130

July 1997

In this issue

MVS update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: stevep_xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

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

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

Steve Piggott

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.

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

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.

Printed in England.

A utility to search for a range of numbers

UTILITY DESCRIPTION

There are a number of tools and utilities available to search for a specific string or strings, but hardly any that can search for a numeric range across files. On a number of occasions I have had to search for a range – for example, in locating hard-coded values in programs or to find control cards and sort cards that have been affected because of the introduction of a new field in a record description (imagine the effort needed to locate such cards!). Now I have a utility that scans for a range.

The format for using this utility is:

```
XAMIN <filename(s)> <LowerRange> <UpperRange>
```

where *filename* can be a flat file or PDS. Within a PDS, you can select all members or limit your selection by specifying a name with a wild character. The wild character can be a prefix or a suffix, or it can be in the middle. Only one wild character is permitted though, ie your selection can be UP*E, UPD*, or *DATE. *LowerRange* and *UpperRange* are numeric values for the lower range and upper range.

When you run the EXEC, it will prompt you for:

- A file name, if not passed as a parameter when run. (TSO rules apply, ie if the file name is not within quotes your user-id will be suffixed to the file name.)
- The lower and upper ranges, if not passed as parameters when run. (If the lower range is greater than the upper range, the ranges are flipped.)
- Any columns that you may want to ignore for the search. (Eg for COBOL programs you might want to ignore data in columns 1 to 6 and 73 to 80, and for JCL you might want to ignore columns 73 to 80.) You can type 'S' to analyse all columns.
- Whether you want to see the names of members that are being analysed on the screen, if your file for analysis is a PDS. (If you

have a large selection of members for analysis, you will know where you are!)

- Whether you want to analyse members created/modified only by you.

The output is created in a flat file and you will be put in browse mode to see the results.

You can modify the defaults to suit your needs. The default values are in the BuildDflts routine.

THE REXX EXEC

```
/* ---------------------- Rexx -------------------------- */
/* Rexx Exec To Scan for a range of numbers across a file  */
/* or a PDS                                                */
/* Format is                                               */
/*   XAMIN filename LowerRange UpperRange                  */
/* Filename can be a flat file or an entire pds or a       */
/* a partial selection of members                          */
/* ------------------------------------------------------- */
/* Naming Conventions Used:-                               */
/* Rexx Commands & Functions start with a Capital letter   */
/* TSO  Commands are in Upper Case                         */
/* User data (Variables/Constants/Labels) is a combination */
/* Of Upper & Lower case and                               */
/*  Alphanumeric user data starts with x  (lower x)        */
/*  Numeric      user data starts with n  (lower n)        */
/*  Switches / Flags       start  with s  (lower s)        */
/*  Rexx Labels Start with a Capital Letter                */
/* ------------------------------------------------------- */
Trace O
Arg xFromDsn nLRange nURange .
Call BuildDflts                          /* Defaults are Here !!!    */
Call InitVarbls                          /* Defaults are Here !!!    */
If xFromDsn='' Then Call GetInpDsn
Call CheckInpDsn
Call GetRange
Call GetOthrDflts
Call AllocOut
If xType='Pds' Then Call AnalyzePds
  Else Call Analyze
If sHits='Y' Then Call BrowseHits
  Else Call NoMemSelctd
Exit
/*------------  Start Of Subroutines -------------------*/
GetInpDsn:
```

```
'CLRSCRN'
Say Center('Type The Input File Name - TSO Rules Apply',72)
Say Center('ie If The Name Is Not In Quotes, 'Userid() 'Will Be Suffixed',72)
Say Center('-------------------------------------------------------',72)
Say Center('Input can be a flat file Or a PDS, If PDS,',72)
Say Center('Type just the PDS name to analyse all members or',72)
Say Center('Choose a pattern for partial selection - Use * as a wild character',72)
Say Center('Note:- Only one wild character allowed for partial selection',72)
Say Center('Example: Your selection can be  SA*ENA OR NB* OR *ASK',72)
Pull xFromDsn .
xFromDsn=Strip(Translate(xFromDsn,"'",'"'))
Select
 When xFromDsn='' Then Do
  Say '*Error* You Have To Type A Name '
  Say '        Program Aborted'
  Exit
  End
 When Left(xFromDsn,1)\="'" Then Do
  xFromDsn="'"Userid()'.'xFromDsn"'"
  End
 When Left(xFromDsn,1)="'" & Right(xFromDsn,1)\="'" Then Do
  xFromDsn=xFromDsn"'"
  End
 Otherwise Nop
 End
Return
CheckInpDsn:
xMemPat=''
If Pos('(',xFromDsn)>0 Then Do
  Parse Var xFromDsn  xFromDsn '(' xMemPat ')' .
  If Verify(xMemPat,'*')=0 Then xMemPat=''
  xFromDsn=xFromDsn"'"
  End
xAvail=SYSDSN(xFromDsn)
If xAvail\='OK' Then Do
  Say '*Error*' xFromDsn 'Does Not Exist'
  Say '        Program Aborted'
  Exit
  End
Return
GetRange:
If nLRange='' | nURange='' Then Do
  Say Center('The Default Lower Range Is ' nLowerRange,72)
  Say Center('The Default Upper Range Is ' nUpperRange,72)
  Say Center('Key In New Lower & Upper Range To Override',72)
  Pull nLRange nURange .
  End
If nLRange\='' Then nLowerRange=nLRange
If nURange\='' Then nUpperRange=nURange
Select
```

```
 When \Datatype(nLowerRange,'W') Then xMsg='Lower Range Has To be In Numbers'
 When \Datatype(nUpperRange,'W') Then xMsg='Upper Range Has To be In Numbers'
 When nLowerRange>nUpperRange Then Do
  nSaveRange=nUpperRange
  nUpperRange=nLowerRange
  nLowerRange=nSaveRange
  End
 Otherwise Nop
 End
If xMsg\='' Then Do
  Say '*Error*' xMsg
  Say 'Program Aborted'
  Exit
  End
nLowerLen=Length(nLowerRange)
nUpperLen=Length(nUpperRange)
xAvailRc=LISTDSI(xFromDsn NORECALL)
If Rc<4 Then Do
  Select
   When Left(SYSDSORG,2)='PO' Then Do
    nLrecl=SYSLRECL
    xType='Pds'
    End
   When Left(SYSDSORG,2)='PS' Then Do
    nLrecl=SYSLRECL
    xDsName=xFromDsn
    xType='Seq'
    End
   Otherwise Do
    Say '*Error* The Dataset Organization Of' xFromDsn
    Say '        Is Not Supported By This Rexx'
    Say '        Program Aborted'
    Exit
    End
   End
  End
 Else Do
  Say '*Error* Fatal Error While Accessing ' xFromDsn
  Say '        Program Aborted'
  Exit
  End
Return
GetOthrDflts:
'ClrScrn'
xSaveIgnCols=xIgnoreCols
Do Forever
 Say Center('Do You Want To Ignore Data In Any Columns',72)
 Say Center('The Default is ' xIgnoreCols,72)
 Say Center('To Override, Type New Columns As nn-nn <nn-nn>.... OR',72)
 Say Center('Type "S" To Analyse All Columns OR',72)
```

```
 Say Center('Press Blank To Accept Defaults',72)
 Pull xAns
 If xAns='' | xAns='S' Then Do
   If xAns='S' Then xIgnoreCols=''
   Leave
   End
 xNewAns=Translate(xAns,' ','-')
 xIgnoreCols=''
 If Words(xNewAns)//2=1 Then xNewAns=xNewAns Word(xNewAns,Words(xNewAns))
 Do nI=1 By 1 While nI<=Words(xNewAns)
  xCrntWrd=Word(xNewAns,nI)
  sRangeErr='N'
  If Datatype(xCrntWrd,'W') & xCrntWrd>0 Then Do
    xIgnoreCols=xIgnoreCols||xCrntWrd
    If nI//2=1 Then Do
       xIgnoreCols=xIgnoreCols||'-'
       nLowNum=xCrntWrd
       End
      Else Do
       xIgnoreCols=xIgnoreCols||' '
       nHiNum=xCrntWrd
       If nLowNum>nHiNum Then sRangeErr='Y'
       End
    End
   Else sRangeErr='Y'
  If sRangeErr='Y' Then Do
    Say '*Error* Invalid Range Keyed In - Please Key In Correctly'
    xIgnoreCols=xSaveIgnCols
    Leave nI
    End
  End
  If sRangeErr='N' Then Leave
 End
If xAns='S' Then Nop
 Else Do
  Say '------------------------Note-----------------' Copies('-',Length(xIgnoreCols))
  Say 'Data In The Following Columns Will Be Ignored' xIgnoreCols
  Say '------------------------Note-----------------' Copies('-',Length(xIgnoreCols))
  Say
  End
If xType='Pds' Then Do
  Say Center('Do You Want To See The Members That Are Being Analysed?',72)
  Say Center('The Default is ' xExpnd.sWatch,72)
  Say Center('Type "Y" or "N" or Press Blank To Accept Default',72)
  Pull xAns
  Select
   When Translate(xAns)='N' Then sWatch='N'
   When xAns='' Then Nop
   When Translate(xAns)='Y' Then sWatch='Y'
   Otherwise Nop
```

```
   End
  Say Center('Do You Want To Analyse Only Members Created By You?',72)
  Say Center('The Default is ' xExpnd.sOnlyMe,72)
  Say Center('Type "Y" or "N" or Press Blank To Accept Default',72)
  Pull xAns
  Select
   When Translate(xAns)='N' Then sOnlyMe='N'
   When xAns='' Then Nop
   When Translate(xAns)='Y' Then sOnlyMe='Y'
   Otherwise Nop
   End
  End
Return
AnalyzePds:
If Pos('*',xMemPat)>0 Then Do
  Select
   When Left(xMemPat,1)='*' Then Do
    xSufx=Strip(Translate(xMemPat,'','*'))
    nLimit=Words(xPrfx)>1
    End
   When Right(xMemPat,1)='*' Then Do
    xPrfx=Strip(Translate(xMemPat,'','*'))
    nLimit=Words(xSufx)>1
    End
   Otherwise Do
    xTemp=Strip(Translate(xMemPat,'','*'))
    nLimit=Words(xTemp)>2
    Parse Var xTemp xPrfx xSufx .
    End
   End
  If nLimit>0 Then Do
    Say '*Error* Invalid Selection Pattern Found - Pattern = ' xMemPat
    Say 'Program Aborted'
    Exit
    End
  End
'ISPEXEC LMINIT DATAID(xNameInp) DATASET('xFromDsn') ENQ(SHR)'
If Rc\=0 Then Do
  Say 'Unable to Access' xFromDsn
  Exit
  End
'ISPEXEC LMOPEN DATAID('xNameInp') OPTION(INPUT)'
If Rc\=0 Then Do
  Say 'Unable to Open' xFromDsn
  Exit
  End
Do Forever
 'ISPEXEC LMMLIST DATAID('xNameInp') OPTION(LIST) MEMBER(xNextMem) STATS(YES)'
 If Rc\=0 Then Leave
 xNextMem=Strip(xNextMem)
```

```
 sThisMem='N'
 Select
  When xMemPat='' Then sThisMem='Y'
  When Pos('*',xMemPat)=0 & xMemPat=xNextMem Then sThisMem='Y'
  When xSufx\='' & xPrfx\='' Then Do
   If Right(xNextMem,Length(xSufx))=xSufx &,
      Left(xNextMem,Length(xPrfx))=xPrfx Then sThisMem='Y'
   End
  When xSufx \='' & Right(xNextMem,Length(xSufx))=xSufx Then sThisMem='Y'
  When xPrfx \='' & Left(xNextMem,Length(xPrfx))=xPrfx Then sThisMem='Y '
  Otherwise Nop
  End
 If sThisMem='N' Then Iterate
 If sOnlyMe='Y' & zluser\=Userid() Then Do
   If zluser='' Then zluser='*NO ID*'
   If sWatch='Y' Then Say xNextMem 'Not Analysed As It Was Created/Modified By' zluser
   Iterate
   End
 sHits='Y'
 sFirst='Y'
 xDsName=Strip(xFromDsn,'t',"'")'('xNextMem")'"
 Call Analyze
 End
'ISPEXEC LMCLOSE DATAID('xNameInp')'
Return
Analyze:
If sWatch='Y' Then Say 'Analysing' xDsName
Address 'TSO'
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
'ALLOC DD(xInpDsn) DSN('xDsName') SHR KEEP'
If Rc\=0 Then Do
  Say '*Error* Unable To Allocate ' xDSName
  Say '        Program Aborted - Return Code From Allocate = ' Rc
  Exit
  End
'EXECIO * DISKR xInpDsn (STEM XREC.'
If Rc\=0 Then Do
  Say '*Error* Unable To Read'  xDSNameRc
  Say '        Program Aborted - Return Code From Read = ' Rc
  Exit
  End
'EXECIO 0 DISKR xInpDsn (FINIS'
nRec.0=xRec.0
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
Do nI=1 By 1 While nI<=nRec.0
 xData=xRec.nI
```

```
 If xIgnoreCols\='' Then Do
   Do nJ=1 By 1 While nJ<=Words(xIgnoreCols)
     xCrntWrd=Word(xIgnoreCols,nJ)
     Parse Var xCrntWrd nStartCol '-' nEndCol
     If nEndCol='' Then nEndCol=nStartCol
     nLenCol=nEndCol-nStartCol+1
     xData=Overlay(Copies(' ',nLenCol),xData,nStartCol,nLenCol)
     End
   End
 nStart=Verify(xData,'0123456789','M')
 If nStart=0 Then Iterate
 Do Forever
  sFound='N'
  xRange1=Substr(xData,nStart,nLowerLen)
  xRange2=Substr(xData,nStart,nUpperLen)
  say xRange1 xRange2
  If Datatype(xRange1,'W') Then Do
    sInRange=(nLowerRange<=xRange1) & (xRange1<=nUpperRange)
    If sInRange Then sFound='Y'
    End
  Select
   When xRange1=xRange2 Then Nop
   When sFound='Y' Then Nop
   Otherwise Do
    If Datatype(xRange2,'W') Then Do
      sInRange=(nLowerRange<=xRange2) & (xRange2<=nUpperRange)
      If sInRange Then sFound='Y'
      End
    End
   End
  If sFound='Y' Then Do
    Call RiteDetl
    Iterate nI
    End
  nStart=Verify(xData,'0123456789','M',nStart+1)
  If nStart=0 Then Iterate nI
  End
 End
Return
RiteDetl:
If sFirst='Y' Then Do
  If nOutRecs=0 Then Do
    xLine.1='Search Lower Range is ' nLowerRange 'and Upper Range is' nUpperRange
    xLine.1=Center(xLine.1,72)
    Call RiteLine
    End
  If xType='Pds' Then xLine.1=Center('PDS Name: ' xFromDsn ' Member:' xNextMem,72)
    Else  xLine.1=Center('DSname: ' xFromDsn,72)
  Call RiteLine
  xLine.1=' Line #' Center('* * * * *',72)
```

```
  Call RiteLine
  sFirst='N'
  End
xLine.1=Right(nI,7)':' Strip(xData,'t',' ')
If Length(xLine.1)>255 Then xLine.1=Left(xLine.1,255)
Call RiteLine
Return
RiteLine:
'EXECIO 1 DISKW xOutDsn  (STEM xLine.'
If Rc\=0 Then Do
  Say '*Error* Write Failure On ' xFileOut 'RC='rc
  Say '        Program Aborted'
  xSamFir=MSG('OFF')
  'FREE DD(xOutDsn)'
  xAskNb=MSG(xSamFir)
  Exit 16
  End
nOutRecs=nOutRecs+1
Return
AllocOut:
xOutName="'"xOutName"'"
xAvail=SYSDSN(xOutName)
If xAvail='OK' Then Do
  xSamFir=MSG('OFF')
  'DELETE' xOutName
  xAskNb=MSG(xSamFir)
  If Rc\=0 Then Do
    Say '*Error*' xOutName 'Could Not Be Deleted - Return Code = ' Rc
    Say 'Program Aborted'
    Exit
    End
  End
If nLrecl+7<133 Then nLrecl=132
  Else nLrecl=255
xDfltDisp='NEW UNIT(SYSDA) LRECL('nLrecl') SPACE(20) DSORG(PS) RECFM(F,B) TRACKS  RELEASE'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'ALLOCATE DSN('xOutName') DD(xOutDsn)' xDfltDisp
If Rc\=0 Then Do
  Say '*Error* Unable To Alloc' xOutName
  Say 'Return Code Is ' Rc
  Exit 16
  End
Return
BrowseHits:
'EXECIO 0 DISKW xOutDsn (FINIS'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
```

```
'Clrscrn'
If nOutRecs>Ø Then Do
  'ISPEXEC BROWSE DATASET ('xOutName')'
  End
 Else Do
  Say '*Warning*  No Hits  No hits  No Hits For Your Search'
  Say 'PDS Scanned :- ' xFromDsn
  Say 'Lower Range:-'nLowerRange  'Upper Range:-' nUpperRange
  End
Return
NoMemSelctd:
Say '*Error* No Members Found In Pds' xFromPds 'That Could Match' xMemPat
Say '        Program Is Unable To Do Any Analysis'
Return
InitVarbls:
nLrecl=Ø
nOutRecs=Ø
sFirst='Y'
sHits='N'
xExpnd.N='No'
xExpnd.Y='Yes'
xMsg=''
xPrfx=''
xSufx=''
Return
/* ------Note---------------Note----------------------Note-------- */
/* The Defaults are defined here. Modify them to suit your          */
/* installation's standards.                                        */
/* ---------------------------------------------------------------- */
BuildDflts:
xOutName=Userid()'.SEARCH.LIST'       /* Output Dataset Name         */
xIgnoreCols='Ø1-Ø6 73-8Ø'              /* Ignore Data in these columns*/
sWatch='N'                            /* Y will display memname      */
sOnlyMe='N'                           /* Y = select Only My Members  */
                                      /* N = select entire Pds       */
nLowerRange=34                        /* From This Number     */
nUpperRange=786                       /* To This Number       */
Return
```

Moyeen Ahmed Khan
Senior Systems Designer
Manulife Financial (Canada)

TSO TRANSMIT ISPF front-end

THE PROBLEM

With the use of TSO it is possible to transmit data to other users or destinations. TSO, however, lacks a decent user interface to the TRANSMIT command. If a user wishes to send data to another destination, he or she must remember the syntax of the command. Also, when sending data to VM destinations, it must be sent sequentially or it will be unreadable when it arrives, and load modules must not be sent sequentially, otherwise the directory information will be discarded and the load module will be unusable once it is received. These and many other things can be very confusing to the end user, and therefore we have decided at our shop to write a front-end application to TSO TRANSMIT to make it more intuitive (not to mention user-friendly).

