------------*/
)BODY  CMD(ZCMD)
   Menu  RefList  RefMode  Utilities  Help                    &ZUSER on
&ZSYSID
 ------------------------------------------
                              Dataset List Utility
 Option ===> Z
    ..
    ..
    ..
&GRPBOX1 = 'Dataset list options'
.ATTR(GRPBOX1) = 'WIDTH(77) DEPTH(5)'
IF (&ZGUI = ' ')
  &MULTIPMT='Enter "/" to select option    '
ELSE
  &MULTIPMT='Check box to select option    '
                                       /*-------------------------------*/
VGET DSLACT                            /* get action from DS            */
IF (&DSLACT ¬= &Z OR &FIRSTIME = NO)   /*                               */
    IF (&FIRSTIME = &Z)                /* if first time here            */
        &FIRSTIME = NO                 /*  no longer first time         */
        .RESP = ENTER                  /* ..to display the list         */
    ELSE                               /* Returning from list           */
        &FIRSTIME = &Z                 /* reset &FIRSTIME               */
        &DSLACT   = &Z                 /* reset &DSLACT                 */
        VPUT DSLACT                    /*                               */
        .RESP = END                    /* simulate the END key          */
                                       /*-------------------------------*/
)REINIT
REFRESH(ZUPCDV ZMEMCONV)
.CURSOR = ZDLDSNLV
REFRESH(ZCMD ZDLDSNLV ZDLPVL ZUPCDV ZUPIVV ZMEMCONV)
)PROC
VGET (ZRDSN ZRVOL) SHARED
    ..
    ..
    ..
```

ISRDULS0 PANEL

```
)PANEL KEYLIST(ISRSPBC,ISR)
/*-------------------------------------------------------------------------*/
/* &ZUSER on &ZSYSID added, so this is obviously not a standard panel */
/* default Scroll changed from PAGE -> CSR                                 */
/*-------------------------------------------------------------------------*/
/* If there is an option from DS EXEC - insert it next to the first   */
/* dataset listed and simulate the ENTER key.                         */
/* On return - simulate the END key.                                  */
/* This invokes the option without the user ever seeing this panel    */
/* Note: this panel works normally if it is not invoked via DS EXEC   */
/*-------------------------------------------------------------------------*/
)ATTR DEFAULT(   ) FORMAT(MIX)                                  /*  */
 ØB TYPE(AB)
 ØD TYPE(PS)
    ..
    ..
    ..
PDC DESC('Appendices') MNEM(2) PDSEP(ON) ACTION RUN(TUTOR)
PARM('ISRØØØØ4')
PDC DESC('Index') MNEM(1) ACTION RUN(TUTOR) PARM('ISR91ØØØ')
)ABCINIT
.ZVARS=MEMLHELP
/*-------------------------------------------------------------------------*/
/* &ZUSER on &ZSYSID added in the first line of )BODY section         */
/*-------------------------------------------------------------------------*/
)BODY  CMD(ZCMD)
   Menu  Options  View  Utilities  Help                &ZUSER on &ZSYSID
 ----------------------------------------
 Z
 Command ===> Z                                        Scroll ===> Z

 ZDATA

)INIT
.ZVARS = '(ZDLTITLE ZCMD ZUSC)'
    ..
    ..
    ..
 IF (&DLI6 = ' ') &DLI6 = 'LOW'
 IF (&DLI8 = ' ') &DLI8 = 'HIGH'
```

Editor's Note: This article will be continued in the next edition.

Ron Brown
Systems Progammer (Germany)

MVS news

IBM has announced support for S/390 systems in Version 1.1 of its DB2 OLAP Server. Currently in beta, the software gets the S/390 to perform analysis and carry out business intelligence functions. The new version uses DB2 for OS/390 to maintain relational data, allowing the use of SQL programs and other relational tools to access and manage the data. Users can alternatively use the Essbase multidimensional store.

It runs on Unix System Services for OS/390 V2R5 and has the same functions as the DB2 OLAP Server Version 1.1 workstation product. Both are based on Hyperion Essbase 5.0.2. The components and add-on tools available with the workstation product are also available on the OS/390 version. Specific components include DB2 OLAP Server running on OS/390, client support running on 32-bit Windows connecting to the host via TCP/IP (shipped automatically with the server), and a host feature called Partitioning Option. The latter provides features to design and manage multidimensional databases, or cubes, that span OLAP applications or servers. Functions can integrate multiple physical cubes into a single logical cube, centrally administer and share metadata, allow connection of cubes with varying dimensionality, and allow partition replication between centralized and distributed cubes.

All of the components, except for the Partitioning Option, that can be ordered for the workstation product can be ordered for use with the DB2 OLAP Server for OS/390.

Contact your local IBM representative for further information.

* * *

NEON Systems has started shipping Version 4.5 of its Halo SSO, giving NT users a single signon tool for accessing S/390-based applications without the need for custom coding or installing software on PCs.

There is also a graphical user interface and a Windows API that can be called by C/C++, Java, and Visual Basic applications for OS/390-MVS security administration. The product now activates a new single sign-on passticket service allowing NT users with security clearance access to S/390 applications without keying in additional passwords.

Halo SSO consists of distinct NT and MVS components. The NT bit listens for account changes originated on the OS/390-MVS security system and sends changes to NT domains. The MVS part interfaces with OS/390 security packages (IBM RACF, CA-ACF2, or CA-Top Secret), listens for account change requests from remote NT servers, then synchronizes the mainframe and NT accounts.

For further information contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA.
Tel: (281) 491 4200
Fax: (281) 242 3880 or

Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middlesex, TW1 3RW, UK.
Tel: (0181) 607 9911
Fax: (0181) 607 9933.
http://www.neonsys.com

* * *