158 MVS

update

November 1999

In this issue

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

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

Contributions

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

Editor
Jaime Kaminski

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

MVS Update on-line

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Displaying an area in hexadecimal

INTRODUCTION

When programming in Assembler, you sometimes need to know the contents of a given storage location in hexadecimal, for debugging purposes or other reasons. Because of this, I have developed the following macro, which you can include in your program at an appropriate location. The macro has three arguments:

- Firstly, the address of the area to display.
- Secondly, its length. If omitted, four bytes are assumed, but you can easily change this default in the code. The length should be a multiple of four, otherwise a truncation to its nearest multiple will occur.
- Thirdly, also optional, is the keyword ‘BATCH’, which causes the output to be directed to sysprint (or any other DDname of your choice, if you modify the code). This is intended for programs that run in batch. Without this keyword, the output is displayed at a terminal (via TPUT).

This macro can be used more than once in a program, because all generated labels are unique.

```
*====================================================================*
*                                                                    *
*  HEXVIEW - Shows the contents of an area in hexadecimal.           *
*  Place this macro within a program in the desired location.        *
*  All labels generated by this macro start by "H" and are unique.   *
*  Arguments: Address of area, length (should be multiple of 4)      *
*  and, optionally, the keyword "BATCH" to indicate that the output  *
*  should be directed to sysprint instead of the terminal.           *
*  With the BATCH option, R13 must address a standard save area.     *
*  If no length is given, 4 is assumed.                              *
*                                                                    *
*  This macro adds 272 bytes in terminal mode, or 388 bytes in       *
*  batch mode.                                                       *
*                                                                    *
*====================================================================*
*
         MACRO
         HEXVIEW &AREA,&LENG,&PARM
```

```
LENGDEF  EQU   4                   Default length if no leng parm
&DDBATCH SETC  'SYSPRINT'          DDname for BATCH option
&A       SETA  &SYSNDX             Set index for unique labels
         B     HTR&A               Branch around working storage
*
*=========================================================================*
*        HEXVIEW WORKING STORAGE                                          *
*=========================================================================*
*
HREG&A   DS    4F                  Register store area (15,Ø,1,2)
HUNP9&A  DS    ØCL9                Unpack area
HUNP&A   DS    CL8
         DS    CL1
HTR1&A   DC    X'ØFØFØFØFØFØFØFØF'
HTR2&A   DC    C'Ø123456789ABCDEF'
*
*=========================================================================*
*        HEXVIEW MAIN CODE                                                *
*=========================================================================*
*
HTR&A    STM   R15,R2,HREG&A       Store regs 15 thru 2
         STORAGE OBTAIN,           Acquire storage for output line     *
               ADDR=(R2),          and to save remaining regs.         *
               LENGTH=128,                                             *
               LOC=BELOW,                                              *
               COND=YES
         LTR   R15,R15
         BNZ   HEXIT&A
         STM   R3,R14,8Ø(R2)       Store remaining regs
*
         AIF ('&PARM' NE 'BATCH').HSTOR
         B     HOPEN&A
PRINT&A  DCB   DSORG=PS,RECFM=F,MACRF=(PM),                            *
               LRECL=78,BLKSIZE=78,DDNAME=&DDBATCH
HOPEN&A  OPEN  (PRINT&A,OUTPUT)
*
.HSTOR   ANOP
         AIF ('&LENG' NE '').HOKAY
         LA    R8,LENGDEF          If no length given, assume lengdef
         AGO   .HSRL
*
.HOKAY   ANOP
         LA    R8,&LENG            Load length
*
.HSRL    ANOP
         SRL   R8,2                Divide length by four
         LA    R7,8                8 groups of 4 bytes per line
         LA    R9,1                Subtractor for R8
         LA    R6,&AREA            Load address of area to display
*
```

```
HSTART&A EQU   *
         XR    R3,R3               Group counter
         LR    R5,R2               Copy address for output and
         MVI   Ø(R5),X'4Ø'         initialize it with spaces.
         MVC   1(77,R5),Ø(R5)
*
HLOOP&A  EQU   *
         UNPK  HUNP9&A,Ø(5,R6)     Get 4 bytes and turn them into
         NC    HUNP&A,HTR1&A       a viewable stuff.
         TR    HUNP&A,HTR2&A
         MVC   Ø(8,R5),HUNP9&A     Move it to output line
         LA    R3,1(Ø,R3)          Inc group counter
         LA    R6,4(Ø,R6)          Inc input pointer
         LA    R5,1Ø(Ø,R5)         Inc out pointer (plus 2 spaces)
         CR    R3,R7               8 groups attained?
         BE    HDISP&A             Yes, display
         SR    R8,R9               Subtract 1 from R8
         LTR   R8,R8               Length exhausted?
         BH    HLOOP&A             Not yet, continue
*
HDISP&A  EQU   *
         AIF ('&PARM' EQ 'BATCH').HBATCH
         TPUT  Ø(R2),78
         AGO   .HEND
*
.HBATCH  ANOP
         PUT   PRINT&A,Ø(R2),78
*
.HEND    ANOP
HEND1&A  EQU   *
         SR    R8,R9               Subtract 1 from R8
         LTR   R8,R8               Length exhausted?
         BH    HSTART&A            Not yet, continue with a new line.
         LM    R3,R14,8Ø(R2)       Yes, recover regs, free storage
         STORAGE RELEASE,          and exit.                          *
               ADDR=(R2),                                           *
               LENGTH=128,                                          *
               COND=YES
         AIF ('&PARM' NE 'BATCH').EXIT
         CLOSE PRINT&A
*
.EXIT    ANOP
HEXIT&A  EQU   *
         LM    R15,R2,HREG&A
         MEND
*=====================================================================*
*        HEXVIEW END                                                  *
*=====================================================================*
```

Luis Paulo Ribeiro
Systems Engineer
Edinfor (Portugal)

Input tape data from a production environment

THE PROBLEM

In today's automated systems, it is very difficult to pinpoint the input datasets for a workload. This can be complicated further with the addition of an Automated Tape Library to the environment using SMT and OAM with the TCDB catalog. This is where the problems can occur:

- Do you need to shut down the automated tape library for a long period (for technical maintenance or library enlargement for example) and still continue having 24x7 uptime on the batch environment on the outside tape drives?
- Do you need to know which tapes to eject from the automated library and to move a whole production tape workload to another tape library or to the vault?

This occurred at our site when we had to enlarge the single automated tape library. This project took 36 hours during the weekend. We needed to have our batch running with no intervention to our standard 3490Es that were outside the library. The environment was OS/390 Version 2 Release 4, running DFSMSdss Version 1 Release 4.0 with SMT applied. We also had CA-1 Version 5.1, and Library Manager for 3495.

THE SOLUTION

We have developed several REXX procedures to extract the input tape VOLSER from the TMC in the CA-1 by reading the job's JCL from the production JCL library. The steps are as follows:

1 Search *all* datasets in the production JCL library and write them to the 'S038.TAPES.LISTDSB' dataset.

2 Extract only tape datasets from the previous output by running the REXX procedure against the catalog entries.

3 Remove all duplicates from the datastream (using Syncsort Version 3 Release 6).

4 Move the dataset names to CA-1 TMC and check for the specific VOLSER.

5 Create an ‘EJECT’ stream command for the OAM in OS/390 for all required tapes and check again for duplicates.

6 Using the supplied Sync. Routine from CA-1, eject the tapes from the automated tape library and have the CA-1 TMC, TCDB and Library Manager in sync.