THE SOLUTION

When the user enters the front-end application, a panel will be presented, and then they will be able to enter the dataset and destination to send the data to. The user can specify whether the entire dataset is to be sent: if so, when PF3 is pressed, the dataset is transmitted. If only some members are to be transmitted, the user will be placed into a member list, where the members can be selected for inclusion. The members can also be browsed prior to selection to ensure that it is actually the member that is meant to be sent. Once the members are selected, it is verified that the member has not already been selected, so that it is not transmitted more that once. As you can already see, there are tremendous advantages to such a front-end application to the end-user community.

The code for the program was written using the high-level Assembler, to take advantage of some of the new features (some of which are available using the SHARE SLAC mods). The code uses macros to perform condition checking (ie IF...THEN...ELSE, DO, etc), which are available from many sources. These macros may be available when using the high-level Assembler toolkit; however, I have not verified this.

L4JTRANS SOURCE CODE

```
L4JTRANS CSECT
L4JTRANS AMODE 31
L4JTRANS RMODE ANY
         SAVE  (14,12),,'L4JTRANS &SYSDATE &SYSTIME'
         LR    R12,R15
         USING L4JTRANS,R12,R9
         LA    R9,4095(R12)
         LA    R9,1(R9)
         GETMAIN RU,LV=WORKL,SP=77,LOC=ANY
         LR    R11,R1
         USING WORK,R11
         GETMAIN R,LV=80,SP=77
         ST    R1,MSGADDR
         ST    R13,SAVEAREA+4
         LA    R1,SAVEAREA
         ST    R1,8(,R13)
         LR    R13,R1
         LA 2,MEM_LIST_STORE              Initialize
         LA 3,L'MEM_LIST_STORE                    member list
         SLR 5,5
         MVCL 2,4
         MVI   MEM,C' '                   Clear MEM field
         MVC   MEM+1(7),MEM               Clear MEM field
         XC    SRCHTAB,SRCHTAB            Clear table
         MVI   BLANKS,C' '                Initialize
         MVC   BLANKS+1(L'BLANKS-1),BLANKS        blanks variable
         LOAD  EP=ISPLINK                 Get address of ISPLINK
         ST    R0,ISPLINKA                Save ISPLINK address
         LA    R10,CALLL
*-------------------------------------------------------------------*
* CONTROL ERRORS RETURN                                             *
*-------------------------------------------------------------------*
         MVC   CALLL(CALLML),CALLM
         L     R15,ISPLINKA           Set CONTROL
         CALL  (15),(CONTROL,                    ERRORS              X
               ERRORS,                                 RETURN        X
               RETURN),VL,MF=(E,(R10))
*-------------------------------------------------------------------*
* VDEFINE                                                           *
*-------------------------------------------------------------------*
         MVC   CALLL(CALLML),CALLM
         L     R15,ISPLINKA
         CALL  (15),(VDEFINE,       Issue                           X
               NAMELIST,                 VDEFINE to                X
               VARAREA,                          define              X
               VARFMTS,                               vars           X
               VARLENS,                                   needed     X
               OPTIONS),VL,MF=(E,(R10))
```

```
*--------------------------------------------------------------------*
* VGET (NODE USERID DS1 M PRJ1 LIB1 TYP1)                            *
         MVC   CALLL(CALLML),CALLM        VGET vars
         L     R15,ISPLINKA                        from PROFILE
         CALL  (15),(VGET,NAMES1,PROFILE),VL,MF=(E,(R1Ø))        pool
*--------------------------------------------------------------------*
* Infinite Loop of Displays and Transmits, etc.                      *
*--------------------------------------------------------------------*
      DO INF
* DISPLAY PANEL(P4JTRANS)
         MVC   CALLL(CALLML),CALLM        Display
         L     R15,ISPLINKA                      main transmit
         CALL  (15),(DISPLAY,@P4JTRAN,,@DS1),VL,MF=(E,(R1Ø))    panel
         IF (C,15,EQ,=F'12')
            MVC   CALLL(CALLML),CALLM
            L     R15,ISPLINKA
            MVC ZMSGØØØS(24),=CL24'L4JTRANS - Panel Error'
            MVC ZMSGØØØL(8Ø),=CL8Ø'Panel P4JTRANS not found - L4JTRANS X
               terminated.'
            CALL  (15),(SETMSG,ISPZØØØ),VL,MF=(E,(R1Ø))
            B     EXIT
         ENDIF
         DOEXIT C,15,EQ,=F'8'
*--------------------------------------------------------------------*
* VPUT (NODE USERID DS1 M PRJ1 LIB1 TYP1)                            *
         MVC   CALLL(CALLML),CALLM        Update
         L     R15,ISPLINKA                      vars in
         CALL  (15),(VPUT,NAMES1,PROFILE),VL,MF=(E,(R1Ø)) PROFILE POOL
* Initialize some fields
         LA    R2,MEM_LIST_STORE          Initialize
         LA    R3,L'MEM_LIST_STORE                 member
         SLR   R5,R5                                 list
         MVCL  R2,R4                                      area
         MVI   XMIT_DATASET,C' '          Clear field
         MVC   XMIT_DATASET+1(L'XMIT_DATASET-1),XMIT_DATASET
         MVI   BLANK_DATASET,C' '         Clear field
         MVC   BLANK_DATASET+1(L'BLANK_DATASET-1),BLANK_DATASET
         IF (CLC,DS1,EQ,BLANKS)           Other DSN blank  ... then Do
            LA    R4,XMIT_DATASET         Address of XMIT dataset field
            MVI   Ø(R4),X'7D'             put quote at start of dataset
            LA    R4,1(R4)                Increment pointer
            LA    R6,PRJ1                 Address of PROJECT field
            BAS   R14,BUILD_DATASET_NAME  Move in prject to DSN
            MVC   Ø(2,R4),=C'.'           move in . at end of qualifier
            LA    R4,1(R4)                increment pointer
            LA    R6,LIB1                 Address of GROUP field
            BAS   R14,BUILD_DATASET_NAME  Move in group to DSN
            MVC   Ø(2,R4),=C'.'           Move in '.'
            LA    R4,1(R4)                Increment pointer
            LA    R6,TYP1                 Address of TYPE field
```

```
            BAS   R14,BUILD_DATASET_NAME  Move in type to DSN
            MVI   Ø(R4),X'7D'             Move in ending quote
            B     DOINIT                  Proceed
         ELSE
            MVC XMIT_DATASET(56),DS1      If ds1 NE blanks, use it
         ENDIF
*-------------------------------------------------------------------*
* If other dataset name field is filled in, separate the member      *
*  from the dataset name.                                           *
*-------------------------------------------------------------------*
         LA    R3,XMIT_DATASET            Address of dataset name
         MVI   SRCHTAB+C'(',C'('          Set up table
         TRT   Ø(56,R3),SRCHTAB           Search for "("
         BZ    DOINIT                     Proceed .... if not found
         LR    R4,R1
         LA    R1,56                      Length of dsn
         LR    R2,R4
         SR    R2,R3
         SR    R1,R2
         MVI   SRCHTAB+C'(',X'ØØ'         Remove "(" from table
         MVI   SRCHTAB+C')',C')'          Set up table
         BCTR  R1,Ø                       Decrement for EX
         EX    R1,CMPR                    Search for ")"
         BZ    ERROR1                     No ")" ... Issue error msg
         LR    R5,R1
         SR    R5,R4
         LA    R6,1(,R5)
         BCTR  R5,Ø                       Decrement for EX
         BCTR  R5,Ø                       Decrement for EX
         EX    R5,MOVEMEM                 Get member name
         EX    R6,MOVEBLNK                Move in blanks
         CLI   Ø(R3),X'7D'                Quote ?
         BNE   DOINIT                     No ... Proceed
         MVI   Ø(R4),X'7D'                Move in quote
         B     DOINIT                     Proceed
*-------------------------------------------------------------------*
*     Issue error message if unbalanced ")" in dataset name         *
*-------------------------------------------------------------------*
ERROR1   EQU   *
         MVC   CALLL(CALLML),CALLM
         L     R15,ISPLINKA
         MVC ZMSGØØØS(24),=CL24'Invalid DSN - Syntax'
         MVC ZMSGØØØL(8Ø),=CL8Ø'The dataset name contains unbalanced paX
               rentheses'
         CALL  (15),(SETMSG,ISPZØØØ),VL,MF=(E,(R1Ø))
         SLR 15,15                       RC = Ø
         B     DOFREE                     Continue
CMPR     TRT   1(Ø,R4),SRCHTAB
MOVEMEM  MVC   MEM(Ø),1(R4)
MOVEBLNK MVC   Ø(Ø,R4),BLANKS
```

```
*--------------------------------------------------------------------*
* LMINIT DATAID(DSID) DATASET(TST) ENQ(SHRW)                         *
*--------------------------------------------------------------------*
DOINIT   EQU   *
         IF (CLI,M,EQ,C'Y')              All members ?
            BAS   R14,TRANSMIT_FUNCTION  Yes ... then branch
            B   DOFREE                   All done ... branch
         ENDIF
         IF (CLC,DS1,EQ,BLANKS)          Other dsn blank ?
            MVC   CALLL(CALLML),CALLM    Issue LMINIT
            L     R15,ISPLINKA                 for PRJ GRP TYP
            CALL  (15),(LMINIT,                                      X
               @DSID,PRJ1,LIB1,,,,TYP1,,,,,SHRW),VL,MF=(E,(R10))
            LTR   R15,R15                RC=0
            BZ    DOOPEN                 Yes ... Then proceed
            MVC   CALLL(CALLML),CALLM
            L     R15,ISPLINKA
            MVC ZMSG000S(24),ZERRSM      ISPF Short error msg
            MVC ZMSG000L(80),ZERRLM      ISPF Long error msg
            CALL  (15),(SETMSG,ISPZ000),VL,MF=(E,(R10))  Issue msg
         ELSE                            Other DSN not blank ?
            MVC   CALLL(CALLML),CALLM    Issue LMINIT
            L     R15,ISPLINKA                 for other dsn field
            CALL  (15),(LMINIT,                                      X
               @DSID,,,,,,,XMIT_DATASET,,XVOL,,SHRW),VL,MF=(E,(R10))
            LTR   R15,R15                RC=0
            BZ    DOOPEN                 Yes ... Then proceed
            MVC   CALLL(CALLML),CALLM
            L     R15,ISPLINKA
            MVC ZMSG000S(24),ZERRSM      ISPF Short error msg
            MVC ZMSG000L(80),ZERRLM      ISPF Long error msg
            CALL  (15),(SETMSG,ISPZ000),VL,MF=(E,(R10))  Issue msg
         ENDIF
         B RELOOP
*--------------------------------------------------------------------*
* LMOPEN DATAID(&DSID)                                               *
*--------------------------------------------------------------------*
DOOPEN   EQU   *
         MVC   CALLL(CALLML),CALLM       Open
         L     R15,ISPLINKA                    the
         CALL  (15),(LMOPEN,DSID),VL,MF=(E,(R10))    dataset
         LTR   R15,R15                   RC = 0 ?
         BZ    FIRSTDIS                  yes ... branch
         MVC ZMSG000S(24),ZERRSM         ISPF Short error msg
         MVC ZMSG000L(80),ZERRLM         ISPF Long error msg
         MVC   CALLL(CALLML),CALLM       Issue
         L     R15,ISPLINKA                    Error
         CALL  (15),(SETMSG,ISPZ000),VL,MF=(E,(R10))  message
         B     DOLMFREE                Branch to LMFREE routine
FIRSTDIS EQU   *
```

```
         MVI   TOP,C' '                    Initialize top field
         MVC   TOP+1(L'TOP-1),TOP           so we are place at first mbr
         DO INF
*--------------------------------------------------------------------*
* LMMDISP DATAID(&DSID) OPTION(DISPLAY) TOP(&TOP)                    *
*--------------------------------------------------------------------*
      MVC   CALLL(CALLML),CALLM
      L     R15,ISPLINKA
      CALL  (15),(LMMDISP,DSID,DISPLAY,MEM,,@MEMLIST,,TOP,,,ANY),      X
               VL,MF=(E,(1Ø))
      MVC   TOP(8),ZLMEMBER
      IF C,15,EQ,=F'4'             Empty dataset ?
         MVC ZMSGØØØS(24),ZERRSM       ISPF Short error msg
         MVC ZMSGØØØL(8Ø),ZERRLM       ISPF Long error msg
         MVC CALLL(CALLML),CALLM          Issue
         L 15,ISPLINKA                          error
         CALL (15),(SETMSG,ISPZØØØ),VL,MF=(E,(1Ø))     message
         LA 15,8                       RC = 8
      ENDIF
*--------------------------------------------------------------------*
* Figure out what was done in member list                            *
*--------------------------------------------------------------------*
      IF C,15,EQ,=F'8'                 PF3 (END) pressed ?
         LA    R3,MEM_LIST_STORE       Address of member list
         USING MEM_LIST,R3
         IF CLC,MEM_LIST_CUR,EQ,=F'Ø'  No members entered ?
            B    DOFREE                exit
         ENDIF
         LA    R3,MEM_LIST_STORE       Address of member list
         BAS   14,TRANSMIT_FUNCTION    Transmit members
         B   DOFREE                    exit
      ENDIF
      DOEXIT LTR,15,15,NZ
      SLR 15,15
      DO WHILE=(LTR,15,15,Z)
*--------------------------------------------------------------------*
* If "S *" was entered in member list, XMIT entire dataset rather    *
*   then selecting all the members individually.                     *
*--------------------------------------------------------------------*
         LA    R1,XMITZCMD             Get address of command
         OI    Ø(R1),X'4Ø'             Make upper case
         IF (CLC,MEM(8),EQ,=CL8' ')    mem pattern specified ie (A*) ?
            IF (CLC,XMITZCMD(3),EQ,=CL3'S *')    is command "S *"
               MVI   M,C'Y'                turn on xmit full dsn flag
               BAS   R14,TRANSMIT_FUNCTION   XMIT the dataset
               MVI   M,C'N'                reset flag
               B   DOFREE                  Do cleanup and leave
            ENDIF
         ENDIF
*--------------------------------------------------------------------*
```

```
* Can was entered                                                      *
*---------------------------------------------------------------------*
         IF (CLC,ZCMD(3),EQ,=CL3'CAN')    If cmd is CAN, don't process
            MVC ZCMD(133),=CL133' '       Clear cmd field
            B   DOFREE                    do cleanup and leave
         ENDIF
*---------------------------------------------------------------------*
* Browse member
*---------------------------------------------------------------------
         IF (CLI,ZLLCMD,EQ,C'B')          B entered beside member ?
            MVC   CALLL(CALLML),CALLM          then
            L     R15,ISPLINKA                  browse the
            CALL (15),(BROWSE,,,,,DSID,ZLMEMBER),MF=(E,(10)) member
            SLR 15,15                     RC = 0
         ELSE
*---------------------------------------------------------------------*
* Select member for XMIT                                              *
*---------------------------------------------------------------------*
            IF (CLI,ZLLCMD,EQ,C'S')       S entered beside member ?
               IF (CLI,SEQ,EQ,C'Y')       Send sequentially
                  LA 3,MEM_LIST_STORE
                  IF CLC,=F'0',NE,MEM_LIST_CUR-MEM_LIST(3)
                     MVC ZMSG000S(24),=CL24'Selection Error'
                     MVC ZMSG000L(80),=CL80'Select only ONE member for X
               sequential transmission.  Press PF3 to XMIT.'
                      MVC CALLL(CALLML),CALLM          Issue
                      L 15,ISPLINKA                          error
                      CALL (15),(SETMSG,ISPZ000),VL,MF=(E,(10))     msg
                      B     VERIFY_SELECTION
                   ENDIF
                ENDIF
                BAS   R14,CHK_UNIQUENESS  Check to see not already added
                IF LTR,15,15,Z             If RC=0  then ....
                   BAS   R14,ADD_MEM_LIST    add member to member list
VERIFY_SELECTION  EQU  *
*---------------------------------------------------------------------*
* Too many members entered - Issue error message                      *
                  IF LTR,15,15,NZ
                     MVC ZMSG000S(24),=CL24'Too Many Selections'
                     MVC ZMSG000L(80),=CL80'The Maximum Number of Selecx
               tions has been reached. Selection not recorded.'
                     MVC   CALLL(CALLML),CALLM         Issue
                     L     R15,ISPLINKA                     error
                     CALL (15),(SETMSG,ISPZ000),VL,MF=(E,(10))    msg.
                     LA    R15,8             RC = 8
                  ENDIF
               ELSE
*---------------------------------------------------------------------*
* Member already in member list - Issue error message                 *
*---------------------------------------------------------------------*
```

```
               MVC ZMSG000S(24),=CL24'Duplicate Selection'
               MVC ZMSG000L(80),=CL80'The member has been selected ax
            lready.  Select another or PF3 to XMIT'
               MVC   CALLL(CALLML),CALLM           Issue
               L     R15,ISPLINKA                      error
               CALL (15),(SETMSG,ISPZ000),VL,MF=(E,(10))       msg.
               LA    R15,8             RC = 8
            ENDIF
         ELSE
*---------------------------------------------------------------------*
* Invalid selection - Issue error message                             *
*---------------------------------------------------------------------*
            MVC ZMSG000S(24),=CL24'Invalid Selection'
            MVC ZMSG000L(80),=CL80'Enter either S for Select or B fox
            r Browse'
            MVC   CALLL(CALLML),CALLM           Issue
            L     R15,ISPLINKA                      error
            CALL (15),(SETMSG,ISPZ000),VL,MF=(E,(10))          msg.
            LA    R15,8               RC = 8
         ENDIF
      ENDIF
*---------------------------------------------------------------------*
* Get member name                                                     *
*---------------------------------------------------------------------*
      DOEXIT C,15,EQ,=F'8'             Leave when END is entered
      MVC   CALLL(CALLML),CALLM        Get the
      L     R15,ISPLINKA                       member name
      CALL (15),(LMMDISP,DSID,GET),MF=(E,(10))   that was selected
      IF (LTR,15,15,Z)                 RC = 0 for get member name
         MVC   TOP(8),ZLMEMBER         Move member name
      ENDIF
   ENDDO
   SLR   R15,R15                       RC = 0
ENDDO
DOFREE EQU *
*---------------------------------------------------------------------*
* LMMDISP DATAID(&DSID) OPTION(FREE)                                  *
*---------------------------------------------------------------------*
      MVC   CALLL(CALLML),CALLM        Issue
      L     R15,ISPLINKA                    LMMDISP to free
      CALL  (15),(LMMDISP,DSID,FREE),VL,MF=(E,(10))     the DATAID
*---------------------------------------------------------------------*
* LMCLOSE DATAID(&DSID)                                               *
      MVC   CALLL(CALLML),CALLM        Close
      L     R15,ISPLINKA                  the
      CALL  (15),(LMCLOSE,DSID),VL,MF=(E,(R10))     dataset
DOLMFREE EQU *
*---------------------------------------------------------------------*
* LMFREE  DATAID(&DSID)                                               *
*---------------------------------------------------------------------*
```

```
         MVC   CALLL(CALLML),CALLM        Issue
         L     R15,ISPLINKA                     LMFREE to
         CALL  (15),(LMFREE,DSID),VL,MF=(E,(R1Ø))   Free up dataid
RELOOP EQU *
      ENDDO
*-------------------------------------------------------------------*
* Free up storage an return to caller
*-------------------------------------------------------------------
EXIT     EQU   *
         L     R2,MSGADDR                 R1 = start address of storage
         FREEMAIN R,LV=8Ø,A=(R2),SP=77    Free storage
         L     R13,SAVEAREA+4
         FREEMAIN RU,LV=WORKL,A=(R11),SP=77    Free storage
         RETURN  (14,12),RC=(15)          Return to caller
*-------------------------------------------------------------------*
* BUILD_DATASET_NAME - Create dataset name form PRJ1 LIB1 and TYP1    *
*-------------------------------------------------------------------*
BUILD_DATASET_NAME EQU *
         XC    XMASK,XMASK              Clear Table
         MVI   XMASK+C' ',C' '          Set up Table
         TRT   Ø(8,R6),XMASK            Search for First blank
         BNZ   LVL_FULL                 Branch if Len NE Ø
         LA    R1,8                     Otherwise len is 8 chars
         B     LVL_FULLX                Branch and Move in dsn
LVL_FULL EQU   *
         SR    R1,R6                    Length of dataset
LVL_FULLX EQU  *
         BCTR  R1,Ø                     Decrease for execute
         EX    R1,MOVE                  Move in Userid
         LA    R1,1(R1)                 Recover Length
         AR    R4,R1                    Increment Pointer
         BR    R14                      Return
*-------------------------------------------------------------------*
* ADD_MEM_LIST - Add member to list                                  *
*-------------------------------------------------------------------*
ADD_MEM_LIST EQU *
         SLR   R15,R15
         LA    R3,MEM_LIST_STORE          Address of member list
         USING MEM_LIST,R3                Addressability
         IF CLC,MEM_LIST_CUR,EQ,=F'Ø'     Member list empty ?
            LR    R1,R3                   address of start of mbr list
            A     R1,=A(L'MEM_LIST_STORE)    R1 = address of length
            ST    R1,MEM_LIST_END            Save end of mem list
            LA    R1,MEM_LIST_MEMX           Address of member
            ST    R1,MEM_LIST_CUR            Current member address
         ENDIF
         L     R1,MEM_LIST_CUR            Current member address
         IF (C,1,GE,MEM_LIST_END)
            LA    R15,8
         ELSE
```

```
            MVI   Ø(R1),C' '              Space
            MVC   1(L'MEM_LIST_MEMX-1,R1),Ø(R1)   Move in space delim
            MVC   Ø(8,R1),ZLMEMBER        Move in member name
            LA    R1,L'MEM_LIST_MEMX(,R1) Point to addr of next member
            ST    R1,MEM_LIST_CUR         Current member address
         ENDIF
         BR 14                            Return
         DROP 3
*-------------------------------------------------------------------*
* CHK_UNIQUENESS - Check for member's existence in list             *
*-------------------------------------------------------------------*
CHK_UNIQUENESS EQU *
         LA    R3,MEM_LIST_STORE          Address of member list
         USING MEM_LIST,R3                Addressability
         SLR   R15,R15                    R15 = Ø
         IF (CLC,MEM_LIST_CUR,NE,=F'Ø')   IF not Ø, something in list
            LA    R1,MEM_LIST_MEMX        Address of member
            DO WHILE=(LTR,15,15,Z),UNTIL=(C,1,GE,MEM_LIST_CUR)
               IF CLC,Ø(8,1),EQ,ZLMEMBER  Does member already exist ?
                  LA    R15,8             yes ... RC = 8
               ENDIF
               DOEXIT LTR,15,15,NZ        Leave if RC NE Ø
               LA    R1,L'MEM_LIST_MEMX(,1)   Point to next member
            ENDDO
         ENDIF
         BR 14                            Return
         DROP 3
*-------------------------------------------------------------------*
* TRANSMIT FUNCTION -  Build the XMIT command and use the TSO       *
*                      Service facility to issue it.
*-------------------------------------------------------------------
TRANSMIT_FUNCTION EQU *
         STM   R14,R12,12(R13)          save caller's registers
         LA    R15,XSAVE_AREA1          push
         ST    R15,8(,R13)               down
         ST    R13,4(,R15)                save area
         LR    R13,R15                     stack
         CLI   M,C'N'                    Transmit all members ?
         BE    XMIT_BUILD_MEMBER_LIST    No ... Then build member list
         LA    R1,XMIT_FULL_DSN_LEN      Length for full dataset
         GETMAIN RU,LV=(1),LOC=BELOW     Get storage below 16MB
         LR    R8,R1                     Address of start of storage
         USING XMIT_CMD,8
         MVI   XMIT_CMD,C' '             Blank out command field
         MVC   XMIT_CMD+1(XMIT_PFX_LEN-1),XMIT_CMD
         LA    R5,XMIT_FULL_DSN_LEN      Length of command field
         ST    R5,XMIT_SIZE              Save it
         B     XMIT_BUILD_CMD            Go build XMIT command.
*-------------------------------------------------------------------*
*     Build the member list for the members that are to be XMITTED
```

```
*--------------------------------------------------------------------
XMIT_BUILD_MEMBER_LIST EQU *
         LA    R7,MEM_LIST_STORE        Address of member list
         USING MEM_LIST,R7              Addressability
         L     R1,MEM_LIST_CUR          Get current member
         LA    RØ,MEM_LIST_TEXT         address of members
         SR    R1,Ø
         LR    R1Ø,R1
         LA    R1,XMIT_PFX_LEN(,1)
         LR    R2,R1
         GETMAIN RU,LV=(1),LOC=BELOW    Getmain storage below 16MB
         LR    R8,R1                    Address of start of storage
         USING XMIT_CMD,R8
         MVI   XMIT_CMD,C' '            Blank out command field
         MVC   XMIT_CMD+1(XMIT_PFX_LEN-1),XMIT_CMD
         ST    R2,XMIT_SIZE             Save XMIT size
         LR    R2,R8
         LA    R2,4(,R2)
         LR    R3,R1Ø
         SLR   R5,R5                   Clear Storage to blanks
         ICM   5,8,=C' '
         MVCL  R2,R4
*--------------------------------------------------------------------*
*  Build the transmit command                                        *
*--------------------------------------------------------------------*
XMIT_BUILD_CMD EQU *
         MVC   XMIT_XMIT,=C'XMIT '     Move in XMIT to command
         LA    R1,XMIT_NODE_DOT_USERID        Index
         IF (CLC,NODE(8),NE,=CL8' ')    if blank node, assume nickname
            MVC   XMIT_NODE_DOT_USERID(8),NODE  Move in Node
            LA    2,1                      incr = 1
            LA    R3,8(,R1)                End of userid
            DO FROM=1,TO=2
               DOEXIT CLI,Ø(1),EQ,C' '
            ENDDO
            MVI   Ø(1),C'.'                Move in "."
         ENDIF
         MVC   1(8,R1),USERID          Move in USERID
         MVC   XMIT_DSN(4),=C' DA('    Move in DATASET keyword
*--------------------------------------------------------------------*
*  Determine length of dataset, and move it into the command         *
*--------------------------------------------------------------------*
         LA    R4,XMIT_DSN+4           Address of XMIT dataset field
         LA    R6,XMIT_DATASET         Address of dataset name
         XC    XMASK,XMASK             Clear Table
         MVI   XMASK+C' ',C' '         Set up Table
         TRT   Ø(44,R6),XMASK          Search for First blank
         BNZ   FULL                    Branch if Len NE Ø
         LA    R1,44                   Otherwise dsn is 44 chars
         B     FULLX                   Branch and Move in dsn
```

```
FULL     EQU   *
         SR    R1,R6                   Length of dataset
FULLX    EQU   *
         BCTR  R1,Ø                    Decrease for execute
         EX    R1,MOVE                 Move in Userid
         LA    R1,1(R1)                Recover Length
         AR    R4,R1                   Increment Pointer
         MVC   Ø(2,R4),=C') '          Move in closing bracket
         CLI   M,C'Y'                  Xmit all members ?
         BE    XMIT_ALL_MEMBERS        Yes ... then branch
*  Move in members into the command
         MVC XMIT_MEMTAG(9),=C' MEMBERS('  Move in members keyword
         LA    R2,XMIT_MEMBERS
         LR    R3,1Ø
         LA    R4,MEM_LIST_TEXT
         LR    R5,3
         MVCL  R2,R4
         AR    R2,R3
         IF (CLI,SEQ,EQ,C'Y')          Send sequentially ?
            MVC Ø(5,R2),=C') SEQ'      Move in Sequential keyword
            L   RØ,XMIT_SIZE
            A   RØ,=F'4'
            ST  RØ,XMIT_SIZE
         ELSE
            MVI Ø(R2),C')'             Move in close bracket
         ENDIF
*-------------------------------------------------------------------*
*   XMIT dataset                                                    *
*-------------------------------------------------------------------*
XMIT_ALL_MEMBERS EQU *
         L     15,CVTPTR               point to MVS's vector table
         L     15,CVTTVT-CVT(,15)      point to TSO's vector table
         L     15,TSVTASF-TSVT(,15)    point to TSO service facility
         CALL  (15),                                               X
               (=AL1(Ø,Ø,Ø,1),         flags                       X
               XMIT_CMD,               command text                X
               XMIT_SIZE,              length of command text      X
               XRETCODE,               XMIT return code            X
               XREASON,                TSO service routine reason code X
               XABEND),                TSO service routine abend code  X
               VL,MF=(E,XCALL)
*-------------------------------------------------------------------*
*   Free storage and leave transmit function                        *
*-------------------------------------------------------------------*
         IF (CLI,SEQ,EQ,C'Y')          Send sequentially ?
            L   RØ,XMIT_SIZE
            S   RØ,=F'4'
            ST  RØ,XMIT_SIZE
         ENDIF
         L     R1,XMIT_SIZE            get length of freemain
```

```
         FREEMAIN RU,LV=(1),A=(8)      Free storage
         L     R13,4(,R13)             pop save area stack
         RETURN (14,12),RC=(15)        return back to caller
         SLR 15,15                     RC = Ø
         BR 14                         Return to caller
         DROP 7
MOVE     MVC   Ø(Ø,R4),Ø(R6)
*-------------------------------------------------------------------*
* Static Data                                                       *
*-------------------------------------------------------------------*
@DSID    DC    CL8'DSID'
@DS1     DC    CL8'DS1'
@P4JTRAN DC    CL8'P4JTRANS'
@MEMLIST DC    CL8'P4JTRANM'
@ZLMEMBE DC    CL8'ZLMEMBER'
ANY      DC    CL8'ANY '
BROWSE   DC    CL8'BROWSE'
CONTROL  DC    CL8'CONTROL'
DISPLAY  DC    CL8'DISPLAY'
ERRORS   DC    CL8'ERRORS'
FREE     DC    CL8'FREE'
GET      DC    CL8'GET'
ISPZØØØ  DC    CL8'ISPZØØØ '
LMCLOSE  DC    CL8'LMCLOSE'
LMFREE   DC    CL8'LMFREE'
LMINIT   DC    CL8'LMINIT'
LMMDISP  DC    CL8'LMMDISP'
LMOPEN   DC    CL8'LMOPEN'
NAMELIST DC    C'(DSID ZLMEMBER ZLLCMD ZUSER PRJ1 LIB1 TYP1 MEM DS1 '
         DC    C'XVOL ZMSGØØØS ZMSGØØØL M SEQ NODE USERID '
         DC    C'ZCMD ZERRSM ZERRLM XMITZCMD)'
NAMES1   DC    C'(PRJ1 LIB1 TYP1 M SEQ NODE USERID)'
PROFILE  DC    CL8'PROFILE'
RESTORE  DC    CL8'RESTORE'
RETURN   DC    CL8'RETURN'
SAVE     DC    CL8'SAVE'
SETMSG   DC    CL8'SETMSG'
SHARED   DC    CL8'SHARED'
SHRW     DC    CL8'SHRW'
VDEFINE  DC    CL8'VDEFINE
VGET     DC    CL8'VGET'
VPUT     DC    CL8'VPUT'
VARLENS  DS    ØF
         DC    A(L'DSID)
         DC    A(L'ZLMEMBER)
         DC    A(L'ZLLCMD)
         DC    A(L'ZUSER)
         DC    A(L'PRJ1)
         DC    A(L'LIB1)
         DC    A(L'TYP1)
```

```
         DC    A(L'MEM)
         DC    A(L'DS1)
         DC    A(L'XVOL)
         DC    A(L'ZMSG000S)
         DC    A(L'ZMSG000L)
         DC    A(L'M)
         DC    A(L'SEQ)
         DC    A(L'NODE)
         DC    A(L'USERID)
         DC    A(L'ZCMD)
         DC    A(L'ZERRSM)
         DC    A(L'ZERRLM)
         DC    A(L'XMITZCMD)
NUMVARS  EQU   (*-VARLENS)/4
VARFMTS  DC    (NUMVARS)CL8'CHAR'
OPTIONS  DC    C'(NOBSCAN,LIST)'
CALLM    CALL  ,(,,,,,,,,,,,,,,,,,,),VL,MF=L
CALLML   EQU   *-CALLM
XCALL    CALL  ,(,,,,,,,,,,,,,,,,,,,,),VL,MF=L
XMITCMD  DS    CL255
*-------------------------------------------------------------------*
* Work Area Storage                                                 *
*-------------------------------------------------------------------*
WORK     DSECT
BLANKS   DS    CL80
BLANK_DATASET DS    CL44
SAVEAREA DS    18F
ISPLINKA DS    F
CALLL    CALL  ,(,,,,,,,,,,,,,,,,,),VL,MF=L
SRCHTAB  DS    XL256
MSGADDR  DS    A
TOP      DS    CL8
XMASK    DC    64XL1'00'                    TRT Table to test
         DC    XL1'40'                        Blank
         DC    191XL1'00'                   (End of User-id)
XRETCODE DS    A                            return code
XREASON  DS    A                            reason code
XABEND   DS    A                            abend code
XMIT_SIZE DS   F                            Size of this block
XMIT_DATASET   DS    CL56
XMIT_CMD_LEN   DS    A                      XMIT command length
XSAVE_AREA1 DS  18A                          register save area
XSAVE_AREA2 DS  18A                          register save area
*-------------------------------------------------------------------
* ISPF Variable Storage section of work area
*-------------------------------------------------------------------*
VARAREA  DS    0F
DSID     DS    CL8
ZLMEMBER DS    CL8
ZLLCMD   DS    C
```

```
ZUSER    DS    CL8
PRJ1     DS    CL8
LIB1     DS    CL8
TYP1     DS    CL8
MEM      DS    CL8
DS1      DS    CL56
XVOL     DS    CL6
ZMSG000S DS    CL24
ZMSG000L DS    CL80
M        DS    CL1
SEQ      DS    CL1
NODE     DS    CL8
USERID   DS    CL8
ZCMD     DS    CL133
ZERRSM   DS    CL24
ZERRLM   DS    CL80
XMITZCMD DS    CL133
MEM_LIST_STORE DS CL(MEM_LIST_PFX_LEN+(L'MEM_LIST_MEMX*MEM_LIST_SIZE))
WORKL    EQU   *-WORK
MEM_LIST DSECT
MEM_LIST_SIZE  EQU   200                  Member list size
MEM_LIST_CUR   DS    F                    Current member
MEM_LIST_END   DS    F                    End of member list
MEM_LIST_PFX_LEN EQU    *-MEM_LIST        Size of member list prefix
MEM_LIST_TEXT  EQU   *
MEM_LIST_MEMX  DS    CL9                  mem_name,
XMIT_CMD DSECT
XMIT_XMIT      DS    CL5                  "XMIT "
XMIT_NODE_DOT_USERID DS    CL17           'nnnnnnnn.useruser'
XMIT_DSN       DS    CL61                 " da(" dsn ")"
XMIT_FULL_DSN_LEN EQU *-XMIT_CMD         Length for Full dataset xmit
XMIT_MEMTAG    DS    CL9                  " MEMBERS("
XMIT_PFX_LEN   EQU   *-XMIT_CMD           Length of XMIT command
XMIT_MEMBERS   DS    C                    Members
         IKJTSVT
         CVT   DSECT=YES
         YREGS
         END
```

P4JTRANS

```
)attr
  * type(input) intens(non)
)body
%--------------------------  Transmit Dialog  --------------------------------
%Command
===>_zcmd                                                              +
%
%ISPF Library:
```

```
+   Project%===>_prj1    +
+   Group  %===>_lib1    +
+   Type   %===>_typ1    +
+   Member %===>_mem     +       (Blank for sequential dataset)
+
%Other partitioned or sequential dataset:
+   Dataset Name
%===>_ds1                                                    %
+   Volume Serial %===>_xvol  + (For uncatalogued datasets)                    %
+
%All Members (Y/N) ===>_m+    Y for all members, N for member list
+                             Y must be specified for sequential datasets
+
%Sequential (Y/N)  ===>_seq+  Y for send data sequentially
+                             N must be specified if all members transmitted
%Destination (Enter TSO NODES on command line for list of valid names)
+   Node   %===>_node    + Node to send output to
+   Userid %===>_userid  + Userid/Nickname to send output to
+
+
)init
.help = p4jtrant
&mem=' '
)proc
   if (&ds1 = ' ')
     ver (&prj1,nb)
     ver (&lib1,nb)
     ver (&typ1,nb)
   if (&ds1 *= ' ')
     &zfc = trunc(&ds1,1)
     if (&zfc = '''')
       &zrem = .trail
       &zrem2 = trunc(&zrem,'''')
       if (&zrem2 = &zrem)
         &ds1 = '&ds1&zfc'
     &zmcds1 = &ds1
   if (&m = 'Y')
      ver (&seq,nb,list,N)
   ver (&m,nb,list,Y,N)
   ver (&seq,nb,list,Y,N)
   if (&node = ' ')
      ver (&userid,nb,name)
   ver (&userid,nb,name)
)end
```

P4JTRANM

```
)ATTR
  _ TYPE(INPUT) CAPS(OFF) INTENS(HIGH)
```

```
  | AREA(DYNAMIC) SCROLL(ON) EXTEND(ON)
  + TYPE(TEXT) INTENS(LOW)
    TYPE(OUTPUT) INTENS(HIGH) JUST(RIGHT) PAD(Ø)
 Ø1 TYPE(DATAIN) INTENS(HIGH) CAPS(ON)
 Ø2 TYPE(DATAOUT) INTENS(LOW)
 Ø3 TYPE(DATAIN)  INTENS(HIGH) CAPS(ON)
 Ø4 TYPE(DATAOUT)  INTENS(LOW)
 Ø5 TYPE(DATAOUT)  INTENS(HIGH)
 Ø6 TYPE(DATAOUT)  INTENS(LOW)
)BODY EXPAND(//)
%MEMBER LIST -- &ZDSN /-/ %ROWZMLCR%OFZMLTR+
%COMMAND ===>_ZCMD                                              %SCROLL
===>_Z   +
|ZDATA
|
|
|
|
)INIT
  .ZVARS = 'ZSCML'       /* SCROLL AMT VARIABLE NAME  */
  .HELP = ISRØ1130Ø
)PROC
  VPUT (ZSCML) PROFILE
  &XMITZCMD = &ZCMD
)END
```

P4JTRANT

```
%TUTORIAL-------------------  Transmit Dialog  --------------------------------
%Command
===>_zcmd                                                              +
%
%ALL Members (Y/N) ===> +     Y for all members, N for member list
+
+   Enter Y to display member list.  S will select the member for XMIT and
+   B will browse the member.  A maximum of 1Ø members can be selected.
+
%Sequential (Y/N)  ===> +     Y for send data sequentially
+
+   Select Y to send the data as sequential output.  Do not send LMODs
+   as sequential output, as they will be NON-EXECUTABLE.  If sending
+   to a VM system, output MUST be sent sequentially.
+
%Destination
+   Node   %===> +               Node to send output to.
+
+   Enter the node that the output is to be sent to
+     (eg RBSDFA,RBSDFB,TORONTO,MONTREAL,VANCOUVR,etc)
```

```
+
+   Userid %===> +              Userid/Nickname to send output to.
+
)end
```

Paul Poolsaar
Systems Programmer (Canada)

Enqueue contention notification

INTRODUCTION

One function of multi-image integrity which is not implemented in GRS is the enqueue contention notification. This article describes a program (GRSMONSO) that allows a GRS user to implement such a function. It runs as a looping task that checks dataset contention at periodic intervals, and sends messages to TSO users asking them to free datasets that are causing contention.