Here is the complete batch job:

```
//SØ38TP JOB (SS38,B1,3Ø),SEARCH-FOR-DSN,NOTIFY=SØ38
//*
//* *-------------------------------------------*
//* * EJECT ALL DAILY INPUT TAPES FROM     *
//* * ATL.                                 *
//* *-------------------------------------------*
//* *-------------------------------------------*
//* * STEP NO.1:                           *
//* *     check for ALL DSN in DAILY       *
//* *     Prod plan JCL pds.               *
//* *-------------------------------------------*
//*
//S1      EXEC PGM=ISRSUPC,
//            PARM=(SRCHCMP,
//            'ANYC')
//NEWDD  DD DSN=M1ØØ.JOBD26Ø2,DISP=SHR
//OUTDD  DD DSN=SØ38.TAPES.LISTDSB,DISP=OLD
//SYSIN  DD *
SRCHFOR  'DSN='
/*
//*
//* *-------------------------------------------*
//* * STEP NO.2:                           *
//* *     select TAPES ONLY ds             *
//* *-------------------------------------------*
//*
//S2      EXEC ISPBATCH
//SYSPROC  DD DSN=SØ38.LIB.CNTL,DISP=SHR
//         DD DSN=SYS2.CLIST,DISP=SHR
//SYSTSIN  DD *
  PROFILE NOPREFIX
  ISPSTART CMD(TAPES2) BDISPMAX(99999999)
/*
//*
//* *-------------------------------------------*
//* * STEP NO.3:                           *
//* *     SORT for No Equals               *
//* *-------------------------------------------*
//*
```

```
//S3      EXEC PGM=SORT
//STATOUT   DD SYSOUT=Z
//SORTMSG   DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SORTOUT   DD DSN=SØ38.TAPES,DISP=SHR
//SORTIN    DD DSN=SØ38.TAPES,DISP=SHR
//SYSIN DD *
   SORT FIELDS=(5,44,A),FORMAT=CH
   SUM FIELDS=NONE
   END
/*
//$ORTPARM DD *
 CMP=CLC
//*
//* *-------------------------------------------*
//* * STEP NO.4:                               *
//* *     CHECK CA1 for VOLSERS                *
//* *-------------------------------------------*
//*
//S4      EXEC PGM=TMSBINQ
//SYSOUT   DD SYSOUT=*
//SYSUDUMP DD SYSOUT=Z
//TMSRPT   DD DSN=SØ38.TAPES.CA1.RPT,DISP=SHR
//SYSIN    DD DSN=SØ38.TAPES,DISP=SHR
//*
//* *-------------------------------------------*
//* * STEP NO.5:                               *
//* *     create EJECT command for ATL         *
//* *-------------------------------------------*
//*
//S5       EXEC ISPBATCH
//SYSPROC  DD DSN=SØ38.LIB.CNTL,DISP=SHR
//         DD DSN=SYS2.CLIST,DISP=SHR
//SYSTSIN  DD *
  PROFILE NOPREFIX
  ISPSTART CMD(TAPES4) BDISPMAX(99999999)
/*
//*
//* *-------------------------------------------*
//* * STEP NO.6:                               *
//* *     SORT for No Equals                   *
//* *-------------------------------------------*
//*
//S3      EXEC PGM=SORT
//STATOUT   DD SYSOUT=Z
//SORTMSG   DD SYSOUT=*
//SYSOUT    DD SYSOUT=*
//SORTOUT   DD DSN=SØ38.TAPES.EJECT,DISP=SHR
//SORTIN    DD DSN=SØ38.TAPES.EJECT,DISP=SHR
//SYSIN DD *
   SORT FIELDS=(1,3Ø,A),FORMAT=CH
   SUM FIELDS=NONE
   END
```

```
/*
//$ORTPARM DD *
 CMP=CLC
//*
//*
//* *-------------------------------------------*
//* * STEP NO.7:                                *
//* *     EJECT ALL INPUT TAPES FROM ATL        *
//* *-------------------------------------------*
//*
//EJECT   EXEC  PGM=CTSSYNC,PARM='EJECT'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DSN=SØ38.TAPES.EJECT,DISP=SHR
//*
//
```

TAPES2 REXX

```
/* REXX - LIST TAPE FROM DAILY M1ØØ.JOBD26Ø2   */
TRACE N
  SAY 'Step No. 1 ==> EDIT on SØ38.TAPES.LISTDS'
  ADDRESS ISPEXEC "EDIT DATASET('SØ38.TAPES.LISTDSB') MACRO(TAPEDIT)"
  SAY 'Step No. 2 ==> SELECT TAPES from all jobs'
ADDRESS TSO 'ALLOC FILE(IN) DA(SØ38.TAPES.LISTDSB) SHR'
ADDRESS TSO 'ALLOC FILE(O2) DA(SØ38.TAPES) SHR'
I = Ø
LOOP1:
VOL1='     '
"EXECIO 1 DISKR IN "
IF RC > Ø THEN SIGNAL OUT
  PULL MEM
      PARSE VAR MEM . 'DSN=' DSN ',' .
     DSN = STRIP(DSN,T,' ')
     DSN=SUBSTR(DSN,1,44)
       PARSE VAR DSN FULLDSN .                  /* TO MOVE TO CA1   */
     PARSE VAR DSN DSN '(' .
     PARSE VAR DSN BASEO ' '                    /* TO CHECK IF TAPE */
    X = OUTTRAP(INREC.)                         /* GET INREC IN 'X' */
TRACE
   "LISTC ENT(" || BASEO || ") VOL"
TRACE N
    X = OUTTRAP(OFF)
    IF INREC.Ø = Ø THEN EXIT
    DO COUNT = 1 TO INREC.Ø
       PARSE VAR INREC.COUNT VOL1 .
       IF INDEX(STRIP(VOL1),'VOLSER-') x= Ø THEN DO
           VOLUME = SUBSTR(STRIP(VOL1),19,6)
       END
    END
```

```
  I = I + 1
              IF SUBSTR(VOL1,19,1) = 'Ø' THEN SIGNAL WRTTAPE
              IF SUBSTR(VOL1,19,1) = '1' THEN SIGNAL WRTTAPE
              IF SUBSTR(VOL1,19,1) = '2' THEN SIGNAL WRTTAPE
              IF SUBSTR(VOL1,19,1) = '3' THEN SIGNAL WRTTAPE
  SIGNAL LOOP1
WRTTAPE:
O2.  =
O2.1 = JUSTIFY("DSN="FULLDSN",SHORT",6Ø)
  "EXECIO * DISKW O2 (STEM O2."
  IF RC > Ø THEN DO
            SAY "ERROR WRITE ON DATASET :"
            SIGNAL OUT
            END
SIGNAL LOOP1
OUT:
  "EXECIO Ø DISKR IN (FINIS"
  ADDRESS TSO "FREE F(IN)"
  "EXECIO Ø DISKW O1 (FINIS"
  ADDRESS TSO "FREE F(O1)"
  "EXECIO Ø DISKW O2 (FINIS"
  ADDRESS TSO "FREE F(O2)"
  EXIT
```

TAPEDIT FOR TAPES2

The purpose of this macro is to exclude all unneeded text and datasets from the sysout produced in the previous step, such as output datasets (disp=new/dsn=a.b.c(+1) etc).

```
/* REXX */
ADDRESS ISREDIT "MACRO"
ADDRESS ISREDIT "X 'DSN' ALL"
ADDRESS ISREDIT "DEL NX ALL"
ADDRESS ISREDIT "RESET"
ADDRESS ISREDIT "X '%%' ALL"
ADDRESS ISREDIT "X '-' ALL"
ADDRESS ISREDIT "X '&&' ALL"
ADDRESS ISREDIT "X '//*' ALL"
ADDRESS ISREDIT "X 'SOURCE SECTION' ALL"
ADDRESS ISREDIT "X 'SRCHFOR' ALL"
ADDRESS ISREDIT "X 'SEARCH-FOR' ALL"
ADDRESS ISREDIT "X SYS1 ALL"
ADDRESS ISREDIT "X SYS2 ALL"
ADDRESS ISREDIT "X 'DSN=XXXX' ALL"
ADDRESS ISREDIT "X 'DSN=PDB2' ALL"
ADDRESS ISREDIT "X 'DSN=DB2P' ALL"
ADDRESS ISREDIT "X 'DSN=DB2C' ALL"
ADDRESS ISREDIT "X 'DSN=DSN22Ø' ALL"
ADDRESS ISREDIT "X 'DSN=DSN23Ø' ALL"
```

```
 ADDRESS ISREDIT "X ' EXEC ' ALL"
 ADDRESS ISREDIT "X '(+1' ALL"
 ADDRESS ISREDIT "X '(+Ø1' ALL"
 ADDRESS ISREDIT "X '(+ØØ1' ALL"
 ADDRESS ISREDIT "X '(+2' ALL"
 ADDRESS ISREDIT "X '(+Ø2' ALL"
 ADDRESS ISREDIT "X '(+ØØ2' ALL"
 ADDRESS ISREDIT "DEL X ALL"
 ADDRESS ISREDIT "RESET"
 ADDRESS ISREDIT "C ',DISP=(NEW,CATLG,DELETE)' '' ALL"
 ADDRESS ISREDIT "C ',DISP=(NEW,CATLG)' '' ALL"
 ADDRESS ISREDIT "C ',DISP=OLD' '' ALL"
 ADDRESS ISREDIT "C ',DISP=SHR' '' ALL"
 ADDRESS ISREDIT "C '(,CATLG)' '' ALL"
 ADDRESS ISREDIT "C '  CATLG ' '' ALL"
 ADDRESS ISREDIT "C '  NEW ' '' ALL"
 ADDRESS ISREDIT "C ',DISP=(OLD,KEEP)' '' ALL"
 ADDRESS ISREDIT "C ',' '  ' ALL"
 ADDRESS ISREDIT "C 'SHR' '' ALL"
 ADDRESS ISREDIT "C 'SHR' '' ALL"
 ADDRESS ISREDIT "C 'UNIT=DISK' '' ALL"
 ADDRESS ISREDIT "C 'DCB=BUFNO=11' '' ALL"
 ADDRESS ISREDIT "C 'DCB=BUFNO=12' '' ALL"
 ADDRESS ISREDIT "C 'DCB=BUFNO=13' '' ALL"
 ADDRESS ISREDIT "C 'DCB=BUFNO=14' '' ALL"
 ADDRESS ISREDIT "C 'DCB=BUFNO=15' '' ALL"
 ADDRESS ISREDIT "C ''' '' ALL"
 ADDRESS ISREDIT "SORT NX D"
 ADDRESS ISREDIT "SAVE"
 ADDRESS ISREDIT "END"
 ADDRESS ISREDIT "MEND"
RETURN
```

TAPES4

```
/* REXX - CREATE INPUT FOR EJECT FROM TAPE LIBRARY  */
TRACE N
  SAY 'STEP NO.1 ==> EDIT CA1.RPT'
  ADDRESS ISPEXEC "EDIT DATASET('SØ38.TAPES.CA1.RPT') MACRO(TAPEDIT1)"
  SAY 'STEP NO.2 ==> CREATE EJECT COMMANDS FOR ATL'
ADDRESS TSO 'ALLOC FILE(IN) DA(SØ38.TAPES.CA1.RPT) SHR'
ADDRESS TSO 'ALLOC FILE(O2) DA(SØ38.TAPES.EJECT) SHR'
I = Ø
LOOP1:
VOL1='     '
"EXECIO 1 DISKR IN "
IF RC > Ø THEN SIGNAL OUT
  PULL MEM
      PARSE VAR MEM VOLSER .
      VOLLEN = LENGTH(VOLSER)
           IF VOLLEN > 6 THEN
```

```
              VOLSER = SUBSTR(VOLSER,2,6)
/*    SAY VOLSER   */
      SIGNAL WRITAPE
  I = I + 1
  SIGNAL LOOP1
WRITAPE:
O2.  =
O2.1 = JUSTIFY(" EJECT,"VOLSER,6Ø)
  "EXECIO * DISKW O2 (STEM O2."
  IF RC > Ø THEN DO
             SAY "ERROR WRITE ON DATASET :"
             SIGNAL OUT
             END
SIGNAL LOOP1
OUT:
  "EXECIO Ø DISKR IN (FINIS"
  ADDRESS TSO "FREE F(IN)"
  "EXECIO Ø DISKW O1 (FINIS"
  ADDRESS TSO "FREE F(O1)"
  "EXECIO Ø DISKW O2 (FINIS"
  ADDRESS TSO "FREE F(O2)"
  EXIT
```

TAPES4 EDIT MACRO

The purpose of this step is to leave only VOLSER information in the dataset for creating the 'EJECT' command later on.

```
/* REXX */
ADDRESS  ISREDIT "MACRO"
ADDRESS  ISREDIT "X 'DSN' ALL"
ADDRESS  ISREDIT "DEL NX ALL"
ADDRESS  ISREDIT "RESET"
ADDRESS  ISREDIT "X 'TMS' ALL"
ADDRESS  ISREDIT "DEL X ALL"
ADDRESS  ISREDIT "SORT NX D"
ADDRESS  ISREDIT "SAVE"
ADDRESS  ISREDIT "END"
ADDRESS  ISREDIT "MEND"
RETURN
```

Please note that all the datasets are created with DCB=RECFM=FB,lRECL=80,BLKSIZE=27920 (on 3390-3 device).

SETting Level 88 condition codes on COBOL

THE PROBLEM

Switches and condition codes can be confusing to someone who has to perform maintenance on a program they didn't write. For that matter, it's quite easy to forget what codes you used after a year or two has passed. You could be the one trying to wade through the intricacies of you own code.

Let us suppose we have written a program that calculates sales commissions for sales territories within the United States. Part of the code might look like this:

```
 Ø1  SALES-RECORD.
 Ø5 SALESMAN-ID.
  1Ø SALESMAN-TERRITORY     PIC 99
  1Ø SALESMAN-CODE          PIC 9(5).
.
.
.
 Ø1  WORK-AREAS.
 Ø5 SALES-DISTRICT          PIC XX.
 Ø5 PAY-BY-DISTRICT         PIC X.

PROCEDURE DIVISION.
.
.
.
*        ** Establish district **
IF SALESMAN-TERRITORY < 1Ø
  MOVE 'NW'              TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 2Ø
  MOVE 'SP'              TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 3Ø
  MOVE 'NC'              TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 4Ø
  MOVE 'SW'              TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 5Ø
  MOVE 'MW'              TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 6Ø
```

```
  MOVE 'NE'             TO SALES-DISTRICT
ELSE
IF SALESMAN-TERRITORY < 7Ø
  MOVE 'NA'             TO SALES-DISTRICT
ELSE
  MOVE 'SA'             TO SALES-DISTRICT.
.
.
.
*        ** Calculate sales commission **
IF PAY-BY-DISTRICT         = 'P'
  IF SALES-DISTRICT = 'NW'
    COMPUTE COMMISSION = GROSS-SALES * .1
  ELSE
  IF SALES-DISTRICT = 'SP'
    COMPUTE COMMISSION = GROSS-SALES * .15
  ELSE
  IF SALES-DISTRICT - 'NE'
    COMPUTE COMMISSION = GROSS-SALES * .13
  ELSE
    COMPUTE COMMISSION = GROSS-SALES * .Ø95
  END-IF
  END-IF
  END-IF
ELSE
  COMPUTE COMMISSION = GROSS-SALES * .18
END-IF.
```

This type of programming can be very confusing, especially if the codes are obscure. One way to solve the problem is to comment the code extensively (always a good idea). However, properly coded condition names can also improve this program.

A SOLUTION

Using 88 level data names (condition names) will clear up some of the confusion and make the program easier to understand. One of the verbs that will help us is ‘SET’. The ‘SET’ verb is used for several things:

- Setting indexes
- Increasing/decreasing indexes
- Turning external switches on and off
- Setting conditions
- Setting address pointers (IBM extension to the language).

Our solution will use the fourth option to set conditions. To make our previous code more readable, we can modify it as follows:

```
 Ø1  SALES-RECORD.
 Ø5 SALESMAN-ID.
  1Ø SALESMAN-TERRITORY     PIC 99
  1Ø SALESMAN-CODE          PIC 9(5).
.
.
.
 Ø1      WORK-AREAS.
 Ø5 SALES-DISTRICT          PIC XX.
  88 SALES-DIST-NORTHWEST        VALUE 'NW'.
  88 SALES-DIST-SOUTH-PACIFIC-COAST
                                 VALUE 'SP'.
  88 SALES-DIST-NORTH-CENTRAL    VALUE 'NC'.
  88 SALES-DIST-SOUTH-WEST       VALUE 'SW'.
  88 SALES-DIST-MID-WEST         VALUE 'MW'.
  88 SALES-DIST-NEW-ENGLAND      VALUE 'NE'.
  88 SALES-DIST-NORTH-ATLANTIC   VALUE 'NA'.
  88 SALES-DIST-SOUTH-ATLANTIC   VALUE 'SA'.
```

Notice that we used condition names that indicate which variable they belong to as well as explaining what the code means:

```
Ø5 PAY-BY-DISTRICT-CODE     PIC X.
  88 PAY-BY-DISTRICT             VALUE 'P'.
  88 DO-NOT-PAY-BY-DISTRICT      VALUE 'N'.

PROCEDURE DIVISION.
.
.
.
*        ** Establish district **
IF SALESMAN-TERRITORY < 1Ø
  SET SALES-DIST-NORTHWEST       TO TRUE
*        ** "SET" moves the value of the condition **
*        ** name to its parent variable. In this   **
*        ** case, it will move "NW" to              **
*        ** SALES-DISTRICT.                         **
ELSE
IF SALESMAN-TERRITORY < 2Ø
  SET SALES-DIST-SOUTH-PACIFIC-COAST TO TRUE
ELSE
IF SALESMAN-TERRITORY < 3Ø
  SET SALES-DIST-NORTH-CENTRAL   TO TRUE
ELSE
IF SALESMAN-TERRITORY < 4Ø
  SET SALES-DIST-SOUTH-WEST      TO TRUE
ELSE
IF SALESMAN-TERRITORY < 5Ø
  SET SALES-DIST-MID-WEST        TO TRUE
```

```
ELSE
IF SALESMAN-TERRITORY < 60
  SET SALES-DIST-NEW-ENGLAND            TO TRUE
ELSE
IF SALESMAN-TERRITORY < 70
  SET SALES-DIST-NORTH-ATLANTIC  TO TRUE
ELSE
  SET SALES-DIST-SOUTH-ATLANTIC  TO TRUE.
.
.
.
*        ** Calculate sales commission **
IF PAY-BY-DISTRICT
  IF SALES-DIST-NORTHWEST
    COMPUTE COMMISSION = GROSS-SALES * .1
  ELSE
  IF SALES-DIST-SOUTH-PACIFIC-COAST
    COMPUTE COMMISSION = GROSS-SALES * .15
  ELSE
  IF SALES-DIST-NEW-ENGLAND
    COMPUTE COMMISSION = GROSS-SALES * .13
  ELSE
    COMPUTE COMMISSION = GROSS-SALES * .095
  END-IF
  END-IF
  END-IF
ELSE
  COMPUTE COMMISSION = GROSS-SALES * .18
END-IF.
```