To avoid swamping an inattentive TSO user with messages, a limit is set on the number of messages sent to a TSO user for a particular dataset. This limit can be changed by modifying the definition of symbol MAXMSGS and re-assembling.

The program has default limits on the number of dataset owners it can track, and the amount of information it can get back from GQSCAN. Installations that have a great deal of dataset contention may increase these limits by changing the definitions for symbols MAXOWNER and AREASIZE, and re-assembling. AREASIZE should be in the neighbourhood of 200 times MAXOWNER, but must be at least 296 bytes in total (as required by GQSCAN).

The program can be stopped using the stop command: P GRSMONSO.

The dynamic storage used by this program must be obtained from below the 16-megabyte line in order to satisfy the requirements of the TPUT macro.

This program will not notify users on other systems of contention. However, the same effect may be achieved by running a a copy of this program on every system.

GRSMONSO SOURCE CODE

```
GRSMONSO CSECT
GRSMONSO AMODE  31
GRSMONSO RMODE  24
*********************************************************************
*   CONSTANTS FOR INSTALLATION TWEAKING                             *
*********************************************************************
MAXMSGS  EQU    5                    MAXIMUM NUMBER OF MESSAGES TO
*                                    SEND TO A TSO USER WHO OWNS A
*                                    DATASET
MAXOWNER EQU    50                   MAXIMUM NUMBER OF DATASETS
*                                    OWNERS THAT CAN BE TRACKED
AREASIZE EQU    10000                SIZE OF AREA FOR RIBS AND RIBES
*                                    RETURNED BY GQSCAN
*********************************************************************
*   STANDARD ENTRY LINKAGE                                          *
*********************************************************************
         STM    R14,R12,12(R13)
         BALR   BASEPTR,0
         USING  *,BASEPTR
         MODID  BR=YES
         LR     R2,R1                 SAVE INPUT PARAMETER
         LR     R3,R13                SAVE CALLER'S SAVEAREA ADDRESS
         L      R0,DYNASIZE           GET AMOUNT OF STORAGE NEEDED
         GETMAIN RU,LV=(R0),LOC=(BELOW,ANY) OBTAIN DYNAMIC STORAGE
         LR     DATAPTR,R1
         LA     DATAPTR2,4095(,R1)
         USING  DYNA,DATAPTR
         USING  DYNA+4095,DATAPTR2
         ST     R3,SAVEAREA+4         SAVE # OF CALLER'S SAVEAREA
         ST     R1,8(,R3)             CHAIN OUR SAVEAREA TO CALLERS
         LTR    R2,R2                 TEST FOR PARAMETERS
         BE     DEFAULT               NONE, TAKE THE DEFAULT
         L      R1,0(,R2)             ADDRESS PARAMETER AREA
         USING  PARMAREA,R1
         LH     R3,PARMLEN            GET LENGTH OF PARAMETERS
         LTR    R3,R3                 TEST FOR PARAMETERS
         BNE    GETPARM
DEFAULT  MVC    INTERVAL,=F'6000'     NO PARAMETER WAS SPECIFIED,
         MVC    INT,=CL3'60'
*                                     DEFAULT TO 60 SECONDS
         B      GO
```

```
*********************************************************************
*   CONVERT INPUT PARAMETER INTO .Ø1 SECOND UNITS FOR STIMER        *
*********************************************************************
GETPARM  EQU    *                       PARAMETER WAS SPECIFIED
         MVC    INT,PARM
         MVC    PARMBUF,=C'ØØØØ'
         LA     R2,L'PARMBUF
         SLR    R2,R3
         LA     R2,PARMBUF(R2)
         EX     R3,COPYPARM             RIGHT JUSTIFY PARAMETER
         PACK   PACKAREA,PARMBUF        EBCDIC -> DECIMAL
         CVB    R4,PACKAREA             DECIMAL -> BINARY
         MH     R4,=H'1ØØ'              CONVERT TO .Ø1 SECOND UNITS
         ST     R4,INTERVAL             SAVE FOR STIMER
         DROP   R1
*********************************************************************
*   INITIALIZE THE CURRENT LIST TO EMPTY                            *
*********************************************************************
GO       EQU    *
         EXTRACT ANSAREA,               EXTRACT THE COMMUNICATION AREA  X
               FIELDS=COMM,                                             X
               MF=(E,EXTRACTL)
         L      R3,ANSAREA
         USING  COM,R3
         QEDIT  ORIGIN=COMCIBPT,CIBCTR=255  SET LIMIT COUNT
*********************************************************************
*   INITIALIZE THE CURRENT LIST TO EMPTY                            *
*********************************************************************
INIT     EQU    *
         SR     CURCOUNT,CURCOUNT       CURRENT LIST STARTS EMPTY
LOOP     EQU    *
*********************************************************************
*   CHECK FOR DATASET CONTENTION VIA GQSCAN                         *
*   LIMIT SCAN TO                                                   *
*         RESNAME=SYSDSN      TO GET ONLY DATASET ENQS.             *
*         WAITCNT=1           TO GET ONLY RESOURCES WITH CONTENTION.*
*         REQLIM=2            TO GET INFORMATION ON ONLY THE FIRST  *
*                               TWO REQUESTORS FOR A DATASET, SINCE *
*                               THIS PROGRAM DOES NOT WORRY ABOUT   *
*                               OTHER WAITING JOBS.                 *
*   WHEN GQSCAN RETURNS:                                            *
*         RØ CONTAINS SIZE VALUES FOR THE RIB AND RIBE              *
*         R1 CONTAINS THE NUMBER OF RIBS RETURNED IN SCANAREA       *
*         R15 CONTAINS A RETURN CODE.                               *
*********************************************************************
         GQSCAN AREA=(SCANAREA,AREASIZE),SCOPE=ALL,RESNAME=QNAME,      X
               WAITCNT=1,REQLIM=2,MF=(E,SCANLIST)
         C      R15,=F'8'               GOOD RETURN CODE (<=8)
         BH     ERROR                   NO, SOME UNEXPECTED ERROR, DO
*                                       NOT PROCESS ANY DATA.
```

```
*************************************************************************
*   COPY CURRENT LIST (POSSIBLY EMPTY) INTO PREVIOUS LIST AND           *
*   CLEAR CURRENT LIST.                                                 *
*************************************************************************
         LA      R2,CURRENT              AREA TO COPY FROM
         LA      R4,PREVIOUS             AREA TO COPY TO
         LA      R3,LENCUR               LENGTH OF DATA TO MOVE
         LR      R5,R3
         MVCL    R4,R2                   COPY CURRENT -> PREVIOUS
         LR      PRECOUNT,CURCOUNT       SET SIZE OF PREVIOUS LIST
         SR      CURCOUNT,CURCOUNT       CLEAR CURRENT LIST BY SETTING
*                                        THE NUMBER OF ENTRIES TO ZERO
*************************************************************************
*   SCAN THROUGH THE RIBS (POSSIBLY NONE) THAT WERE RETURNED.           *
*************************************************************************
         LTR     R1,R1                   TEST NUMBER OF RIBS RETURNED
*                                        BY GQSCAN
         BE      CHKPREV                 IF NONE WERE RETURNED, THEN
*                                        GO CHECK IF ANY CONTENTION
*                                        HAS BEEN RELIEVED
         ST      RØ,SIZES                SAVE SIZE OF RIB AND RIBES
         LA      RIBPTR,SCANAREA         GET ADDRESS OF FIRST RIB
         USING   RIB,RIBPTR
*************************************************************************
*   CHECK RIB TO SEE IF IT MATCHES THE SEARCH CRITERIA.                 *
*************************************************************************
CHECKRIB EQU     *
         ST      R1,RIBSLEFT             SAVE REMAINING NUMBER OF RIBS
         LR      RIBVPTR,RIBPTR
         AH      RIBVPTR,LENRIB          COMPUTE ADDRESS OF RIBVAR
         USING   RIBVAR,RIBVPTR
         LR      RIBEPTR,RIBVPTR
         AH      RIBEPTR,RIBVLEN         GET ADDRESS OF FIRST RIBE
         LR      R2,RIBEPTR              GET ADDRESS OF FIRST RIBE
         AH      R2,LENRIBE              COMPUTE ADDRESS OF SECOND RIBE
         TM      RIBESFLG-RIBE(R2),RIBESTAT  IS THE SECOND REQUESTOR
*                                        WAITING FOR THE DATASET?
         BNZ     NEXTRIB                 NO, THERE ARE MULTIPLE OWNERS
*                                        SHARING THE DATASET, SKIP THIS
*                                        RIB
         USING   RIBE,RIBEPTR
         L       R2,CVTPTR               FIND THE CVT
         CLC     CVTSNAME-CVTMAP(L'RIBESYSN,R2),RIBESYSN IS THE DATASET
*                                        OWNER FROM THIS SYSTEM ?
         BNE     NEXTRIB                 NO, UNABLE TO NOTIFY USERS
*                                        ON OTHER SYSTEMS, SKIP THIS RIB
*************************************************************************
*   RESOURCE & JOB WERE FOUND THAT MATCH THE CRITERIA. ADD TO THE       *
*   CURRENT LIST IF THERE IS AN AVAILABLE ENTRY.                        *
*************************************************************************
```

```
         CL     CURCOUNT,=A(MAXOWNER)  IS THE CURRENT LIST FULL?
         BNL    NEXTRIB                YES, MUST SKIP THIS RIB
         LA     R3,LENOWNER            GET SIZE OF AN ENTRY
         MR     R2,CURCOUNT            MULTIPLY (USING REGISTER PAIR)
*                                       TO GET RELATIVE OFFSET WITHIN
*                                       LIST OF FIRST FREE ENTRY
         LA     R2,CURRENT(R3)         COMPUTE ADDRESS OF FIRST FREE
*                                       ENTRY IN CURRENT LIST
         LA     CURCOUNT,1(CURCOUNT)   GET INDEX OF FREE ENTRY
         USING  OWNER,R2               ADDRESS ENTRY IN CURRENT LIST
         XC     OWNER(LENOWNER),OWNER  INITIALIZE ENTRY TO ZEROS
         MVC    JOBNAME,RIBEJBNM       STORE NAME OF DATASET OWNER
         MVC    JOBASID,RIBEASID       STORE ASID OF DATASET OWNER
         SR     R4,R4
         IC     R4,RIBRNMLN            GET LENGTH OF DATASET NAME
         LA     RØ,L'DSNAME            GET MAXIMUM LENGTH IN OUTPUT
         CR     R4,RØ                  NAME EXCEEDS OUTPUT LENGTH?
         BL     NOTRIM
         LR     R4,RØ                  YES, TRIM LENGTH FOR OUTPUT
NOTRIM   STH    R4,LENDSN              SAVE LENGTH OF DATASET NAME
         BCTR   R4,Ø                   ADJUST LENGTH FOR MVC
         EX     R4,MOVEDSN             STORE DATSET NAME
*************************************************************************
*   SEARCH THE PREVIOUS LIST FOR AN ENTRY WITH JOB NAME, ASID, AND    *
*   DATASET NAME THAT MATCHES THE NEW ENTRY IN THE CURRENT LIST.      *
*************************************************************************
         LA     R3,1
         LA     R15,PREVIOUS           FIND START OF PREVIOUS LIST
PREVLOOP CR     R3,PRECOUNT            HAS THE ENTIRE PREVIOUS LIST
*                                       BEEN SEARCHED?
         BH     NEXTRIB                FINISHED EXAMINING THE PREVIOUS
*                                       LIST, NO MATCH FOUND. SKIP TO
*                                       THE NEXT RIB.
COMPARE  EQU    *                      COMPARE THE ENTRY FOUND IN THE
*                                       PREVIOUS LIST WITH THE NEW
*                                       ENTRY IN THE CURRENT LIST
         CLC    KEY-OWNER(LENKEY,R15),KEY  DO THE KEYS (JOB NAME, ASID
*                                       AND DATASET NAME) FOR THE TWO
*                                       ENTRIES MATCH?
         BE     MATCH                  YES, MATCH FOUND
NEXTOWN  EQU    *                      NO MATCH YET, LOCATE NEXT ENTRY
*                                       IN PREVIOUS LIST
         LA     R3,1(R3)               GET INDEX OF NEXT ENTRY
         LA     R15,LENOWNER(R15)      GET ADDRESS OF NEXT ENTRY
         B      PREVLOOP
*************************************************************************
*   MATCHING ENTRY WAS FOUND. COPY THE DATA FROM THE PREVIOUS ENTRY   *
*   INTO THE CURRENT ENTRY AND CALL CONTENTIONEXISTS.                 *
*     R2 -> ENTRY IN CURRENT LIST                                     *
*     R15-> ENTRY IN PREVIOUS LIST                                    *
*************************************************************************
```

```
MATCH    EQU    *                       DATASET OWNER IN PREVIOUS
*                                       LIST MATCHED CURRENT ENTRY
         MVC    DATA(LENDATA),DATA-OWNER(R15)  SAVE DATA FROM PREVIOUS
*                                       LIST INTO CURRENT LIST
         OI     STATUS-OWNER(R15),ONGOING INDICATE IN THE PREVIOUS LIST
*                                       THAT CONTENTION STILL EXISTS
*                                       FOR THIS ENTRY.
         LR     R1,R2                   POINT AT ENTRY IN CURRENT LIST
         DROP   R2
         BAL    14,CONTENTIONEXISTS
*************************************************************************
*   FINISHED PROCESSING THIS RIB, GO ON TO THE NEXT ONE.                *
*************************************************************************
NEXTRIB  EQU    *
         L      R2,RIBNRIBE             GET NUMBER OF RIBES FOR THIS RIB
         MH     R2,LENRIBE              COMPUTE TOTAL SIZE OF THE RIBES
         ALR    RIBEPTR,R2              COMPUTE ADDRESS OF NEXT RIB
         LR     RIBPTR,RIBEPTR          SAVE ADDRESS OF NEXT RIB
         L      R1,RIBSLEFT             GET NUMBER OF RIBS LEFT
         BCT    R1,CHECKRIB             PROCESS NEXT RIB, IF ANY
*************************************************************************
*   FINISHED PROCESSING ALL RIBS RETURNED BY GQSCAN.                    *
*   SEARCH THROUGH THE PREVIOUS LIST TO FIND ENTRIES FOR DATASETS       *
*   THAT WERE NOT FOUND ON THE CURRENT SCAN.                            *
*************************************************************************
CHKPREV  EQU    *
         USING  OWNER,R1                ADDRESS ENTRIES IN PREVIOUS LIST
         LA     R3,1
         LA     R1,PREVIOUS             FIND START OF PREVIOUS LIST
CHKLOOP  CR     R3,PRECOUNT             HAS THE ENTIRE PREVIOUS LIST
*                                       BEEN SEARCHED?
         BH     WAIT                    FINISHED EXAMINING THE PREVIOUS
*                                       LIST. PAUSE BEFORE NEXT GQSCAN
         TM     STATUS,ONGOING          DOES THIS DATASET STILL HAVE
*                                       CONTENTION?
         BNZ    NEXTENT                 YES, DO NOT CALL ROUTINE
         BAL    R14,CONTENTIONRELIEVED
NEXTENT  EQU    *                       FIND NEXT ENTRY IN PREVIOUS LIST
         LA     R3,1(R3)                GET INDEX OF NEXT ENTRY
         LA     R1,LENOWNER(R1)         GET ADDRESS OF NEXT ENTRY
         B      CHKLOOP
         DROP   R1
*************************************************************************
*   ERROR, GQSCAN WAS UNSUCCESSFUL. ASSUME THAT THIS IS A TEMPORARY     *
*   ERROR AND TAKE NO ACTION.                                           *
*************************************************************************
ERROR    EQU    *
*************************************************************************
*   ALL RIBS RETURNED BY GQSCAN HAVE BEEN EXAMINED.                     *
*   CONTENTIONEXISTS WAS CALLED FOR EACH DATASET THAT IS STILL BEING    *
```

```
*     CONTENDED FOR.                                                *
*   CONTENTIONRELIEVED WAS CALLED FOR EACH DATASET FOR WHICH THE    *
*     CONTENTION CLEARED UP DURING THE LAST INTERVAL.               *
*   PAUSE BEFORE REPEATING THE PROCESS.                             *
*********************************************************************
WAIT     EQU    *
         STIMER WAIT,BINTVL=INTERVAL   PAUSE BEFORE NEXT SCAN
         L      R3,ANSAREA             RESTORE ADDRESS OF COM AREA
         USING  COM,R3
         L      R5,COMECBPT            GET ECB ADDRESS
         TM     Ø(R5),X'4Ø'            ECB POSTED ?
         BNO    LOOP
         L      R4,COMCIBPT            GET CIB ADDRESS
         USING  CIB,R4
         CLI    CIBVERB,CIBSTOP        STOP COMMAND ?
         BE     COMPLETE
         CLI    CIBVERB,CIBMODFY       MODIFY COMMAND ?
         BNE    FREE
         LH     R5,CIBDATLN            DATA LENGTH
         LTR    R5,R5                  ZERO ?
         BZ     FREE                   NO DATA
         BCTR   R5,RØ
         MVC    COMMAND,=CL8Ø' '       CLEAR
         EX     R5,DATAMOVE
         CLC    COMMAND(Ø7),=CLØ7'DISPLAY'
         BE     DISPLAY
         MVC    WTO(WTOL),WTOC
         MVC    WTO+Ø4(8Ø),=CL8Ø'GRSMONSO: INVALID COMMAND - 12345678'
         MVC    WTO+32(Ø8),COMMAND
         WTO    MF=(E,WTO)
         B      FREE
DISPLAY  EQU    *
         MVC    WTO(WTOL),WTOC
         MVC    WTO+Ø4(8Ø),=CL8Ø'GRSMONSO: MONITORING INTERVAL = 123 S'
         MVC    WTO+36(Ø3),INT
         WTO    MF=(E,WTO)
FREE     EQU    *
         QEDIT  ORIGIN=COMCIBPT,BLOCK=(R4) FREE CIB
         B      LOOP
*********************************************************************
*   RETURN TO THE CALLER                                            *
*********************************************************************
COMPLETE EQU    *
         LR     R2,R13
         L      R13,SAVEAREA+4
         L      RØ,DYNASIZE
         FREEMAIN RU,A=(R2),LV=(RØ)
         LM     R14,R12,12(R13)
         SR     R15,R15
         BR     R14
```

```
*************************************************************************
*   SUBROUTINE: CONTENTIONEXISTS                                        *
*   FUNCTION  : THIS ROUTINE IS CALLED WHENEVER CONTENTION IS           *
*               DETECTED ON A DATASET, AND THE DATASET OWNER IS A       *
*               JOB ON THIS SYSTEM. IT WILL TAKE WHATEVER ACTION IT     *
*               CAN TO ATTEMPT TO RELIEVE THE CONTENTION.               *
*   OPERATION : IF THE OWNER IS A TSO USER, CALL SENDMSG.               *
*   INPUT     : R1 POINTS TO AN OWNER BLOCK WHICH DESCRIBES THE         *
*               DATASET BEING CONTENDED FOR AND ITS OWNER.              *
*   NOTE      : THIS ROUTINE MAY READ INFORMATION IN THE KEY SECTION    *
*               OF THE OWNER BLOCK, AND MAY READ AND WRITE THE          *
*               DATA SECTION. THE STATUS SECTION SHOULD BE IGNORED.     *
*************************************************************************
CONTENTIONEXISTS EQU *
         STM    R14,R12,SAVE1          SAVE CALLER'S REGS
         LR     R2,R1                  GET ADDRESS OF ENTRY
         USING  OWNER,R2
         LOCASCB ASID=JOBASID          GET ASCB OF OWNER
         LTR    R15,R15                ASCB WAS FOUND ?
         BNE    CEEND
         L      R1,ASCBOUCB-ASCB(,R1)  YES, GET OUCB OF OWNER
         TM     OUCBYFL-OUCB(R1),OUCBLOG LOGON CREATED USER?
         BZ     NOTTSO                 NO, NOT A TSO USER
         LR     R1,R2                  RESTORE R1 FOR SENDMSG
         BAL    R14,SENDMSG            YES, SEND MESSAGE TO TSO USER
         B      CEEND
NOTTSO   EQU    *                      OWNER IS NOT A TSO USER,
*                                       TAKE NO ACTION
CEEND    EQU    *                      END OF CONTENTIONEXISTS
         LM     R14,R12,SAVE1          RESTORE CALLER'S REGS
         BR     R14                    RETURN TO CALLER
         DROP   R2
*************************************************************************
*   SUBROUTINE: CONTENTIONRELIEVED                                      *
*   FUNCTION  : THIS ROUTINE IS CALLED WHENEVER CONTENTION ON A         *
*               DATASET IS FOUND TO HAVE BEEN RELIEVED. IT SHOULD       *
*               CLEAN UP ANY RESOURCES THAT WERE BEING USED WHILE       *
*               TRACKING THAT CONTENTION.                               *
*   OPERATION : NOTHING, NO RESOURCES NEED TO BE CLEANED UP.            *
*   INPUT     : R1 POINTS TO AN OWNER BLOCK WHICH DESCRIBES THE         *
*               DATASET THAT WAS BEING CONTENDED FOR AND ITS            *
*               LAST OWNER.                                             *
*   NOTE      : THIS ROUTINE MAY READ INFORMATION IN THE KEY AND        *
*               DATA SECTIONS OF THE OWNER BLOCK. THE STATUS SECTION    *
*               SHOULD BE IGNORED.                                      *
*************************************************************************
CONTENTIONRELIEVED EQU *
         STM    R14,R12,SAVE1          SAVE CALLER'S REGS
         USING  OWNER,R1
CREND    EQU    *                      END OF CONTENTIONRELIEVED
```

```
         LM     R14,R12,SAVE1          RESTORE CALLER'S REGS
         BR     R14                    RETURN TO CALLER
         DROP   R1
*************************************************************************
*   SUBROUTINE: SENDMSG                                                 *
*   FUNCTION  : SEND A CONTENTION MESSAGE TO A TSO USER.                *
*   OPERATION : IF THE MESSAGE LIMIT HAS NOT BEEN EXCEEDED:             *
*                 BUILD THE MESSAGE TEXT IN THE MSGBUF BUFFER.          *
*                 SEND THE MESSAGE VIA TPUT.                            *
*   INPUT     : R1 POINTS TO AN OWNER BLOCK WHICH DESCRIBES THE         *
*               DATASET AND THE TSO USER THAT OWNS IT.                  *
*************************************************************************
SENDMSG  EQU    *
         STM    R14,R12,SAVE2          SAVE CALLER'S REGS
         USING  OWNER,R1
         L      R3,MSGCOUNT
         C      R3,=A(MAXMSGS)         ALREADY SENT ENOUGH MESSAGES?
         BNL    NOSEND                 YES, DON'T SEND ANY MORE
         LA     R3,1(R3)
         ST     R3,MSGCOUNT            ONE MORE MESSAGE BEING SENT
         MVI    MSGBUF,C' '
         MVC    MSGBUF+1(L'MSGBUF-1),MSGBUF CLEAR MESSAGE BUFFER
         MVC    MSGBUF(L'MSGSTART),MSGSTART COPY BEGINNING OF MESSAGE
         LH     R4,LENDSN              GET LENGTH OF DATASET NAME
         BCTR   R4,Ø                   ADJUST LENGTH FOR MVC
         EX     R4,MOVEMSG             COPY DATASET NAME TO BUFFER
         LA     R2,1+MSGBUF+L'MSGSTART(R4) FIND ADDRESS IN THE MESSAGE
*                                      BUFFER FOLLOWING THE DATASET
*                                      NAME
         MVC    Ø(L'MSGEND,R2),MSGEND  COPY END OF MESSAGE TO BUFFER
         LA     R3,MSGBUF
         LA     R4,L'MSGBUF
         LA     R5,JOBNAME
         TPUT   (R3),(R4),USERIDL=(R5) NOTIFY USER OF CONTENTION
NOSEND   EQU    *
         LM     R14,R12,SAVE2          RESTORE CALLER'S REGS
         BR     R14                    RETURN TO CALLER
         DROP   R1
*************************************************************************
*   TARGETS OF EX INSTRUCTIONS                                          *
*************************************************************************
DATAMOVE MVC    COMMAND(Ø),CIBDATA
         USING  PARMAREA,R1
COPYPARM MVC    Ø(Ø,R2),PARM           USED FOR RIGHT-JUSTIFYING INPUT
         DROP   R1
         USING  OWNER,R2
MOVEDSN  MVC    DSNAME(Ø),RIBRNAME     USED TO COPY DATASET NAME INTO
*                                      ENTRY IN CURRENT LIST
         DROP   R2
         USING  OWNER,R1
```

```
MOVEMSG  MVC    MSGBUF+L'MSGSTART(Ø),DSNAME USED TO PUT DATASET NAME
*                                    INTO THE MESSAGE BUFFER
         DROP   R1
*************************************************************************
*   MACRO LIST FORMS                                                    *
*************************************************************************
EXTRACTL EXTRACT MF=L
WTOC     WTO    '                                                      X
                                                  ',MF=L,ROUTCDE=(11)
WTOL     EQU   *-WTOC                LENGTH OF MACRO EXPANSION
*************************************************************************
*   REGISTER DECLARES                                                   *
*************************************************************************
RIBVPTR  EQU    5                    ADDRESS OF RIBVAR SECTION
RIBPTR   EQU    6                    ADDRESS OF RIB
RIBEPTR  EQU    7                    ADDRESS OF RIBE
PRECOUNT EQU    8                    # OF ENTRIES IN PREVIOUS LIST
CURCOUNT EQU    9                    # OF ENTRIES IN CURRENT LIST
R1Ø      EQU    1Ø                   RESERVED FOR FUTURE EXPANSION
*                                    OF THE CODE OR THE DYNAMIC AREA
DATAPTR2 EQU    11                   SECOND DATA REGISTER
BASEPTR  EQU    12                   CODE REGISTER
DATAPTR  EQU    13                   FIRST DATA REGISTER
*************************************************************************
*   STATIC DATA                                                         *
*************************************************************************
         DS     ØF
DYNASIZE DC     AL4(LENDYNA)         AMOUNT OF DYNAMIC STORAGE NEEDED
QNAME    DC     CL8'SYSDSN  '        MAJOR NAME FOR DATASET ENQS
*  MESSAGE TO SEND TO TSO USERS
MSGSTART DC     C'PLEASE FREE '''
MSGEND   DC     C'''. OTHER JOBS ARE WAITING TO USE IT. (GRSMONSO)'
*************************************************************************
*   DYNAMIC DATA                                                        *
*************************************************************************
DYNA     DSECT
SAVEAREA DS     18F                  STANDARD SAVEAREA
SAVE1    DS     15F                  FIRST LEVEL SUBROUTINE SAVEAREA
SAVE2    DS     15F                  SECOND LEVEL SUBROUTINE SAVEAREA
ANSAREA  DS     F                    AREA FOR EXTRACT MACRO
INTERVAL DS     F                    TIME TO PAUSE BETWEEN SCANS
PACKAREA DS     D                    INTERVAL IN PACKED DECIMAL FORM
PARMBUF  DS     CL4                  RIGHT-JUSTIFIED PARAMETER
COMMAND  DS     CL8Ø
INT      DS     CL3
WTO      DS     CL(WTOL)
SIZES    DS     F
         ORG    SIZES
LENRIB   DS     H                    RETURNED SIZE OF RIB
LENRIBE  DS     H                    RETURNED SIZE OF RIBE
```

```
RIBSLEFT DS     F                       NUMBER OF RIBS LEFT TO PROCESS
MSGBUF   DS     CL120                   BUFFER TO BUILD MESSAGE IN
SCANLIST GQSCAN MF=L
*  TABLE OF DATASET OWNERS FROM THE CURRENT SCAN.
CURRENT  DS     0F
         ORG    CURRENT+MAXOWNER*LENOWNER
LENCUR   EQU    *-CURRENT               SIZE OF CURRENT AREA
*  TABLE OF DATASET OWNERS FROM THE PREVIOUS SCAN.
PREVIOUS DS     0F
         ORG    PREVIOUS+MAXOWNER*LENOWNER
*  AREA FOR RESOURCE DATA RETURNED BY GQSCAN.
SCANAREA DS     0F                      AREA FOR GQSCAN DATA
         ORG    SCANAREA+AREASIZE
LENDYNA  EQU    *-DYNA                  TOTAL SIZE OF DYNAMIC STORAGE
*************************************************************************
*   DATASET OWNER ENTRY MAPPING                                         *
*************************************************************************
OWNER    DSECT
* KEY THAT UNIQUELY IDENTIFIES AN OWNER ENTRY
KEY      DS     0C
JOBNAME  DS     CL8                     USER OWNING DATASET RESOURCE
JOBASID  DS     H                       ADDRESS SPACE OF USER
LENDSN   DS     H                       TRUE LENGTH OF DATASET NAME
DSNAME   DS     CL44                    BUFFER FOR DATASET NAME
LENKEY   EQU    *-KEY                   TOTAL SIZE OF THE ENTRY KEY
* DATA FOR USE BY THE CONTENTIONEXISTS AND CONTENTIONRELIEVED ROUTINES
DATA     DS     0C
MSGCOUNT DS     F                       NUMBER OF MESSAGES THAT HAVE
*                                       BEEN SENT TO THIS USER FOR
*                                       THIS DATASET NAME.
LENDATA  EQU    *-DATA                  TOTAL SIZE OF THE ENTRY DATA
* DATA FOR USE BY THE MAINLINE CODE
STATUS   DS     BL1
ONGOING  EQU    X'80'                   ENTRY HAS A CURRENT MATCH
         DS     0F                      ALIGN TO FULLWORD BOUNDARY
LENOWNER EQU    *-OWNER                 TOTAL SIZE OF THE ENTRY
*************************************************************************
*   INPUT PARAMETER MAPPING                                             *
*************************************************************************
PARMAREA DSECT
PARMLEN  DS     H
PARM     DS     CL3                     INTERVAL IN SECONDS
*************************************************************************
*   MAPPING MACROS                                                      *
*************************************************************************
         IHAASCB
         CVT    DSECT=YES
         IRAOUCB
         ISGRIB
COM      DSECT
```

```
        IEZCOM
CIB     DSECT
        IEZCIB
        END
```

SAMPLE SYS1.PROCLIB MEMBER (GRSMONSO)

```
//grsmonso proc interval=60
//grsmonso exec pgm=grsmonso,parm='&interval'
//*
//*  invoke from console with
//*       start grsmonso
//*  to have the program scan for contention
//*  every 60 seconds.
//*
//*  optionally, invoke from console with
//*       start grsmonso,interval=ssss
//*  where ssss is the number of seconds (in decimal)
//*  to pause between scans.
//*
```

Patrick Renard
CTRNE (France)

DFSMS data collection facility

The DCOLLECT facility is a function of the access method services system and can be used to collect data in a sequential file for use as input to other programs or utilities. The facility can get you information on the following:

- Active datasets
- VSAM dataset information
- Volumes
- Inactive data
- Capacity planning
- SMS configuration information.

ACTIVE DATASETS

DCOLLECT will produce information about space usage, dataset attributes, and indicators on selected volumes and storage groups.

VSAM DATASET INFORMATION

Specific information relating to VSAM datasets can be produced for the specified volumes and storage groups.

VOLUMES

DCOLLECT will produce volume statistics.

INACTIVE DATA

Data on migrated and backed-up datasets regarding space information and dataset attributes can be collected.

CAPACITY PLANNING

DASD and tape data can be collected.

SMS CONFIGURATION INFORMATION

The information about SMS configurations can be from the active control datasets or the source control datasets or from an active configuration. Information can include:

- Data class constructs
- Storage class constructs
- Management class constructs
- Storage group constructs
- SMS volume information
- SMS base configuration information
- Aggregate group construct information
- Optical drive information
- Optical library information
- Cache names
- Accounting information for the ACS.

EXECUTION

The syntax of the DCOLLECT command is:

```
DCOLLECT OUTFILE(DDNAME)
         OUTDATASET(DSNAME)
```

Optional parameters are:

```
         VOLUMES(VOLSER...)
         STORAGEGROUP(STGNAME)
         MIGRATEDATA
         BACKUPDATA
         CAPPLANDATA
         SMSDATA(SCDSNAME(ENTRYNAME)|ACTIVE
         REPLACE|APPEND
         NODATAINFO
         NOVOLUMEINFO
         ERRORLIMIT(VALUE)
         EXITNAME(ENTRYPOINT)
         MIGRSNAPALL|MIGRSNAPERR
```

Although the BACKUPDATA, CAPPLANDATA, MIGRATEDATA, STORAGEGROUP, VOLUMES, and SMSDATA are optional, at least one must be coded. The command must be APF-authorized so if called from a user program it must have the correct APF attributes. RACF also provides a profile named STGADMIN.IDC.DCOLLECT and the user executing the command must have READ authority or the request will be refused.

For full details on the meanings of the various syntax keywords you should refer to the *DFSMS/MVS 1.3 Access Method Services for ICF* manual. The keywords are discussed in detail in that manual.

Below is an example of the JCL to collect data for any volumes that begin with the volume serial number SYSB. All data is written to the file referred to in the OUTDS DD statement.

```
//STS01A   JOB  (SDTS),CLASS=A,MSGCLASS=Q,MSGLEVEL=(1,1)
//STEP1    EXEC PGM=IDCAMS
//SYSPRINT DD   SYSOUT=*
//OUTDS    DD   DSN=STS01.BD.DATA.DCOLLECT,
//         DISP=(NEW,CATLG,DELETE),
//         DSORG=PS,
//         RECFM=VB,
//         LRECL=644,
//         BLKSIZE=0,
//         SPACE=(CYL,(100,10),RLSE),
```

```
//         UNIT=WORK
//SYSIN    DD    *
      DCOLLECT              -
           OFILE(OUTDS)     -
           VOLUME(SYSB*)
/*
```

The print output produced by the IDCAMS program for this job looks something like this:

```
IDCAMS SYSTEM SERVICES

     DCOLLECT  -
       OFILE(OUTDS) -
       VOLUME(SYSB*)

     IDC01811I NUMBER OF 'D ' RECORDS PROCESSED WAS 2002
     IDC01811I NUMBER OF 'A' RECORDS PROCESSED WAS 181
     IDC01811I NUMBER OF 'V ' RECORDS PROCESSED WAS 11
     IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

     IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE 0
```

The output file record format differs depending on the output requested. Full details on the format and meaning of all the fields are contained in Appendix G of the *DFSMS/MVS V1R3 Access Method Services* manual. The records produced are identified by a record type. The different record types are:

D	Active dataset record
A	VSAM association information
V	Volume information
M	Migrated dataset information
B	Back-up dataset information
C	DASD capacity planning information
T	Tape capacity planning information
DC	Data class construct information
SC	Storage class construct information
MC	Management class information
BC	Base configuration information
SG	Storage group construct information
VL	Storage group volume information
AG	Aggregate group information
DR	OAM drive record information

LB OAM library record information
CN Cache names from the base configuration information
AI Accounting information from the ACS.

SPACE CALCULATION FORMULAE

The dataset created by DCOLLECT is partitioned sequential and has a variable blocked record format. It is possible to estimate the size required for the space parameter for each type of request issued. The space calculation formulae are shown below.

Volume list
space = record size (336+4) * average number of datasets on volume * number of volumes scanned

Storage group list
space = record size (260+4) * average number of datasets on volume * number of volumes in each storage group * number of storage groups

Migration data
space = record size (228+4) * number of datasets migrated

Back-up data
space = record size (208+4) * number of dataset back-up versions

Data class construct
space = record size (284+4) * number of data class constructs

Storage class construct
space = record size (280+4) * number of storage class constructs

Management class construct
space = record size (308+4) * number of management class constructs

Storage group construct
space = record size (848+4) * number of storage group constructs

SMS-managed volumes
space = record size (440+4) * number of SMS-managed volumes

Base configuration
space = record size (928+4)

Aggregate group constructs
space = record size (640+4) * number of aggregate group constructs

Optical drives
space = record size (424+4) * number of optical drives

Optical libraries
space = record size (448+4) * number of optical libraries

Cache names
space = record size (176+4) * number of cache names

Accounting information
space = record size (352+4) * number of records

All of the record formats can be mapped using the IDCDOUT mapping macro for formats D, A, and V records and ARCUTILP mapping macro for formats M, B, C, and T records.