Now our program becomes self-documenting. A maintenance programmer does not have to know the actual codes to understand the business logic of the program. If (s)he has to make a change to the commission rate for New England, it is immediately obvious where the change needs to take place.

This is not the best solution, but I used it to illustrate the ‘SET’ verb. If the condition codes (level 88) had been defines on SALESMAN-TERRITORY, the entire section of ‘SET’s could be dropped, as well as the variable SALES-DISTRICT in the working storage area. This could have been done like this:

```
05 SALESMAN-ID.
  10 SALESMAN-TERRITORY           PIC 99
  10 SALESMAN-CODE                PIC 9(5).
  88 SALES-DIST-NORTHWEST     VALUE 01 THRU 09.
  88 SALES-DIST-SOUTH-PACIFIC-COAST
                              VALUE 10 THRU 19.
  88 SALES-DIST-NORTH-CENTRAL VALUE 20 THRU 29.
  88 SALES-DIST-SOUTH-WEST    VALUE 30 THRU 39.
```

```
88 SALES-DIST-MID-WEST      VALUE 40 THRU 49.
88 SALES-DIST-NEW-ENGLAND   VALUE 50 THRU 59.
88 SALES-DIST-NORTH-ATLANTIC
                            VALUE 60 THRU 69.
88 SALES-DIST-SOUTH-ATLANTIC
                            VALUE 70 THRU 99.
```

Notice that ranges work just as well as a single literal. If you need discrete literals, you can use more than one literal, such as:

```
10 SALESMAN-CODE                PIC 9(5).
88 SALES-DIST-NORTHWEST    VALUE 01 02 05 07.
88 SALES-DIST-SOUTH-PACIFIC-COAST
                           VALUE 10 11 15 18.
 05 PRODUCT-SOLD                PIC X(10).
   88 TOWER-CASE                VALUE 'TWR       '
                                      'TWR CASE  '
                                      'TOWER CASE'
                                      'TC        '.
   88 KEYTRONICS-KEYBOARD       VALUE 'KEYTRONICS'
                                      'KT KEYBD  '
                                      'KEY T KB  '.
```

A better solution for our rate calculation might involve a rate table, but this is beyond the scope of this article.

Alan Kalar
Systems Programmer (USA)

Library Search Facility

INTRODUCTION

It is sometimes useful to have a search utility that is able to search any of the standard concatenations of libraries. I have written a utility in REXX, called the Library Search Facility (LSF). This allows the searching of any of the given standard concatenations, or sets of concatenations, of PDS libraries for a specified member.

Not only does LSF search the standard concatenations, it can also search *ad hoc* user-defined concatenations of PDS libraries. LSF lists all duplicate occurrences of the given member in the different libraries.

LSF CONCATENATIONS

LSF allows the following concatenations to be searched:

1 ISPLLIB

2 STEPLIB

3 LPA

4 LNK (linklist)

5 STEPLIB

6 LOAD (all of the above, 1-5)

7 SYSEXEC

8 SYSPROC

9 CMDS (both 7 and 8 above)

10 ISPPLIB

11 ISPMLIB

12 ISPSLIB

13 ISPTLIB

14 ISPTABL

15 ISPF (all of the above, 10-14)

16 ds_list (name of member in USERLIB library specified in the REXX code).

The ds_list in item 16 above is a member of the PDS assigned to the USERLIB variable in the REXX code for LSF (this defaults to userid.SPFLIB.CNTL – it is assumed that the TSO PROFILE PERFIX is set to the ID of the TSO user). This ds_list member contains one PDS dataset name per line, eg:

```
********** TOP OF DATA **********
SYS1.PROCLIB
SYS2.PROCLIB
OPC.PROCLIB
USER.JOBLIB
******** BOTTOM OF DATA *********
```

To execute the LSF utility type in the command from the TSO READY prompt (prefixing the command name with TSO). Thirdly, the ISPF Command Table, ISPCMDS, could be updated to include 'LSF' as an valid command). The syntax is as follows:

```
LSF member L(ds_list|LNK) OPT(T|V)
```

Where:

- member is the name of the member to search for
- ds_list is the dataset list to search. The default is LNK (linklist concatenation). The full list of possible values is shown above (items 1 to 16).
- T – TERSE (list only libraries that contain the specified member) this is the default.
- V – VERBOSE (list all libraries searched).

The 'L(...)' and 'OPT(…)' are optional parameters, and could be omitted – in which case the defaults mentioned above would apply.

Sample output, for various invocations of LSF, follows:

```
LSF SORT L(LOAD) OPT(V)
_____________________________________

Searching for member, SORT in the load concatenation...
Found in: TSG.SYNCSORT.V364.LOAD (LINKLIST)
SORT found in 1 out of 20 libraries.
_____________________________________


LSF SORT L(LOAD) OPT(V)
_____________________________________

Searching for member, SORT in the load concatenation...
       ZZ272Z.SPF.LOAD       (ISPLLIB)
       SYS1.LINKLIB          (LINKLIST)
       SYS1.MIGLIB           (LINKLIST)
       SYS1.CMDLIB           (LINKLIST)
       SYS1.CSSLIB           (LINKLIST)
       ISR.V3R5M0.ISRLOAD    (LINKLIST)
       ISP.V3R5M0.ISPLOAD    (LINKLIST)
       ISF.V1R3M3.ISFLOAD    (LINKLIST)
       GIM.SGIMLMD0          (LINKLIST)
       SYS1.DGTLLIB          (LINKLIST)
       SYS1.DFQLLIB          (LINKLIST)
Found in: TSG.SYNCSORT.V364.LOAD (LINKLIST)
       SYS1.GDDMLOAD         (LINKLIST)
```

```
       SYSA.LINKLIB          (LINKLIST)
       SYSA.CSLIB1           (LINKLIST)
       SYS1.LPALIB           (LPA LIST)
       ISR.V3R5M0.ISRLPA     (LPA LIST)
       ISP.V3R5M0.ISPLPA     (LPA LIST)
       ISF.V1R3M3.ISFLPA     (LPA LIST)
       TSG.SYNCSORT.V364.LPALIB  (LPA LIST)
SORT found in 1 out of 20 libraries.
```

The LSF utility uses a number of internal subroutines to locate the system concatenations of the LPA and linklist, as well as to determine the datasets allocated to specified (STEPLIB, ISPPLIB,…etc) DDnames. The LSF utility has been built using code that I have created over the years, as parts of other utilities, and found very useful. The source for LSF follows:

```
/*———————————— REXX ————————————————————————————————————————*/
/* Function   : Search specified list for all occurrences of the     */
/*              given member.                                        */
/* Syntax     : LSF member L(ds_list|LNK) OPT(T|V)                   */
/*                                 —      -                         */
/* Where      : member - is member to search for                     */
/*              ds_list- is the dataset list to search               */
/*                       Possible values:                            */
/*                         1. ISPLLIB                                */
/*                         2. STEPLIB                                */
/*                         3. LPA                                    */
/*                         4. LNK     (linklist)                     */
/*                         5. STEPLIB                                */
/*                         6. LOAD    (All of the above, 1-5)        */
/*                         7. SYSEXEC                                */
/*                         8. SYSPROC                                */
/*                         9. CMDS    (Both 7 and 8 above)           */
/*                        10. ISPPLIB                                */
/*                        11. ISPMLIB                                */
/*                        12. ISPSLIB                                */
/*                        13. ISPTLIB                                */
/*                        14. ISPTABL                                */
/*                        15. ISPF    (All of the above, 10-14)      */
/*                        16. ds_list (list member_name in USERLIB   */
/*                                     library specified below)      */
/*                       The default is LLS (linklist concatenation) */
/*             T      - Terse (list only libraries that contain     */
/*                             specified member) This is default.    */
/*             V      - Verbose (list all libraries searched).      */
/*————————————————————————————————————————————————————————————————*/
arg mem parms
numeric digits 8
opt = ''; lst = ''
x = pos('OPT(',parms)
```

```
if x > Ø then parse value parms with . 'OPT(' opt ')' .
x = pos('L(',parms)
if x > Ø then parse value parms with . 'L(' lst ')' .
if lst = '' then lst = 'LNK'
if opt ¬= 'V' then opt = 'T'
/* Name of library containing the ds_list member */
                        /* TSO prefix will be automatically appended */
userlib = 'SPFLIB.CNTL'
if mem = '' then
  do
    say 'You MUST supply a member to search for?'
    say
    say 'Syntax: LSF member L(ds_list|LNK) OPT(T|V)'
    say '                              —      -   '
    say 'Where : member - is member to search for       '
    say '        ds_list- is the dataset list to search          '
    say '         Possible values:                               '
    say '           1. ISPLLIB                                   '
    say '           2. STEPLIB                                   '
    say '           3. LPA                                       '
    say '           4. LNK     (linklist)                        '
    say '           5. STEPLIB                                   '
    say '           6. LOAD    (All of the above, 1-5)           '
    say '           7. SYSEXEC                                   '
    say '           8. SYSPROC                                   '
    say '           9. CMDS    (Both 7 and 8 above)              '
    say '          1Ø. ISPPLIB                                   '
    say '          11. ISPMLIB                                   '
    say '          12. ISPSLIB                                   '
    say '          13. ISPTLIB                                   '
    say '          14. ISPTABL                                   '
    say '          15. ISPF    (All of the above, 1Ø-14)         '
    say '          16. ds_list (list member_name in USERLIB      '
    say '                       library specified below)         '
    say '         The default is LLS (linklist concatenation)    '
    say 'T      - Terse (list only libraries that contain        '
    say '                specified member) This is default.      '
    say 'V      - Verbose (list all libraries searched).         '
    exit
  end
Select
   when lst = 'ISPLLIB' then
     do
       x = LDD(lst)
       /* On return, if x = Ø, lls. array contains DSN list */
       if x > Ø then
         do
            say 'DDname: 'lst' not allocated?'
            exit
         end
     end
```

```
when lst = 'STEPLIB' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'LNK' then call GETLNK
when lst = 'LPA' then call GETLPA
when lst = 'STEPLIB' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'LOAD' then
  do
    t = Ø
    tmp. = ''
    x = LDD('ISPLLIB')
    if x = Ø then
      do i = 1 to lls.Ø
         t = t + 1
         tmp.t = lls.i
      end
    x = LDD('STEPLIB')
    if x = Ø then
      do i = 1 to lls.Ø
         t = t + 1
         tmp.t = lls.i
      end
    call GETLNK             /* Get the Linklist DS list */
    do i = 1 to lls.Ø
       t = t + 1
       tmp.t = lls.i
    end
    call GETLPA             /* Get the LPA DS list */
    do i = 1 to lls.Ø
       t = t + 1
       tmp.t = lls.i
    end
    tmp.Ø = t
    drop lls.
    do i = 1 to tmp.Ø
      lls.i = tmp.i         /* Get full list of DS's to search */
```

```
      end
      lls.Ø = tmp.Ø
      drop tmp.
    end
  when lst = 'SYSEXEC' then
    do
      x = LDD(lst)
      /* On return, if x = Ø, lls. array contains DSN list */
      if x > Ø then
        do
           say 'DDname: 'lst' not allocated?'
           exit
        end
    end
  when lst = 'SYSPROC' then
    do
      x = LDD(lst)
      /* On return, if x = Ø, lls. array contains DSN list */
      if x > Ø then
        do
           say 'DDname: 'lst' not allocated?'
           exit
        end
    end
  when lst = 'CMDS' then
    do
      t = Ø
      tmp. = ''
      x = LDD('SYSEXEC')
      if x = Ø then
        do i = 1 to lls.Ø
           t = t + 1
           tmp.t = lls.i
        end
      x = LDD('SYSPROC')
      if x = Ø then
        do i = 1 to lls.Ø
           t = t + 1
           tmp.t = lls.i
        end
      tmp.Ø = t
      drop lls.
      do i = 1 to tmp.Ø
        lls.i = tmp.i          /* Get full list of DSs to search */
      end
      lls.Ø = tmp.Ø
      drop tmp.
    end
  when lst = 'ISPPLIB' then
    do
      x = LDD(lst)
      /* On return, if x = Ø, lls. array contains DSN list */
```

```
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'ISPMLIB' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'ISPSLIB' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'ISPTLIB' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'ISPTABL' then
  do
    x = LDD(lst)
    /* On return, if x = Ø, lls. array contains DSN list */
    if x > Ø then
      do
         say 'DDname: 'lst' not allocated?'
         exit
      end
  end
when lst = 'ISPF' then
  do
    t = Ø
    tmp. = ''
    x = LDD('ISPPLIB')
    if x = Ø then
      do i = 1 to lls.Ø
```

```
            t = t + 1
            tmp.t = lls.i
          end
        x = LDD('ISPMLIB')
        if x = Ø then
          do i = 1 to lls.Ø
            t = t + 1
            tmp.t = lls.i
          end
        x = LDD('ISPSLIB')
        if x = Ø then
          do i = 1 to lls.Ø
            t = t + 1
            tmp.t = lls.i
          end
        x = LDD('ISPTLIB')
        if x = Ø then
          do i = 1 to lls.Ø
            t = t + 1
            tmp.t = lls.i
          end
        x = LDD('ISPTABL')
        if x = Ø then
          do i = 1 to lls.Ø
            t = t + 1
            tmp.t = lls.i
          end
        tmp.Ø = t
        drop lls.
        do i = 1 to tmp.Ø
          lls.i = tmp.i          /* Get full list of DSs to search */
        end
        lls.Ø = tmp.Ø
        drop tmp.
      end
    otherwise ,     /* Get list of DSNs to search from USERLIB */
      do
        lst = userlib"("lst")"
        address tso "ALLOC FI(lst) DA("lst") SHR"
        retc = RC
        if retc ^= Ø then
          do
            say 'Allocate for DSN list failed RC='retc'.'
            say 'LS - abended.'
            exit
          end
        "EXECIO * DISKR lst (STEM lls. FINIS)"
        address tso "FREE FI(LST)"
      end
end
n = Ø
Say 'Searching for member, 'mem' in the 'lst' concatenation...'
do i = 1 to lls.Ø
```

```
   lls_dsn = word(lls.i,1)
   dsn = "'"lls_dsn"("mem")'"
   if SYSDSN(dsn) = 'OK' then
     do
       n = n + 1
       say 'Found in: 'strip(lls.i)
     end
   else ,
     if opt = 'V' then say '          'strip(lls.i)
end
say mem' found in 'n' out of 'lls.Ø' libraries.'
exit
```

LDD

```
/*———————————— REXX ————————————————————————————————————————*/
/* Function   : List datasets currently allocated to a given ddname  */
/* Syntax     : LDD ddname                                           */
/*———————————————————————————————————————————————————————————————*/
ARG ddname .
null = OUTTRAP('ddlist.')
"LISTA STATUS"

do i = 1 to ddlist.Ø
   if substr(ddlist.i,1,2) = '  ' & words(ddlist.i) = 2 & ,
     strip(word(ddlist.i,1)) = ddname then
       leave i
   else nop
end
if i > ddlist.Ø then return 8
/*———————————————————————————————————————————————————————————————*/
/* if the name is found, look for the next name between which are*/
/* listed the datasets allocated to the specified ddname             */
/*———————————————————————————————————————————————————————————————*/
do j = i+1 to ddlist.Ø
   if substr(ddlist.j,1,2) = '  ' & words(ddlist.j) = 2 then
     leave j
   else nop
end
if j > ddlist.Ø then j = j + 1
cnt = (j - i)/2
/*———————————————————————————————————————————————————————————————*/
/* copy DSNs allocated to name to lls. array                        */
/*———————————————————————————————————————————————————————————————*/
c = Ø
do k = i-1 to j-2 by 2
   c = c + 1
   lls.c = substr(ddlist.k,1,44)||' ('ddname')'
end
lls.Ø = c
return 0
```