IBM also supplies a user exit routine that can be used to intercept DCOLLECT records after they are created but before they are output. The exit is named IDCDCX1 by default but can have any name by using the EXITNAME parameter when DCOLLECT is invoked. The exit allows output to be enhanced, modified, or rejected. The IDCDCX1 routine has to be link edited into the IDCDC01 load module. If a separate routine is used it has to be placed in an authorized library. The exit has to be re-entrant. The exit places the length of the record in register zero and its address in register one. If the record is modified in any way then these registers must be updated. The exit cannot modify the record to have a length greater than 32760 bytes.

The user exit has the option of leaving the record unmodified, changing one or more existing fields, adding new fields to the end of the record, or specifying that the record has to be dropped. Each operation is flagged by placing a particular return code in register fifteen. If a 0 is placed in record fifteen, then the record is left unchanged, if existing fields are changed, then a four is placed in register fifteen: if new fields are added, a four signifies this but the length in register zero has to be modified and register one should be updated with the new record address. To drop the record, code a twelve in register fifteen.

The DCOLLECT facility is a very powerful reporting tool and can be used to enhance DASD administration processes.

John Bradley
Systems Programmer (UK)

Extracting ISPF table information

INTRODUCTION

I was prompted to write the application described in this article when I was asked to amend a number of REXX programs and ISPF panels for our application development department. One of the REXX programs in question was run only once a year, but created and updated a large number of ISPF tables. From the information contained in the tables, which I was told had been accumulated over the past three years, a number of detailed reports were produced. When I came to amend this program, I found that the source had been deleted, and there apparently appeared to be no back-ups. There was very little documentation around, and what there was did not contain any information on how the ISPF tables had been set up, and which KEY and/or NAME variables had been used to create or access the table rows, so I decided to write a function which would extract this information for me. The method I used was to invoke the TBQUERY(returns information about a specified table) and

TBSTATS (obtains statistical information for a specified table) Dialog Manager Table commands, and to display the information which was returned into pop-up windows. Once I had this information I was able to code a new REXX EXEC from a new program specification and access the existing ISPF tables.

This application has proved to be very useful on a number of other occasions. It has now been included in our 'systems programmer's toolkit' and has also been passed to our application development teams.

TABLE INFORMATION

The following options are available:

- Retrieve table statistics
- Obtain table information.

For each option, the user is prompted for the table name, and the library the table is stored in: this library name is then passed as a parameter to the TSO LIBDEF command to allocate the ISPTLIB DDNAME. The following information is then displayed depending on the option selected.

Table statistics

- Date and time the table was created
- Time the table was created
- Date the table was last updated
- Time the table was last updated
- The TSO user-id which last created or updated the table
- The number of table rows generated at create time
- The number of current rows in the table
- The number of updated table rows
- The number of times the table has been updated.

Table information

- The number of rows in the table
- The number of key variables in the table
- The number of name variables in the table
- A list of the key variables used.

TABQUERY REXX EXEC

The LIBDEF command contained in the REXX EXEC must be updated as required. No special authorization is required for this application.

```
/*rexx                                                          */
/*  Program-id           TABQUERY                               */
/*  Remarks              This REXX will obtain the following    */
/*                       information for an ISPF/PDF Table member. */
/*                                                              */
/*       Table Statistics                                       */
/*                       1.  Date table was created             */
/*                       2.  Time table was created             */
/*                       3.  Date of last update                */
/*                       4.  Time of last update                */
/*                       5.  Last user to update the table      */
/*                       6.  Number of rows when the table was  */
/*                           created                            */
/*                       7.  Current no of rows, zero if the table */
/*                           is empty                           */
/*                       8.  Number of existing rows that have been*/
/*                           updated                            */
/*                       9.  Number of times the table has been */
/*                           updated                            */
/*                                                              */
/*       Table Query Information                                */
/*                       1.  List of key variable names in the  */
/*                           table                              */
/*                       2.  List of variable names in the table */
/*                       3.  No of rows contained in the table  */
/*                       4.  No of key variables in the table   */
/*                       5.  No of variables in the table       */
/*trace i*/                                 /* turn tracing on     */
return_code= Ø                             /* init the return code*/
table_alloc_flag = 'ØØ'x                   /* init tab alloc flag */
call house_keeping                         /* let's initialize    */
do forever                                 /* main loop           */
   call display_the_main_panel             /* display main panel  */
   select
      when (pØ1sel= '1') then do           /* retrieve table stats?*/
           call get_table_name             /* get the table name   */
           if (return_code = Ø) then do    /* carry on?            */
              call alloc_table_libs        /* allocate table lib   */
              call retrieve_table_statistics /* yes-               */
              call dealloc_table_library   /* deallocate table lib */
           end
      end
      when (pØ1sel= '2') then do           /* query table info?    */
           call get_table_name             /* get the table name   */
```

```
            if (return_code = Ø) then do     /* carry on?            */
               call alloc_table_libs         /* allocate table lib   */
               call query_table_information  /* yes-                 */
               call dealloc_table_library    /* deallocate table lib */
            end
         end
         otherwise
            nop
      end
      ADDRESS "ISPEXEC" "REMPOP"             /* remove pop-up      */
   end
   display_the_main_panel:
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(4)"  /* pop-up position    */
   ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN1)"  /* display menu panel */
   if (rc = 8) then do                     /* exit?                  */
      ADDRESS "ISPEXEC" "REMPOP"           /* remove pop-up          */
      call dealloc                         /* let's deallocate       */
      exit(Ø)                              /* and quit               */
   end
   if (rc > 8) then do                     /* error?                 */
      say 'TABQPAN1  Display error    rc = 'rc''
      call dealloc                         /* let's deallocate       */
      exit(Ø)                              /* and quit               */
   end
   return
   get_table_name:
   return_code= Ø                          /* set the return code    */
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(4)"  /* pop-up position    */
   ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN2)"  /* display menu panel */
   if (rc = 8) then do                     /* exit?                  */
      return_code= rc                      /* set the return code    */
      ADDRESS "ISPEXEC" "REMPOP"           /* remove pop-up          */
      return                               /* and quit               */
   end
   if (rc > 8) then do                     /* error?                 */
      say 'TABQPAN2  Display error    rc = 'rc''
      call dealloc                         /* let's deallocate       */
      exit(Ø)                              /* and quit               */
   end
   if substr(ttabdsn,1,1) = '''' then do   /* quoted dsn             */
      ttabdsn= translate(ttabdsn,' ','''')  /* yes- remove them       */
      ttabdsn= strip(ttabdsn,b,' ')        /* strip the spaces       */
   end
   else do
      ttabdsn= sysvar(sysuid)'.'ttabdsn    /* move in HLQ            */
   end
   ADDRESS "ISPEXEC" "REMPOP"              /* remove pop-up          */
   return
   retrieve_table_statistics:
   ADDRESS "ISPEXEC" "TBSTATS "ttabname" cdate(tdatecrt) ctime(ttimecrt)
```

```
         udate(tdtelupd) utime(ttmelupd) user(tcuuid)
         rowcreat(tctabrow) rowcurr(tnorows) rowupd(tutabrow)
         tableupd(tttabupd) status1(status1)"
if (rc = 20) then do                        /* severe error?          */
   say 'TABQUERY  TBSTATS error    rc = 'rc''
   call dealloc                             /* let's deallocate       */
   exit(0)                                  /* and quit               */
end
if (status1 = 2) then do
   tabinfo= ' Table Does Not Exist In The Table Input Library'
   ADDRESS "ISPEXEC" "ADDPOP POPLOC(ZCMD)"
   ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN5)"/* display panel     */
   if (rc > 8) then do                      /* error?                 */
      say 'TABQPAN5  Display error    rc = 'rc''
      call dealloc                          /* let's deallocate       */
      exit(0)                               /* and quit               */
   end
   else do
      ADDRESS "ISPEXEC" "REMPOP"            /* remove pop-up          */
      return                                /* return to caller       */
   end
end
if (rc = 0) then do
   tctabrow= strip(tctabrow,l,'0')          /* strip the leading zeros*/
   tnorows=  strip(tnorows,l,'0')           /* strip the leading zeros*/
   tutabrow= strip(tutabrow,l,'0')          /* strip the leading zeros*/
   tttabupd= strip(tttabupd,l,'0')          /* strip the leading zeros*/
   ADDRESS "ISPEXEC" "ADDPOP POPLOC(ZCMD)"
   ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN3)"/* display panel     */
   if (rc > 8) then do                      /* error?                 */
      say 'TABQPAN3  Display error    rc = 'rc''
      call dealloc                          /* let's deallocate       */
      exit(0)                               /* and quit               */
   end
end
ADDRESS "ISPEXEC" "REMPOP"                  /* remove pop-up          */
return
query_table_information:
ADDRESS "ISPEXEC" "TBOPEN "ttabname" NOWRITE"
select
  when (rc = 8) then do
       tabinfo= ' Table Does Not Exist In The Table Input Library'
       ADDRESS "ISPEXEC" "ADDPOP POPLOC(ZCMD)"
       ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN5)" /*display panel */
       if (rc > 8) then do                  /* error?                 */
         say 'TABQPAN5  Display error    rc = 'rc''
         call dealloc                       /* let's deallocate       */
         exit(0)                            /* and quit               */
       end
       else do
```

```
          ADDRESS "ISPEXEC" "REMPOP"      /* remove pop-up         */
          return                          /* return to caller      */
        end
    end
    when (rc = 12) then do
        tabinfo= ' ENQ Failed: Table In Use By Another User'
        ADDRESS "ISPEXEC" "ADDPOP POPLOC(ZCMD)"
        ADDRESS "ISPEXEC" "DISPLAY PANEL(TABQPAN5)" /* display panel */
        if (rc > 8) then do               /* error?                */
          say 'TABQPAN5  Display error    rc = 'rc''
          call dealloc                       /* let's deallocate     */
          exit(0)                            /* and quit             */
        end
        else do
          ADDRESS "ISPEXEC" "REMPOP"         /* remove pop-up        */
          return                             /* return to caller     */
        end
    end
    when (rc \= 0) then do
        say 'TABQUERY  TBOPEN Error  rc = 'rc''
        call dealloc                         /* let's deallocate     */
        exit(0)                              /* and quit             */
    end
    otherwise
        nop
end
ADDRESS "ISPEXEC" "TBQUERY "ttabname" KEYS(tkeyname) names(tvarname)
                  rownum(trownum) keynum(tkeynum) namenum(tnamenum)"
if (rc \= 0) then do                      /* call okay?            */
   say 'TBQUERY error    rc = 'rc''       /* no- inform the user   */
   call close_the_table                   /* close the table       */
   call dealloc                           /* Dealloc               */
   exit(0)                                /* let's quit            */
end
call close_the_table                      /* close the table       */
ztdmark='<<------- End Of List ------>>'
ADDRESS  "ISPEXEC" "VPUT    (ZTDMARK) SHARED"
tabrows=  500                             /* max table rows        */
ADDRESS "ISPEXEC" "TBCREATE KEYTAB NOWRITE REPLACE"
if (rc >  4) then do                      /* call okay?            */
   say 'TBCREATE error    rc = 'rc''      /* no- inform the user   */
   call dealloc                           /* Dealloc               */
   exit(0)                                /* let's quit            */
end
if (tkeynum > 0) then do                  /* any key variables?    */
   tkeyname= strip(tkeyname,l,'(')        /* strip the brackets    */
   tkeyname= strip(tkeyname,t,')')        /* strip the brackets    */
end
if (tnamenum > 0) then do                 /* any name variables?   */
   tvarname= strip(tvarname,l,'(')        /* strip the brackets    */
```

```
   tvarname= strip(tvarname,t,')')       /* strip the brackets     */
end
varname= tvarname                        /* switch for parse       */
keyname= tkeyname                        /* switch for parse       */
do until (keyname = '' & varname= '')
   tkey= ''                              /* set to null            */
   tvar= ''                              /* set to null            */
   if (keyname \= '') then               /* set to null            */
      parse var keyname tkey ' ' keyname /* parse in the keyname   */
   if (varname \= '') then               /* set to null            */
      parse var varname tvar ' ' varname  /* parse in the varname  */
   ADDRESS "ISPEXEC" "TBADD KEYTAB       /* add the entries        */
            SAVE(tkey,tvar)
            MULT("TABROWS")"
   if (rc \= Ø) then do                  /* call okay?             */
      say 'TBADD error    rc = 'rc''     /* no- inform the user    */
      call dealloc                       /* Dealloc                */
      exit(Ø)                            /* let's quit             */
   end
end
ADDRESS "ISPEXEC" "TBTOP KEYTAB"         /* position to top of tab */
if (rc \= Ø) then do                     /* call okay?             */
   say 'TBTOP error    rc = 'rc''        /* no- inform the user    */
   call dealloc                          /* Dealloc                */
   exit(Ø)                               /* let's quit             */
end
trownum=  strip(trownum,l,'Ø')           /* strip the leading zeros */
tkeynum=  strip(tkeynum,l,'Ø')           /* strip the leading zeros */
tnamenum= strip(tnamenum,l,'Ø')          /* strip the leading zeros */
ADDRESS "ISPEXEC" "ADDPOP POPLOC(ZCMD)"  /* pop-up position         */
ADDRESS "ISPEXEC" "TBDISPL KEYTAB PANEL(TABQPAN4)"
if (rc = 8) then do                      /* pf3 or return?         */
   call close_the_temp_table             /* close the table        */
   ADDRESS "ISPEXEC" "REMPOP"            /* remove pop-up          */
   return                                /* and quit               */
end
if (rc > 8) then do                      /* error?                 */
   call close_the_temp_table             /* close the table        */
   say 'TABQPAN4  Display error    rc = 'rc''
   call dealloc                          /* let's deallocate       */
   exit(Ø)                               /* and quit               */
end
call close_the_temp_table                /* close the table        */
ADDRESS "ISPEXEC" "REMPOP"               /* remove pop-up          */
return
house_keeping:
/*                                                                 */
/* Set the error mode                                             */
/*                                                                 */
address "ISPEXEC" "CONTROL ERRORS RETURN"
```

```
/*                                                                  */
/* Dynamically allocate the panel library                           */
/*                                                                  */
address "ISPEXEC" "LIBDEF ISPPLIB DATASET ID('XXXXXXX.PANELS')"
if (rc > Ø) then do                        /* error?                */
   say 'TABQUERY  LIBDEF ISPPLIB Error    rc = 'rc''
   exit(Ø)                                 /* and quit              */
end
return
alloc_table_libs:
/*                                                                  */
/* Dynamically allocate the table library                           */
/*                                                                  */
address "ISPEXEC" "LIBDEF ISPTLIB DATASET ID('"ttabdsn"')"
if (rc > Ø) then do                        /* error?                */
   say 'TABQUERY  LIBDEF ISPTLIB Error    rc = 'rc''
   call dealloc                            /* remove allocation     */
   exit(Ø)                                 /* and quit              */
end
table_alloc_flag = 'Ø1'x                   /* table file allocated  */
return
dealloc_table_library:
ADDRESS "ISPEXEC" "LIBDEF ISPTLIB"         /* remove allocation     */
table_alloc_flag = 'ØØ'x                   /* reset allocate flag   */
return
dealloc:
if (table_alloc_flag = 'Ø1'x) then         /* table allocated?      */
  call dealloc_table_library               /* yes                   */
ADDRESS "ISPEXEC" "LIBDEF ISPPLIB"         /* remove allocation     */
return
close_the_table:
ADDRESS "ISPEXEC" "TBCLOSE "ttabname""
if (rc \= Ø) then do
   say 'TABQUERY  TBCLOSE error    rc = 'rc''
   call dealloc                            /* let's deallocate     */
   exit(Ø)                                 /* and quit             */
end
return
close_the_temp_table:
ADDRESS "ISPEXEC" "TBCLOSE  KEYTAB"        /* close the table       */
if (rc \= Ø) then do                       /* call okay?            */
   say 'TBCLOSE error    rc = 'rc''        /* no- inform the user   */
   call dealloc                            /* Dealloc               */
   exit(Ø)                                 /* let's quit            */
end
table_open_flag= 'ØØ'x                     /* reset table open flag */
return
```

TABQPAN1 ISPF PANEL

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(BLUE)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(58,9)
%
%COMMAND ===>_ZCMD          %     SCROLL ===>_AMT   +
%
%
#          _z#  1. Retrieve Table Statistics
#               2. Obtain Table Information
#
% Please select an option by entering the required number.
%
)INIT
.ZVARS=  '(p01sel)'
&ZCMD=   ' '
&p01sel= '_'
&check=  '1,2'
.CURSOR= p01sel
&ZWINTTL=  'Table Information Dialog'
)REINIT
&p01sel= '_'
&ZCMD=   ' '
&ZWINTTL=  'Table Information Dialog'
.CURSOR= p01sel
)PROC
ver (&p01sel,listv,&check)
)END
```

TABQPAN2 ISPF PANEL

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(BLUE)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(55,8)
%
%COMMAND ===>_ZCMD          %   SCROLL ===>_AMT   +
%
```

```
%  ------------------------------------------------
%  >#Table Name:    _z       %                     >
%  >#Table Library: _z                            %>
%  ------------------------------------------------
)INIT
.ZVARS=   '(ttabname,ttabdsn)'
&ttabname= ''
&ttabdsn=  ''
&ZCMD=    ' '
.CURSOR=   ttabname
&ZWINTTL= 'Table And DSN Prompt'
)REINIT
.CURSOR=   ttabname
&ZCMD=    ' '
&ttabname= ''
&ttabdsn=  ''
&ZWINTTL= 'Table And DSN Prompt'
)PROC
ver (&ttabname,NONBLANK,NAME)
ver (&ttabdsn,NONBLANK,DSNAME)
)END
```

TABQPAN3 ISPF PANEL

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(RED)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(YELLOW)
# TYPE(TEXT)    INTENS(HIGH) COLOR(BLUE)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
£ TYPE(OUTPUT)  COLOR(YELLOW)
)BODY WINDOW(56,14)
%
% COMMAND ===>_ZCMD          %    SCROLL ===>_AMT   +
%
# Table Name:                          £z       %
# Date Table Created:                  £z       %
# Time Table Created:                  £z       %
# Date Table Last Updated:             £z       %
# Time Table Last Updated:             £z       %
# Created Or Updated By USERID:        £z       %
# Number Of Table Rows At Create Time: £z       %
# Number Of Current Table Rows:        £z       %
# Number Of Updated Rows:              £z       %
# Number Of Times Table Updated:       £z       %
)INIT
.ZVARS= '(ttabname,tdatecrt,ttimecrt,tdtelupd,ttmelupd, +
```