GETLNK

```
/*———————————— REXX ————————————————————————————————————————*/
/* Function   : Get list of datasets allocated to linklist          */
/* Syntax     : GETLNK                                              */
/*———————————————————————————————————————————————————————————————*/
ARG .
/* System related information from CVT etc. */
cvt = storage(10,4)
/* pointer to llt */
cvtlltp   = storage(d2x(c2d(cvt) + 1244),4)
/* point past info */
lltp   = d2x(c2d(cvtlltp) + c2d('8'x))

i = 0
do forever
   if storage(lltp,1) = '80'x then leave
   i = i + 1
   lls.i = storage(d2x(x2d(lltp) + 1),44)||' (Linklist)'
   lltp = d2x(x2d(lltp) + 45)
end
lls.0 = i
return
```

GETLPA

```
/*———————————— REXX ————————————————————————————————————————*/
/* Function   : Get list of datasets allocated to LPA list          */
/* Syntax     : GETLPA                                              */
/*———————————————————————————————————————————————————————————————*/
ARG .
cvt = storage(10,4)
/* pointer to smext*/
cvtsmext  = storage(d2x(c2d(cvt) + 1196),4)
/* point to lpat */
lpatp  = storage(d2x(c2d(cvtsmext) + 56),4)
/* point past info */
lpat   = d2x(c2d(lpatp) + c2d('8'x))
i = 0
do forever
   if storage(lpat,1) = '00'x then leave
   i = i + 1
   lls.i = storage(d2x(x2d(lpat) + 1),44)||' (LPA list)'
   lpat = d2x(x2d(lpat) + 45)
end
lls.0 = i
return
```

Ghias Din
Systems Programmer

Turn off SMF wait timing

In most installations, TSO users have a timeout limit based on SMF values. If your user-id remains idle for a length of time, you will be cancelled. This program prevents you from being automatically logged off. The program evaluates the sysid and your TSO-id prefix. Not all users can execute this program. If your TSO-id prefix 'First char' is not matched in the table, program access will be denied.

To install the program first assemble and link the program. Add the program to the IKJTSOxx member of SYS1.PARMLIB. The program must be APF authorized. The invocation method is TSO IEETME.

```
IEETME   TITLE 'TURN OFF SMF WAIT TIMING'
*+IEETME       TURN OFF SMF WAIT TIMING
*        USES SVC 233 TO BECOME AUTHORIZED.(SPFCOPY SVC)
*        SETS ASCBRCTF TO X'Ø1' DONT TIME
*        THIS NEGATES SMF TIMING - SAME AS TIME=144Ø IN JCL
*        AUTHORIZE THRU TSO TABLE IKJTSOØØ
*        ADDED DATE AND TIME TO MESSAGE
         REGS
IEETME   CSECT
         SAVE  (14,12),,IEETME_&SYSDATE._&SYSTIME
         LR    R12,R15                 SET PROGRAM ADDRESSABILITY
         USING IEETME,R12
         LA    R15,SAVEAREA            LOAD SAVE AREA ADDRESS
         ST    R15,8(R13)              STORE IN CALLERS SAVEAREA
         ST    R13,SAVEAREA+4          STORE SA POINTER IN OUR SA
         LR    R13,R15                 POINT TO MY SAVEAREA
* GET TSO USERID
         L     R3,16                   POINTER TO CVT
         L     R4,Ø(R3)                CVT TCB POINTER
         L     R4,4(R4)                POINTER TO TCB
         L     R4,12(R4)               POINTER TO TIOT
         MVC   USER,Ø(R4)              MOVE USERID TO MESSAGE
* GET SMFID
         USING CVT,R3                  COVER CVT DSECT
         USING SMCABASE,R5             COVER SMCA
         L     R5,CVTSMCA              LOAD SMCA ADDRESS
         MVC   SYSID,SMCASID           SYSTEM ID
         DROP  R5
         DROP  R3
* Evaluate the sysid by obtaining the fourth digit
         CLI   SYSID+4,C'A'            IS IT SYSA
         BE    TEST
         CLI   SYSID+4,C'B'            IS IT SYSB
         BE    USEROK                  BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'C'            IS IT SYSC
```

```
         BE    TEST
         CLI   SYSID+4,C'D'             IS IT SYSD
         BE    USEROK                   BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'E'             IS IT SYSE
         BE    TEST
         CLI   SYSID+4,C'F'             CHECK SYSID
         BE    USEROK                   BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'G'             CHECK SYSID
         BE    USEROK                   BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'H'             CHECK SYSID
         BE    USEROK                   BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'I'             CHECK SYSID
         BE    USEROK                   BYPASS CHECK FOR Y2K
         CLI   SYSID+4,C'J'             CHECK SYSID
         BE    USEROK                   BYPASS CHECK FOR Y2K
TEST     DS    ØH
         L     R2,Ø(R1)                 LOAD PTR TO PARM
         LH    R3,2(R2)                 LOAD OFFSET TO DATA
         LA    R3,4(R3)                 ADD PREFIX LENGTH
         AR    R2,R3                    R2-->DATA
         MVC   REQ,Ø(R2)                SAVE REQUEST
         OI    REQ,X'4Ø'                UPPERCASE REQ
         CLI   REQ,C'R'                 REPORT REQUESTED?
         BE    REPORT                   DO REPORT
         CLI   REQ,C'F'                 TURN OFF
         BNE   NOPARM                   NOT OFF ASSUME ON
         OI    SWITCH,X'Ø1'             SET OFF INDICATOR
NOPARM   DS    ØH                       NO PARM PRESENT
         TESTAUTH FCTN=1
         LTR   R15,R15
         BNZ   NOK
*        B     USEROK             +++++BYPASS USERID CHECK
*---------------------------------------------------------------
*        EXCLUDE ALL 'H' AND 'U' USERS
*        USE EXCLUDE TABLE FOR 'T' AND 'O' USERS
*---------------------------------------------------------------
         CLI   USER,C'O'
         BE    CHECK_USER               CHECK FOR VALID O USERS
         CLI   USER,C'U'                USER?
         BE    REJECT_USER
         CLI   USER,C'H'                USER?
         BE    REJECT_USER
         CLC   USER(2),=C'TC'           CONTRACTOR
         BE    REJECT_USER
         CLI   USER,C'T'                USER?
         BE    REJECT_USER
         B     USEROK                   ACCOUNT FOR VRU SESSIONS
REJECT_USER DS ØH
         LA    R15,VALID_USER_TABLE
RJ_LOOP  CLC   USER(4),Ø(R15)           CHECK FOR VALID USER
         BE    USEROK                   FOUND VALID USER
         LA    R15,4(R15)               NEXT TABLE ENTRY
         CLI   Ø(R15),X'FF'             END OF TABLE
```

```
         BNE   RJ_LOOP                 NO - CHECK NEXT
         MVI   WTO1+4+22,C'X'          INDICATE REJECTED USER
         B     SEND_MSG
VALID_USER_TABLE DS ØH
         DC    C'TD9 ',C'TO6Ø','P  P'
         DC    C'TBØ '
         DC    X'FF',C'END'            END OF TABLE
CHECK_USER DS  ØH                      VERIFY EXCLUDED USERS
         LA    R15,TABLE               POINT AT AUTH USER TABLE
CK_LOOP  CLI   Ø(R15),X'FF'            END OF TABLE?
         BE    USEROK                  VALID USER
         CLC   USER(4),Ø(R15)          CHECK ENTRY
         BE    REJECT_USER             REJECTED USER
         LA    R15,4(R15)              POINT AT NEXT ENTRY
         B     CK_LOOP
USEROK   DS    ØH
         MODESET KEY=ZERO
         L     R4,X'224'               POINT TO CURRENT ASCB
         TM    SWITCH,X'Ø1'            SET OFF?
         BO    SET_OFF                 YES
         OI    X'66'(R4),X'Ø1'         SET NO SMF TIMING
         B     SET_EXIT
SET_OFF  NI    X'66'(R4),X'FE'         SET SMF TIMING ON
         MVC   MSG1R,=C'RESET'         INDICATE TIMER RESET
SET_EXIT DS    ØH
         MODESET KEY=NZERO
         CLC   USER(3),=C'O6Ø'
         BE    RETURN
SEND_MSG DS    ØH
         TIME  DEC
         ST    RØ,TIME
         ST    R1,DATE
         UNPK  MDATE,DATE+1(3)
         OI    MDATE+4,X'FØ'           SIGN
         OI    MTIME,X'ØF'             MAKE VALID SIGN
         UNPK  DOUBLE,TIME
         MVC   MTIME,DOUBLE+1          MOVE TO MSG
         WTO   MF=(E,WTO1)
         TPUT  MSG1,2Ø
RETURN   L     R13,SAVEAREA+4
         LH    R15,RCODE               LOAD RETURN CODE
         RETURN (14,12),RC=(15)
NOK      TPUT  =CL2Ø'NOT AUTH',2Ø
         B     RETURN
REPORT   DS    ØH
         L     R4,X'224'               POINT AT CURRECT ASCB
         TM    X'66'(R4),X'Ø1'         TEST IF TIMING ON
         BO    RPT_ON                  IS ON
         MVC   MSG2V,=C'OFF'           NOT ON
         B     RPT_MSG
RPT_ON   MVC   MSG2V,=C'ON '           INDICATE ON
RPT_MSG  TPUT  IEETME+5,21
         MVC   MSG2SID,SYSID           DISPLAY SYSID
```

```
         TPUT  MSG2,20
         B     RETURN                  EXIT
DOUBLE   DC    D'0'
SAVEAREA DS    18F
DATE     DC    F'0'
TIME     DC    F'0'
RCODE    DC    H'0'                    RETURN CODE
SWITCH   DC    X'00'
*        DC    X'01'                   TURN OFF TIMER
         DS    0F
REQ      DC    C' '
SYSID    DC    CL4' '
TABLE    DS    0H                      EXCLUDED USERS
         DC    C'TO60'
         DC    C'T77 ',C'T2T ',C'    '
         DC    C'TD9 '
         DC    C'TB0 '
         DC    X'FF',C'END'            END OF TABLE
*               0----+----1----+----2----+----3----+----4----+----5--
WTO1     WTO   'XXY321 UUUU INVOKED TMR FACILITY D(00000) T(000000)',  X
               ROUTCDE=(7),DESC=(6),MF=L
USER     EQU   WTO1+4+7,4
MDATE    EQU   WTO1+4+35,5
MTIME    EQU   WTO1+4+44,6
MSG1     DC    CL20'TIMER SET           '
MSG1R    EQU   MSG1+6,5
MSG2     DC    CL20'TIMER IS XXX        '
MSG2V    EQU   MSG2+09,3
MSG2SID  EQU   MSG2+16,4
         LTORG
*NOTES
* ASVT+200 HAS EYECATCHER 'ASVT' -  ACTUALLY THE START OF THE TABLE
* ASVT+210 STARTS THE VECTOR TABLE ASCBS START AT ASCB(1)
*
*-AN AVAILABLE ASVT ENTRY ISX'80AAAAAA' WHERE AAAAAA IS THE NEXT
*    ASVT ENTRY ADDRESS. NO ACTUAL ASCB EXISTS.
*
* IEF352I ADDRESS SPACE UNAVAILABLE
*
*-AN ASCB MARKED UNAVAILABLE WILL HAVE THE AVAILABLE BIT ON X'80'
*    AND THE ADDRESS WILL POINT AT THE THE ASVT+X'1E0'
*    EX:  ASVT+X'1E0'= 00F93E88
*         X'80F93E88' IS AN UNUSEABLE ASCB
*    THE ADDRESS SPACE IS UNAVAILABLE BECAUSE RESIDUAL CROSS MEMORY
*    POINTERS REMAIN IN MEMORY.
*-AN ASCB WITH JUNK IN THE JOBNAME IS MOST LIKELY 'STARTING'
         PRINT NOGEN
         CVT   DSECT=YES               CVT
         IEESMCA                       SMCA
         END   IEETME
```

MVS system monitor

INTRODUCTION

TASID is an MVS system monitor, which runs in TSO/ISPF on OS/390. TASID is an IBM 'internally developed tool' that is available to download from the Internet. It is not officially supported by IBM, only provided on an 'as is' basis. The author is Doug Nadel from the ISPF development team at IBM, and he provides a new release every few months at irregular intervals.

FUNCTIONALITY

It can display information about many things, including:

- System configuration.
- What is running.
- ENQs.
- Initiator status (JES2 only).
- Linklist libraries, LPA libraries, APF libraries, PARMLIBs.
- DASD space, active devices, available units.
- SVC list, LPARs, subsystems, link pack directory, nucleus map.
- Your dataset allocations (an enhanced version of the ISRDDN program).
- Display storage (that is not fetch protected).

HOW TO GET IT

You can get TASID from the URL, www.software.ibm.com/ad/ispf/downloads, by clicking on a 'download' button. I would suggest getting the tasid.exe and tasid.zip files. Then you run tasid.exe and it creates tasid.xmi and tasidp.xmi plus a readme file.

PANELS

It has many HELP panels; these provide the only available documentation. The first panel in TASID looks like this:

```
   File  Navigate  Settings
 ------------------------------------------------------------------------------
                                TASID option menu              Limited ENQ
data
 Option  ===>

    Select one of the following options:                    Version 5.05k
        1 - Address space list        5 - Miscellaneous displays
        2 - System ENQ contention     6 - Current dataset allocations
        3 - Total system ENQ status   7 - Storage View Facility
                                      8 - Snapshot

   +----------------------------------------+-----------------------------+
   !   Current time  15:47 on 1999/06/10    !   TSO users          271    !
   !   Last IPL time 20:15 on 1999/05/08    !   Started tasks      112    !
   !   IPL Parameters 1606 2B   1           !   Jobs                26    !
   !   OS/390   02.05.00 JES version JES3   !   System addrs        25    !
   !   SMF ID   MVSB     JES level   2.5.0  !   Free initiators      0    !
   !   User ID  XV88653  RACF level  2.4.0  !-----------------------------!
   !   Node     CPXMVS   TSO version 2.6.0  !   Total              434    !
   !   VTAM Adr XV88653  VTAM Level  4.4    !-----------------------------!
   !   Proc     LOGONRUV DFSMS level 1.4.0  !   CPU utilization 94%       !1
   !   Region   8172K    DSS Level   1.4.0  !   CPU 9672-RY5   (10 CPUs)  !
   !   RACF Grp $ZISBST  DSF  level  1.16.  !   ENQ Contention  None      !
   !   Mode     PR/SM    ISPF Level  4.5.0  !   Real Storage    1572864K  !
   !   LPARs    6        HSM Level   1.4.0  !   Expand Storage  524,288K  !
   !----------------------------------------+-----------------------------!
   !   MVS Information: OS/390 02.05.00                                    !
   !   JES Information: JES3 / OS 2.5.0 / HJS6605                          !
   !   Sysres: RSSEA1     System: MVSB       PLEX: CPXMVS                  !
   !--------------------------------------------------------------------- !
   !   This system keeps a history of 3 passwords.                         !
   !   Automatic revocation after 5 invalid logon attempts.                !
   !   Password warning is 4 days before password expires.                 !
   !   Revocation for inactivity is not in effect.                         !
   !   RACF program control is available.                                  !
   !--------------------------------------------------------------------- !
   !   SMS is available with PDSE support.                                 !
   !   TASID 5.05k - Compiled at 19.06 on 04/01/99                         !
   +---------------------------------------------------------------------- +
 (c) Copyright IBM Corp, 1993, 1998. All Rights Reserved.
```

INSTALLATION

The readme file gives instructions to copy the two xmi files to your host then use TSO RECEIVE to create a load library (with only one load module) and an ISPF panel library. Allocate these libraries to your TSO session and you can use it immediately, via the command TSOTASID. There is nothing more to do.

CONCLUSION

The information provided is well presented on all of the panels, and usage is fairly intuitive (unlike some commercial products that I have used).

TASID does not run authorized so there are limits to what it can do. It is not as comprehensive as some of the commercial products, but it can give you quite a lot of useful system information quickly and easily. It is well worth a look.

Ron Brown
Systems Programmer (Germany)

Catalog clean-up for disaster recovery testing

THE PROBLEM

We have had a contract for Disaster Recovery (DR) services for many years with a prominent DR vendor. The set-up of our disaster recovery testing requires that back-ups of our system volumes (and their associated ICF catalogs) be taken on a weekly basis and used at the DR site. In order to run our production batch cycle at the DR site, all of the catalog entries for datasets residing on public DASD volumes need to be removed. Many years ago we modified some public domain code from a public domain MVS software collection (the CBT tape) that would read each ICF catalog as a dataset searching for entries that matched our criteria in order to generate IDCAMS

DELETE control cards. In our most recent DR test, we noticed a problem in the program that had apparently been there all along but only just surfaced. Faced with the task of having to perform major changes to the program in order to fix the problem, we decided to investigate other methods for generating the required DELETE statements.