```
tcuuid,tctabrow,tnorows,tutabrow,tttabupd)'
&ZCMD= ' '
.CURSOR=   ZCMD
&ZWINTTL= 'Table Statistics'
)REINIT
&ZCMD=    ' '
.CURSOR=   ZCMD
&ZWINTTL= 'Table Statistics'
)PROC
)END
```

TABQPAN4 ISPF PANEL

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(BLUE)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
£ TYPE(OUTPUT)  COLOR(YELLOW)
)BODY WINDOW(55,15)
%
% COMMAND ===>_ZCMD          %    SCROLL ===>_AMT   +
%
# Table Name:                          _z       %
# Number Of Rows In Table:             _z       %
# Number Of Key Variables In Table:  _z       %
# Number Of Name Variables In Table: _z       %
#
# Key Variable       Variable
# Names              Names
%
)MODEL
 £z       %         £z       %
)INIT
.ZVARS= '(ttabname,trownum,tkeynum,tnamenum,tkey,tvar)'
&ZCMD= ' '
.CURSOR=   ZCMD
&ZWINTTL= 'Table Information'
)REINIT
&ZCMD=    ' '
.CURSOR=   ZCMD
&ZWINTTL= 'Table Information'
)PROC
)END
```

TABQPAN5 ISPF PANEL

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(BLUE)
£ TYPE(OUTPUT)  INTENS(HIGH) COLOR(YELLOW)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(RED)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(58,9)
%
%COMMAND ===>_ZCMD          %     SCROLL ===>_AMT   +
%
%    ------------------------------------------------
%    >                                              >
%    >£z                                           %>
%    >                                              >
%    ------------------------------------------------
%
)INIT
.ZVARS=  '(tabinfo)'
&ZCMD=   ' '
.CURSOR= ZCMD
&ZWINTTL=  'Table Query Information Panel'
)REINIT
.CURSOR= ZCMD
&ZWINTTL=  'Table Query Information Panel'
)PROC
)END
```

R F Perretta
Senior Systems Programmer (UK)

Providing EXCP counts for unit record datasets

INTRODUCTION

JES2 exit 33 is defined as the Subsystem Interface (SSI) dataset close exit. It is intended to be an exit point whereby someone can receive control during JES2's close processing of a subsystem dataset and modify its characteristics, or validity check those characteristics. The

type of dataset being processed is specified in a parameter list whose address is passed to exit 33 in general purpose register 1.

Be forewarned! Providing EXCP counts for unit record datasets is **not** an IBM-conceived use of exit 33. The successful functioning of this code is intimately dependent upon the constancy of the programming interface provided by IBM, as dictated by its philosophy of protecting its customer's investment in IBM software, and the continued ability to accurately modify IEFTB728 via IMASPZAP.

IEFTB728 is an integral part of providing EXCP counts for unit record datasets since it is the ESA routine that constructs SMF type 30 records, and from which IEFACTRT obtains statistical billing information related to datasets.

Providing EXCP counts for unit record datasets is a process that has three parts:

1 Create and enable JES2 exit 33
2 Modify IEFTB728
3 Include JES2 as a type of unit in the accounting exit, IEFACTRT.

The first and simplest part has been done – the code is included in this article. The second part is slightly trickier and involves several steps: caution is advised. For the first step, use IBM's IMASPZAP utility to obtain a DUMPT listing of IEFTB728: it is a CSECT within IEEMB836 that resides within SYS1.LPALIB. Then scan that listing for the instruction that zeroes the EXCP count for unit record datasets (D715 2000 2000). It will actually be the second such instruction in IEFTB728. If an assembled listing of IEFTB728 is available, scan for XC SMF30EXP(22,Q02_@GN01336),SMF30EXP(@02_@GN01336). It can be found after the following comments:

```
SMF3ØEXP = SMF3ØEXP && SMF3ØEXP; /* CLEAR OUT ENTRY
```

Modify the branch instruction that follows the XC SMF30EXP(22,@02_... instruction to branch to the handle on the following instruction (phew!): 58E0,3008 which with a listing would be:

```
L     @14,TCTDCTR(,@Ø3_DDPTR)
```

The handle of that instruction for HBB5520 dated 94297 is

@RC00803. The handle of the instruction that follows the branch instruction that is to be modified is @RF00800. The last step is to establish this modification to ESA in the proper way, as a USERMOD using SMP as shown later on in this article.

The third part is to modify your installation's IEFACTRT routine. Here are a few helpful hints to guide you in the proper direction. Include JES2 as a valid 'device type' for device entries whenever SMF30DEV in an EXCP section equals hexadecimal zeroes. However, if SMF30DEV = 0 and SMF30CUA = X'80', then the 'device type' is VIO. Continuing along this merry path, if SMF30DEV = 0, SMF30CUA is not equal to X'80', and SCTSBYT1 = SCTDUMMY, then the device type is a dummy one. Locating SCTSBYT1 is fairly easy. Once again, using a non-standard non-IBM defined 'programming interface', general purpose register 12 currently points to the OS Linkage Control Table (LCT) upon entry to IEFACTRT from the OS initiator. This is not clearly documented as such, even though allusions are made to this fact in the *MVS/ESA Installations Exit* manual, and the assumption that register 12 will always point to the LCT may not be valid. It is entirely possible that in future releases of ESA that the LCT may not be available, or will be pointed at by some other register. Rather than describing the code in IEFACTRT that was used to locate SCTSBYT1, I stripped it from there and included it with this article. It is fairly simple code and should be easy for anyone to customize.

Remember to clear the high-order bit in SMF30BLK in order to obtain an accurate 'image' count for Advanced Function Presentation's (AFP's) page datasets (unless your installation needs the revenue). If your installation has no IEFACTRT exit that processes SMF type 30 records to display step-end and job-end statistical information within a job's SMBs, ask Xephon to notify me and I will be only too willing to send in a copy of mine, not necessarily for publication, but for distribution to needy installations.

I have provided a sample job stream that I used to test JES2 exit 33, the results of that test, and the required ZAP to IEFTB728 in SMP format. Lest I forget, and in case someone needs them, I have also included the parameters that JES2 requires before invoking this exit.

JES2 EXIT 33 SOURCE CODE

```
         TITLE 'JES2 USER''S EXIT 33 FOR SSI CLOSE OF DATASET'
*********************************************************************
* MODULE NAME = HASPXJ33                                            *
* FUNCTION = EXIT 33 RECEIVES CONTROL DURING THE SUBSYSTEM'S        *
*             PROCESS OF CLOSING A DATASET.                         *
* CONTENTS OF REGISTERS AT ENTRY TO HASPXJ33:                       *
*     R1  - POINTER TO TWENTY-FIVE BYTE PARAMETER LIST              *
*           THAT HAS BEEN FORMATTED AS FOLLOWS:                     *
*           +Ø  - TYPE OF DATASET                                   *
*                 Ø  - INTERNAL READER FOR A BATCH JOB              *
*                 4  - INTERNAL READER FOR A STARTED TASK           *
*                 8  - INTERNAL READER FOR A TIME SHARING USER      *
*                 12 - SYSIN DATASET                                *
*                 16 - SYSOUT DATASET                               *
*                 2Ø - PROCESS SYSOUT DATASET                       *
*                 24 - SPOOL BROWSE DATASET                         *
*                 28 - THE DATASET'S TYPE IS UNKNOWN                *
*           +2  - ERROR INDICATOR                                   *
*                 BIT 7 - Ø ERROR-FREE CLOSE PROCESSING             *
*                         1 ERROR DURING CLOSE PROCESSING           *
*           +4  - FOR AN INTERNAL READER, THE ADDRESS OF ITS DCT    *
*                 FOR A SYSIN OR SYSOUT DATASET, ITS SDB ADDRESS    *
*           +8  - THE ADDRESS OF THE JOB'S SJB, OR Ø                *
*           +12 - THE ADDRESS OF THE DATASET'S JFCB                 *
*           +2Ø - THE ADDRESS OF THE DATASET'S DEB                  *
*           +24 - THE ADDRESS OF THE DATASET'S PDDB, OR Ø FOR INTRDR *
*           +28 - THE ADDRESS OF THE DATASET'S IOT, OR Ø FOR INTRDR  *
*     R11 - ADDRESS OF HCCT                                         *
*     R13 - ADDRESS OF A SAVE AREA                                  *
*     R14 - ADDRESS OF THE RETURN POINT                             *
*     R15 - ADDRESS OF HASPXJ33'S ENTRY POINT                       *
*********************************************************************
HASPXJ33 $MODULE ENVIRON=USER,                                      C
               RMODE=ANY,                                           C
               TITLE='JES2 USER EXIT 33',                           C
               $BUFFER,            GENERATE HASP'S BUFFER DSECT     C
               $CADDR,             GENERATE HASP'S CADDR DSECT      C
               $HASPEQU,           GENERATE HASP'S EQUATES DSECT    C
               $HCCT,              GENERATE HASP'S HCCT DSECT       C
               $HFAME,             GENERATE HASP'S HFAME DSECT      C
               $IOT,               GENERATE HASP'S IOT DSECT        C
               $JCT,               GENERATE HASP'S JCT DSECT        C
               $MIT,               GENERATE HASP'S MIT DSECT        C
               $MITETBL,           GENERATE HASP'S MTE DSECT        C
               $PADDR,             GENERATE HASP'S PADDR DSECT      C
               $PARMLST,           GENERATE HASP'S PARMLST DSECT    C
               $PDDB,              GENERATE HASP'S PCE DSECT        C
               $PSV,               GENERATE HASP'S PSV DSECT        C
               $SCAT,              GENERATE HASP'S SCAT DSECT       C
```

```
               $SDB,               GENERATE HASP'S SDB DSECT           C
               $SJB,               GENERATE HASP'S SJB DSECT           C
               $TAB,               GENERATE HASP'S TAB DSECT           C
               $TQE,               GENERATE HASP'S TQE DSECT           C
               $USERCBS,           GENERATE HASP'S USERCBS DSECT       C
               $XECB,              GENERATE HASP'S XECB DSECT          C
               IEDB,               GENERATE ESA'S IEDB DSECT           C
               IOBE,               GENERATE ESA'S IOBE DSECT           C
               JFCB,               GENERATE ESA'S JFCB DSECT           C
               PSA,                GENERATE ESA'S PSA DSECT            C
               RPL,                GENERATE ESA'S RPL DSECT            C
               TIOT,               GENERATE ESA'S TIOT DSECT           C
               TCB,                GENERATE ESA'S TCB DSECT            C
               TCT                 GENERATE ESA'S TCT DSECT
         TITLE 'JES2 USER EXIT 33 -- SSI DATASET CLOSE - MAIN ROUTINE'
EXITØ33A $ENTRY  BASE=R12          ENTRY POINT OF ROUTINE
         $SAVE                     SAVE CALLER'S REGISTERS
         USING TN33DSEC,R9         ESTABLISH PARM ADDRESSABILITY
         USING SDB,R4              ESTABLISH SDB ADDRESSABILITY
         USING PSA,RØ              ESTABLISH PSA ADDRESSABILITY
         USING SJB,R5              ESTABLISH SJB ADDRESSABILITY
         USING INFMJFCB,R6         ESTABLISH JFCB ADDRESSABILITY
         USING BFPDSECT,R1Ø        ESTABLISH BFPDSECT ADDRESSABILITY
         LR    R12,R15             PRIME BASE REGISTER
         LR    R9,R1               SAVE PARAMETER REGISTER
         LM    R4,R6,TN33SDB       PRIME ADDRESSABILITY TO A FEW DSECTS
         SR    R1,R1               CLEAR A VOLATILE GENERAL PURPOSE REG
         IC    R1,TN33TYPE         FETCH TYPE-OF-DATASET INDICATOR
         B     *+4(R1)             ENTER ROUTINE TO PROCESS THAT TYPE
         B     TN33EXIT            INTERNAL READER FOR A BATCH JOB
         B     TN33EXIT            INTERNAL READER FOR A STARTED TASK
         B     TN33EXIT            INTERNAL READER FOR A TSO USER
         B     TN33INR             SYSIN DATASET
         B     TN33OUTR            SYSOUT DATASET
         B     TN33EXIT            PROCESS SYSOUT DATASET
         B     TN33EXIT            SPOOL BROWSE DATASET
         B     TN33EXIT            UNKNOWN TYPE OF A DATASET
TN33INR  DS    ØH
TN33OUTR DS    ØH
*************************************************************************
*           ESTABLISH ADDRESSABILITY TO THE CURRENT JOB'S TASK          *
*           CONTROL BLOCK(TCB) AND ITS TIMING CONTROL TABLE(TCT)        *
*************************************************************************
         L     R1,PSATOLD          POINT TO CURRENT TCB
         USING TCB,R1              ESTABLISH TCB ADDRESSABILITY
         ICM   R7,15,TCBTCT        SET TCT POINTER OR RETURN CODE
         BE    TN33EXIT            EXIT IF NON-EXISTENT
         USING SMFTCT,R7           ESTABLISH TCT ADDRESSABILITY
         L     R8,TCBTIO           FETCH TIOT POINTER
         USING TIOT1,R8            ESTABLISH TIOT ADDRESSABILITY
         DROP  R1                  FORGET TCB
```

```
         ICM   RØ,15,TCTIOTBL      FETCH POINTER TO TCT I/O TABLE
         BE    TN33EXIT           BRANCH IF THIN AIR
         LR    R15,RØ             REPEAT RØ
         USING TCTTIOT,R15        ESTABLISH TCTTIOT ADDRESSABILITY
*************************************************************************
*        LOCATE THE ENTRY IN THE TCT THAT IS ASSOCIATED WITH            *
*        THIS DATASET                                                   *
*************************************************************************
         LA    R15,TCTIODSP       DD LOOK UP TABLE ENTRIES
         USING TCTIODSP,R15       ESTABLISH TCTIODSP ADDRESSABILITY
TN33LSRC L     R1,TCTDCBTD        OFFSET TO DD ENTRY IN TIOT
         LA    R1,TIOCNJOB(R1)    ADDRESS OF DD ENTRY IN TIOT
         USING TIOENTRY,R1        ESTABLISH TIOENTRY ADDRESSABILITY
         CLC   TIOEDDNM,SDBDDNM   TEST FOR CORRESPONDING ENTRY
         BE    TN33LCNT           BRANCH IF FOUND
TN33LNXT LA    R15,TCTDCBLE       NEXT TABLE ENTRY
         SR    R1,R1              ZERO WORK REGISTER
         C     R1,TCTDCBTD        HAS AN EXHAUSTIVE SEARCH BEEN DONE?
         BNE   TN33LSRC           IF NOT, CHECK NEXT DD STATEMENT
         B     TN33EXIT           OTHERWISE EXIT
         DROP  R1                 FORGET TIOT
*************************************************************************
*        ACCUMULATE EXCP COUNTS FOR THIS DATASET:  THE COUNT OF         *
*        LOGICAL RECORDS THAT IS MAINTAINED IN SDBDRECD IS COMBINED     *
*        WITH THE RECORD NUMBER IN THE DATASET'S CURRENT UN-            *
*        PROTECTED JES2 BUFFER.                                         *
*************************************************************************
TN33LCNT L     R1,TCTIOTSD        OFFSET FROM TCT I/O TBL TO DD ENTRY
         LR    R15,RØ             POINT TO TCT I/O TABLE
         AR    R1,R15             COMPUTE ADDRESS OF DD ENTRY IN TABLE
         USING TCTDDENT,R1        ESTABLISH TCTDDENT ADDRESSABILITY
         SR    R2,R2              CLEAR A GENERAL PURPOSE REGISTER
         ICM   R1Ø,15,SDBUBF      IS AN UNPROTECTED BUFFER PRESENT?
         BE    TN33NBUF           BRANCH IF NOT
         ICM   R2,7,BFDRBA+5      FETCH NR OF NEXT RECORD
         BE    TN33NBUF           BRANCH IF NONE
         BCTR  R2,RØ              SET NR OF CURRENT RECORD
TN33NBUF CVB   RØ,SDBDRECD        LOGICAL RECORD COUNT
         TM    SDBFLG1,SDB1GET    TEST IF SYSIN DATASET
         BZ    TN33IOC1           BRANCH IF NOT
         A     RØ,TCTDCTR         ACCUMULATE I/O REQUESTS
TN33IOC1 TM    SDBFLG3,SDB3PAGE   TEST IF THIS IS A PSF PAGE COUNT
         BZ    TN33IOCT           BRANCH IF NOT(GO ACCUM I/O COUNT)
         CVB   RØ,SDBDPAGE        IF SO, LOAD PSF PAGE COUNT
         O     RØ,TN33PPG8        SET PSF PAGE FLAG BIT FOR IEFACTRT
*************************************************************************
*           SAVE THE EXCP COUNT AND BLOCK SIZE OF A UNIT-RECORD         *
*           DATASET IN THE CURRENT JOB'S TCT                            *
*************************************************************************
TN33IOCT AR    RØ,R2              ADD EXCP COUNT FROM PREVIOUS BUFFERS
         ST    RØ,TCTDCTR         SAVE I/O COUNT FOR DATASET IN TCT
```

```
         MVC   TCTBLKSZ,JFCBLKSI   STOW BLOCK-SIZE OF DATASET IN TCT
TN33EXIT SR    R15,R15             SHOW I'M A SUCCESSFUL DUDE
         $RETURN RC=(R15)          BACK TO DUST
         TITLE 'CONSTANTS, DSECTS, AND OTHER JUNK FOR JES2''S EXIT 33'
*********************************************************************
*        CONSTANTS AND OTHER JUNK                                   *
*********************************************************************
         DS    ØF
TN33PPG8 DC    X'80000000'
         LTORG
*********************************************************************
*        DSECT FOR PARAMETER LIST PASSED TO JES52X33                *
*********************************************************************
TN33DSEC DSECT
TN33TYPE DS    X                   TYPE OF DATASET BEING CLOSED
TN33COND DS    X                   CONDITION BYTE FOR EXIT 33
         DS    X                   RESPONSE BYTE FOR EXIT 33
         DS    X                   RESERVED
TN33SDB  DS    A                   ADDRESS OF DCT, OR SDB
TN33SJB  DS    A                   ADDRESS OF SJB, OR ZERO
TN33JFCB DS    A                   ADDRESS OF JFCB
TN33DEB  DS    A                   ADDRESS OF DEB
TN33PDDB DS    A                   ADDRESS OF PDDB
TN33IOT  DS    A                   ADDRESS OF IOT
TN33BAT  EQU   Ø                   INTERNAL READER FOR BATCH JOBS
TN33ERR  EQU   1                   ERROR ENCOUNTERED CLOSE OF DATASET
TN33IN   EQU   X'ØC'               SYSIN DATASET
TN33OUT  EQU   X'1Ø'               SYSOUT DATASET
         TITLE 'JES2''S USER EXIT 33 -- EPILOG ($MODEND)'
        $MODEND
         END   ,                   END OF HASPXJ33
```

IEFACTRT AMENDMENT

This is that portion of code in IEFACTRT that is used to locate SCTSBYT1. General purpose register 8 is used to enter it. ADDRLCT is within a previously acquired storage area where GPR 12 was stowed at entry to IEFACTRT. CMRUNIT and CMRADDR are fields within an informational line that will be written to a job's SMBs. The two constants used are:

```
CMRDYN  DC  CL4'DYN'
CMRDUMY DC  CL4'DMY'

*        DETERMINE WHETHER DATASET IS ACTUALLY A DUMMY ONE
CMRCKDMY ICM   R1,15,ADDRLCT       FETCH LINKAGE CONTROL TABLE ADDRESS
         BCR   8,R8                RETURN IF UNAVAILABLE
         USING LCTDSECT,R1         ESTABLISH LCT ADDRESSABILITY
         SR    RØ,RØ               CLEAR VOLATILE REGISTER
```

```
         ICM   RØ,7,LCTSCTAD+1     ADDRESS OF STEP'S STEP CONTROL TABLE
         BCR   8,R8                RETURN IF ZERO
         LR    R1,RØ               PRIME SCT BASE
         USING INSMSCT,R1          SET STEP CONTROL TBL ADDRESSABILITY
         SR    R14,R14             CLEAR VOLATILE REGISTER
         ICM   R14,7,SCTFSIOT      GET TTR OF FIRST SIOT
         BCR   8,R8                RETURN IF ZERO
         LA    R14,16(R14)         SET VIRTUAL ADDRESS OF SIOT
         USING INDMSIOT,R14        ESTABLISH SIOT ADDRESSABILITY
CMRFINDD CLC   SMF3ØDDN,SCTDDNAM   TEST IF SAME DDNAME
         BE    CMRSAMDD            BRANCH IF SO
         ICM   R14,7,SCTPSIOT      FETCH TTR OF NEXT SIOT
         LA    R14,16(R14)         VIRTUAL ADDRESS OF SIOT
         BNZ   CMRFINDD            BRANCH IF NOT
         MVC   CMRADDR,CMRDYN      SHOW UNLOCATABLE DD STATEMENT
         BCR   15,R8               RETURN TO CALLER
CMRSAMDD TM    SCTSBYT1,SCTDUMMY   TEST IF DUMMY DATASET
         BNO   Ø(R8)               BRANCH IF NOT
         MVC   CMRUNIT,CMRDUMY     SHOW DATASET IS A DUMMY ONE
         BCR   15,R8               RETURN TO CALLER
         DROP  R1,R14
```

The following DSECTs are required:

```
LCTDSECT DSECT
         IEFALLCT

         IEFASIOT

SCTDSECT DSECT
         IEFASCTB
```

PARAMETERS REQUIRED BY JES2 VERSION 5

```
LOADMOD(HASPXJ33) STORAGE=LPA
EXIT(Ø33) ROUTINE=EXITØ33A,
         STATUS=ENABLED,
         TRACE=NO
```

SYSTEM MESSAGE BLOCKS (SMBS)

This is a listing of the the System Message Blocks (SMBs) created for the test job.

```
IEF236I ALLOC. FOR XEPHON STEP1
IEF237I VIO  ALLOCATED TO SYSUT2
IEF237I DMY  ALLOCATED TO SYSPRINT
IEF237I DMY  ALLOCATED TO SYSIN
IEF237I JES2 ALLOCATED TO SYSUT1
```

```
IEF142I XEPHON STEP1 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS97063.T083129.RA000.XEPHON.LIMERICK.H01   PASSED
IEF285I   AG0305K.XEPHON.JOB00717.D0000101.?           SYSIN
*******************************************************************************
*   JOB       STEP    NUM  PGM NAME      STEP START            STEP END        *
* XEPHON    STEP1       1  IEBGENER 03/04/97 08:31:30.08  03/04/97 08:31:30.38 *
*                                                                             *
* ELAPSED TIME: 00:00:00.30  CPU TIME: TCB = 00:00:00.01     SRB = 00:00:00.00 *
* SERVICE UNITS: CPU =        0  SRB =        0  I/O =        0  MSO =       0 *
*                                                                             *
* PI      0  PO      0                  VI      0  VO      0  VR      0        *
* CSA: PAGE-IN      0  HYPER-IN      0   LPA: PAGE-IN      0  HYPER-OT      0  *
* SWAPPING: SEQUENCES    0  IN     0  OUT     0      PAGES STOLEN:      0      *
*                                                                             *
* REGION(VIRT) SIZE      :       100K REQUESTED                               *
*                                 52K USED BELOW 16MEG                        *
*                                  4K USED ABOVE 16MEG/EXTENDED               *
* MAX STORAGE ALLOCATED          276K BELOW 16MEG                             *
* TO SYSTEM (LSQA+SWA)   :     9,252K ABOVE 16MEG/EXTENDED AREA               *
*                                                                             *
* DDNAME   UNIT ADDR BLKSIZ -- EXCPS --  DDNAME   UNIT ADDR BLKSIZ -- EXCPS -- *
* SYSUT2   VIO  ****    N/A           5  SYSPRINT DMY  ****    N/A         N/A *
* SYSIN    DMY  ****    N/A         N/A  SYSUT1   JES2 ****     80           6 *
*                                                                             *
* DISK EXCPS =         0   TAPE EXCPS =         0   JES + VIO =        11      *
* TAPE MOUNTS: SPECIFIC =  0  NON-SPECIFIC =  0    TAPE UNITS USED:  0        *
*                                                                             *
* STEP COMPLETION CODE: CC=00                                                 *
*******************************************************************************
IEF373I STEP /STEP1   / START 97063.0831
IEF374I STEP /STEP1   / STOP  97063.0831 CPU    0MIN 00.01SEC SRB    0MIN 00.00S
IEF236I ALLOC. FOR XEPHON STEP2
IEF237I DMY  ALLOCATED TO SYSPRINT
IEF237I DMY  ALLOCATED TO SYSIN
IEF237I VIO  ALLOCATED TO SYSUT1
IEF237I JES2 ALLOCATED TO SYSUT2
IEF142I XEPHON STEP2 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS97063.T083129.RA000.XEPHON.LIMERICK.H01   DELETED
IEF285I   AG0305K.XEPHON.JOB00717.D0000102.?           SYSOUT
*******************************************************************************
*   JOB       STEP    NUM  PGM NAME      STEP START            STEP END        *
* XEPHON    STEP2       2  IEBGENER 03/04/97 08:31:30.44  03/04/97 08:31:30.62 *
*                                                                             *
* ELAPSED TIME: 00:00:00.18  CPU TIME: TCB = 00:00:00.02     SRB = 00:00:00.00 *
* SERVICE UNITS: CPU =        0  SRB =        0  I/O =        0  MSO =       0 *
*                                                                             *
* PI      0  PO      0                  VI      0  VO      0  VR      1        *
* CSA: PAGE-IN      0  HYPER-IN      0   LPA: PAGE-IN      0  HYPER-OT      0  *
* SWAPPING: SEQUENCES    0  IN     0  OUT     0      PAGES STOLEN:      0      *
*                                                                             *
```

```
* REGION(VIRT) SIZE      :       100K REQUESTED                                *
*                                 56K USED BELOW 16MEG                        *
*                                  4K USED ABOVE 16MEG/EXTENDED               *
* MAX STORAGE ALLOCATED          276K BELOW 16MEG                             *
* TO SYSTEM (LSQA+SWA)   :     9,260K ABOVE 16MEG/EXTENDED AREA               *
*                                                                             *
* DDNAME   UNIT ADDR BLKSIZ -- EXCPS --  DDNAME   UNIT ADDR BLKSIZ -- EXCPS -- *
* SYSPRINT DMY  ****    N/A         N/A  SYSIN    DMY  ****    N/A         N/A *
* SYSUT1   VIO  ****     80           5  SYSUT2   JES2 ****     80           5 *
*                                                                             *
* DISK EXCPS =          0   TAPE EXCPS =         0   JES + VIO =        10     *
* TAPE MOUNTS: SPECIFIC =  0  NON-SPECIFIC =  0    TAPE UNITS USED:  0        *
*                                                                             *
* STEP COMPLETION CODE: CC=00                                                 *
*******************************************************************************
IEF373I STEP /STEP2   / START 97063.0831
IEF374I STEP /STEP2   / STOP  97063.0831 CPU    0MIN 00.02SEC SRB    0MIN 00.00S
*******************************************************************************
*                                                                             *
*   JOB ACCOUNTING INFORMATION  -  O I R  -  CENTRAL FACILITY RESOURCES ONLY  *
*                                                                             *
* JOB NAME P Q  JOB#  P/D   SYSTEM  BILLING CODE       PROGRAMMER'S NAME FIELD *
* XEPHON   1 J  0717  P05K   VS03   BOGUS000           ANONYMOUS               *
*                                                                             *
*      JOB START               JOB END          JOB ELAPSED TIME             *
* 03/04/97 08:31:30.08   03/04/97 08:31:30.71       00:00:00.63               *
*                                                                             *
* TCB CRU SECONDS.....................       .03  COST.............       $.29 *
* PROCESS EXCPS.......................        22  COST.............       $.06 *
* TOTAL CARDS READ....................        18  COST.............       $.20 *
* TOTAL LINES GENERATED...............         5  COST IF PRINTED...       $.60 *
* ESTIMATED COST OF COMPUTER RESOURCES CONSUMED BY THIS JOB.......      $1.15 *
*******************************************************************************
IEF375I  JOB /XEPHON  / START 97063.0831
IEF376I  JOB /XEPHON  / STOP  97063.0831 CPU    0MIN 00.03SEC SRB    0MIN 00.00S
```

SMP JCL

These are the SMP control statements that can be used to APPLY the required ZAP to IEFTB728. Never, never, never, ACCEPT USERMODs!

```
...
...
Installation-dependent JCL for an SMP run
...
...
//SMPPTFIN DD  *
++USERMOD (TN00004).  /* EXCP COUNTS RESOLUTION */
```

```
++VER (Z038) FMID(HBB5520). /* MVS/ESA 5.2 */
++ZAP (IEFTB728) .
*******************************************
*                                         *
*  THIS ZAP FORCES THE EXCP COUNT AND     *
*  BLOCK SIZE OF SUBSYSTEM DATASETS TO BE  *
*  MOVED INTO TYPE 30 SMF RECORDS.        *
*                                         *
*  LOCATE THE TEST FOR DEVICE ENTRY OF ALL  *
*  ZEROES( NEAR COMMENT /* EXCP COUNTS NOT  *
*  VALID ).                               *
*  ALTER BRANCH AFTER                     *
*  XC SMF30EXP(22,@02_...),SMF30EXP(@02_...)*
*  INSTRUCTION TO GO TO LABEL @RC00803    *
*  (AFTER COMMENT /* GET BLOCK COUNT DELTA  *
*   AFTER LABEL @RF00803)                 *
*                                         *
*@RC00803 L  @14,TCTDCTR(,@03_DDPTR)      *
*         LR @15,@14                      *
*                                         *
*******************************************
*
 NAME   IEEMB836 IEFTB728
   VERIFY 0E08 47F0,CE88          B     @RC00798
   VERIFY 0E0C 5870,3000 @RF00798 L     @07,TCTUCBP(,@03_DDPTR)
   VERIFY 0E1A 4770,CE28          BNZ   @RF00800
   VERIFY 0E1E D715,2000,2000     XC    SMF30EXP(22,@02_@GN01336),
*                                              SMP30EXP(@02_@GN01336)
   VERIFY 0E58 58E0,3008 @RC00803 L     @14,TCTDCTR(,@03_DDPTR)
   VERIFY 0E5C 18FE               LR    @15,@14
   VERIFY 0E24 47F0,CE88          B     @RC00800
   REP    0E24 47F0,CE58          B     @RC00803
/*
//SMPCNTL DD *
 SET BOUNDARY(GLOBAL).
 RECEIVE S(TN00004).
 SET BOUNDARY(SMPTARG).
 APPLY   S(TN00004).
```

TEST JCL

This job stream is used to test HASPXJ33.

```
//XEPHON   JOB (1,BOGUS000,P05K,1),ANONYMOUS,MSGCLASS=H,CLASS=J
/*ROUTE XEQ DON
//STEP1    EXEC PGM=IEBGENER,REGION=100K
//SYSUT2   DD UNIT=VIO,SPACE=(TRK,1),DSN=&&LIMERICK,DISP=(,PASS)
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD *
```

```
THERE WAS A YOUNG LADY NAMED HARRIS
WHOM MOST NOTHING WOULD EVER EMBARRASS
'TIL THE BATH SALTS ONE DAY
IN THE TUB WHERE SHE LAY
TURNED OUT TO BE PLASTER OF PARIS
//STEP2    EXEC PGM=IEBGENER,REGION=100K
//SYSPRINT DD DUMMY
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=&&LIMERICK,DISP=(OLD,DELETE)
//SYSUT2   DD SYSOUT=*
```

Systems Programmer
State of Tennessee (USA)

MQSeries for MVS/ESA channel security exit

THE PROBLEM

We have MQSeries on OS/2 linked to MQSeries on MVS/ESA. OS/2 has limited security capabilities so it is important to verify the authenticity of a connecting MQSeries Queue Manager from an OS/2 platform.

In MQSeries, Queue Managers on different platforms are connected together across communication links by 'channels'. A channel is an MQSeries term for a logical link between two Queue Managers.

THE SOLUTION

To resolve the problem we have implemented a Channel Security Exit. There is a Channel Security Exit program at both ends of the channel. These exits get invoked when the channel between the two Queue Managers is started.

An encryption key is used at both ends of the channel and the security information is exchanged between the two Channel Exit programs in order to verify authenticity. The exit might also decide to change the encryption key dynamically, depending on certain circumstances, and reuse this new encryption key. This would involve issuing an MQGET

followed by an MQPUT with the updated key as the key exists in a queue on our MVS/ESA platform.

We decided to keep the encryption key in an MQSeries queue on the MQSeries MVS/ESA platform. We did not want to keep the encryption key in DB2 or VSAM as this would cause problems if these systems were unavailable and it is recommended that I/Os should be avoided.

The *MQSeries Distributed Queuing Guide* manual explains how to write the Channel Security Exit for MQSeries on MVS/ESA. It explains that all MQI calls except MQCMIT/CSQBCMT and MQBACK/CSQBBAK are allowed in the exit program.

THE EXIT PROGRAM

The problem

As mentioned earlier, in certain circumstances, our Channel Security Exit program changes the encryption key. While testing we found that the queue containing the encryption key was not getting updated. We assumed that the MQDISC call would commit all changes made to queues as it does in all other circumstances. Similar exit code worked perfectly OK in batch, but would not work from the Channel Exit.

The manual does not explain that MQDISC within the Channel Security exit will not commit any changes to MQSeries objects. The MQDISC call from within the exit is a dummy call which does not call thread termination for the channel.

The solutions

One solution would be to set our MQGET MESSAGE OPTIONS with MQGMO_NO_SYNCPOINT when we needed to issue a destructive get. This would ensure that the MQGET would operate outside the normal unit of work protocols. The message would be removed immediately and would not be restored by back-out. Similarly, we would have to set our MQPUT MESSAGE OPTIONS with MQPMO_NO_SYNCPOINT when putting the updated message back to the queue. Again, this would operate outside normal unit of work protocols.

However, this was not a satisfactory solution in our circumstances.

Channel Exits should not issue MQCMIT or MQBACK calls as it may interfere with the channel protocols and may lead to possible data loss. However, in the Channel Security Exit, all security exchanges are completed before any application data is sent across the channel.

The alternative solution was to issue a 'commit' at a consistent point within the Channel Security Exit program. We used CSQXCMT when we were ready to commit our changes. CSQXCMT and CSQXBAK are both in the CSQXSTUB which is link-edited with the Channel Exits programs. The format of the CSQXCMT call is similar to other MQSeries calls:

```
*
         CALL  CSQXCMT,(HCONN,COMPCODE,REASON),VL,                +
               MF=(E,PARMLIST)              Commit changes
*
```

This solution has resolved our problem very neatly and provides the security and integrity we require.

Clifton Hunt
Chase – Bournemouth (UK)

MVS news

Platinum Technology Inc has launched Enterprise Tester MVS, an application testing tool incorporating the company's TransCentury Date Simulator. The product simulates live applications with multiple users in batch or on-line. It runs under TSO using ISPF, from where it can log onto CICS, IMS, or other VTAM applications. It has a record facility for capturing keystrokes and generating scripts in REXX or its own language.

For further information contact:
Platinum Technology Inc, 1815 South Meyers Road, Oakbrook Terrace, IL 60181, USA
Tel: (708) 620 5000/(800) 442 6861
Fax: (708) 691 0710 or
Platinum Technology UK Ltd, Turnberry House, 30 Caldecotte Lake Drive, Milton Keynes, Bucks, MK7 8LE, UK
Tel: (01908) 274777
Fax: (01908) 274888.

* * *

EDA S/390 Data Server for Netscape, a product that gives users and JavaScript developers access to enterprise data, has been announced by Information Builders Inc. The product includes a Netscape Enterprise Server and all the required EDA services and appropriate EDA data drivers. It provides location-transparent distributed data access allowing JavaScript applications to read, write, and join data between dissimilar databases and fields located in heterogeneous operating environments.

For further information contact:
Information Builders Inc, 1250 Broadway, New York, NY 10001-3782, USA
Tel: (212) 736 4433/(800) 969 INFO
Fax: (212) 967 6406 or
Information Builders (UK) Ltd, Wembley Point, Harrow Road, Wembley, Middx, HA9 6DE, UK
Tel: (0181)9824700
Fax: (0181) 903 2191.

* * *

Sybase Inc has unveiled four mainframe tools within its Replication Server range to support its WarehouseNOW end-to-end data warehouse product. Distribution Agent for MVS allows users to load up to 32 databases at the same time with a single pass of the data. This can be done by accessing MVS files directly or by using a programming interface to COBOL, Assembler, and C applications. Batch loading features include parallel file transfer, checkpoint/restart, and character translation. Distribution Director is a GUI aimed at helping to coordinate data warehouse activities supporting the definition and execution of LAN tasks, MVS job submissions, and direct connections to the data warehouse. Replication Agent for IMS and Replication Agent for VSAM provide incremental change capture and mapping support, providing the ability to capture non-relational database transactions and replicate (nearly) in real time to over 25 other data targets. A Replication Toolkit for MVS is also available, supporting custom development of change capture models.

For further information contact:
Sybase Inc, 6475 Christie Avenue, Emeryville, CA 94608, USA
Tel: (510) 922 3500/(800)-8-SYBASE
Fax: (510) 658 9441 or
Sybase (UK) Ltd, Sybase Court, Crown Lane, Maidenhead, Berks, RG13 4LW.
Tel: (01628) 597100
Fax: (01628) 597000.

xephon