A SOLUTION

I was aware of the Catalog Search interface (CSI) that became available with DFSMS/MVS Version 1 Release 4, and decided to use it to perform the catalog search. Starting with the sample code that IBM provides in SYS1.SAMPLIB member IGGCSILC, I modified the sample to search each catalog for only non-VSAM and generation dataset entries, calling the resulting program SCRUNCA. CSI is capable of performing a search of all catalogs by using a filter of ** and not providing a catalog name in the parameter list, returning a catalog name in the parameter list after its invocation. I attempted to use this feature to search all connected user catalogs, but found an problem when doing this. It turns out that the catalog name returned is the catalog name associated with the last dataset name returned. I experienced occasions where, within the returned data area, information for datasets with different high-level qualifiers are returned. If the high-level qualifiers are connected to different user catalogs, then the catalog name returned cannot be applied to all dataset names returned in the list, but rather only to the last dataset name returned. The reason for needing the correct catalog name for each dataset entry is explained later. For this reason, I wrote a small CLIST to build a list of all user catalog names from within the current master catalog using IDCAMS LISTCAT UCAT. You will notice that I also include a LEVEL(SYS1) on this statement. Our catalog naming convention is to begin each catalog name with SYS1. We have recently, however, implemented an IBM automated tape library, which uses a tape catalog database (TCDB) that, when defined, appears as a user catalog in the master catalog. Fortunately, the high-level qualifier for the TCDB is TCDB, so the LISTCAT UCAT LEVEL (SYS1) only returns the names of true user catalogs in our environment.

Our standard for public volume names is to have the characters SCR starting in either position 1 or 2 of the volume serial number, so after the call, tso CSI, is done, the volume serial number data returned is checked for the above convention. For any match that is found, an IDCAMS DELETE statement is generated with the NOSCRATCH and PURGE keywords. The card is also built to contain the CATALOG parameter specifying the name of the catalog where the entry was found. This is necessary to prevent an instance of a correctly catalogued dataset from being uncatalogued if there happens to be a duplicate catalog entry for the dataset in another catalog. For example if USER.CNTL is catalogued in catalog X to volume TSO001, and USER.CNTL is also incorrectly catalogued in catalog Y to volume SCR001, the second occurrence would meet our criteria for uncataloguing. If the catalog name was not included on the DELETE statement, the correctly catalogued dataset would be uncatalogued instead.

The CLIST builds a complete JCL stream to run the SCRUNCA program, building a SYSIN stream of all the user catalog names that need to be searched. I use a local TSO command in the CLIST called VOL to space information on each volume that contains a catalog – this is to verify that the volume is on-line. If the volume containing a catalog is not on-line (as indicated by a non-zero return code from the VOL command), a message is issued by the CLIST and that catalog name is not generated as input to SCRUNCA. The first step creates a temporary dataset into which the DELETE statement will be written. The second step runs SCRUNCA to read each catalog name and calls CSI to generate the required DELETE statements for any dataset catalogued on a public volume. The final step runs IDCAMS to process the DELETE statements. There are options on the CLIST invocation to just build the job stream without submitting the generated job, in case the job needs to be modified before submission. The JCL generated by the CLIST is shown below.

JCL

```
//SCRUNCA  JOB (1111111),NOTIFY=DRUSER1,CLASS=Z,
//         MSGCLASS=V,REGION=6M
//IEFBR14  EXEC PGM=IEFBR14
//TEMPDD   DD DSN=&TEMP,UNIT=VIO,DISP=(NEW,PASS),
//            SPACE=(CYL,(10,2))
```

```
//SCRUNCA  EXEC PGM=SCRUNCA
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD DISP=(OLD,PASS),DSN=&TEMP
//SYSIN    DD *
SYS1.ICFMCAT.VMVSCTA
SYS1.ICFMCAT.VMVSCTB
SYS1.ICFMCAT.VMVSCTC
SYS1.ICFUCAT.VNETØØ1
SYS1.ICFUCAT.VPRODØ1
SYS1.ICFUCAT.VSTGØØ1
SYS1.ICFUCAT.VTSOØØ1
SYS1.ICFUCAT.VUDB1S1
SYS1.ICFUCAT.VVSAMØ1
/*
//IDCAMS   EXEC PGM=IEFBR14
//SYSIN    DD DISP=(OLD,DELETE),DSN=&TEMP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
```

SAMPLE CLIST

```
PROC Ø LIST SYMLIST CONLIST DEBUG TEST
CONTROL NOLIST NOCONLIST NOSYMLIST
IF &SYMLIST ¬= THEN CONTROL SYM
IF &CONLIST ¬= THEN CONTROL CON
IF &LIST    ¬= THEN CONTROL LIST
IF &DEBUG   ¬= THEN CONTROL SYM CON LIST
SET CODE = Ø
ATTN DO
  SET CODE = 8
  GOTO LEAVE
  END
IF &SYSPREF ¬= THEN DO
   SET PREFIX = &SYSPREF
   PROFILE NOPREFIX
   END
WRITE OBTAINING USER CATALOG LIST
SET SYSOUTTRAP = 9999
LISTC ENT('SYS1.LINKLIB') VOLUME
LISTC UCAT VOLUME LEVEL(SYS1)
SET CC = &LASTCC
SET LIM = &SYSOUTLINE
SET SYSOUTTRAP = Ø
IF &CC > Ø THEN DO
   WRITE LISTCAT COMMAND RETURNED A CODE OF &CC
   SET CODE = 12
   GOTO LEAVE
   END
IF &PREFIX ¬= THEN PROFILE PREFIX(&PREFIX)
SET IX = Ø
```

```
CONTROL NOMSG NOFLUSH
SET T = &STR('SYSTEMS.SCRUNCA.CNTL')
ALLOC DA(&T) F(OUT) REUSE SHR
IF &LASTCC > Ø THEN DO
   WRITE UNABLE TO ALLOCATE SHR DATA SET &T.   TRY NEW ONE.
   ALLOC DA(&T) F(OUT) NEW CAT LRECL(8Ø) BLK(2344Ø) SP(1,1) CYL +
       REUSE RECFM(F B) VOL(MVSØ1Ø)
   IF &LASTCC > Ø THEN DO
      WRITE UNABLE TO ALLOCATE NEW DATA SET &T. ABENDING.
      SET CODE = 8
      GOTO LEAVE
      END
   END
CONTROL MSG
OPENFILE OUT OUTPUT
/*             12345678901234567890123456789012345678901234567890  */
SET OUT = &STR(//SCRUNCA  JOB (1111111),NOTIFY=DRUSER1,CLASS=A, )
PUTFILE OUT
SET OUT = &STR(//         MSGCLASS=V,REGION=6M )
PUTFILE OUT
SET OUT = &STR(//IEFBR14  EXEC PGM=IEFBR14 )
PUTFILE OUT
SET OUT = &STR(//TEMPDD   DD DSN=&&TEMP,UNIT=VIO,DISP=(NEW,PASS), )
PUTFILE OUT
SET OUT = &STR(//            SPACE=(CYL,(1Ø,2)) )
PUTFILE OUT
SET OUT = &STR(//SCRUNCA  EXEC PGM=SCRUNCA )
PUTFILE OUT
SET OUT = &STR(//SYSUDUMP DD SYSOUT=* )
PUTFILE OUT
SET OUT = &STR(//SYSOUT   DD DISP=(OLD,PASS),DSN=&&TEMP )
PUTFILE OUT
SET OUT = &STR(//SYSIN    DD * )
PUTFILE OUT
DO WHILE &IX ¬> &LIM
   DO WHILE &STR(&DØ) ¬= &STR(USERCATALOG)
      SET IX = &IX + 1
      IF &IX > &LIM THEN GOTO LOOPEND
      SET SYSDVAL = &&SYSOUTLINE&IX
      SET SYSDVAL = &STR(&SYSDVAL)
      READDVAL &DØ &D1 &CAT
      SET CAT = &STR(&CAT)
      SET DØ = &STR(&DØ)
   END
   DO WHILE &SUBSTR(1:6,&DØ) ¬= &STR(VOLSER)
      SET IX = &IX + 1
      IF &IX > &LIM THEN GOTO LOOPEND
      SET SYSDVAL = &&SYSOUTLINE&IX
      SET SYSDVAL = &STR(&SYSDVAL)
      READDVAL &DØ
      SET DØ = &STR(&DØ)
```

```
   END
   SET VOL = &SUBSTR(19:24,&DØ)
   VOL &VOL
   IF &LASTCC > Ø THEN DO
      WRITE VOLUME &VOL NOT AVAILABLE FOR CATALOG '&CAT'..
      GOTO LOOPEND
   END
   WRITE PROCESSING CATALOG '&CAT' ON VOLUME &VOL
   ALLOC F(X) DA('&CAT') SHR REU
   IF &LASTCC > Ø THEN DO
      FREE F(X)
      WRITE UNABLE TO ALLOCATE CATALOG '&CAT' ON VOLUME &VOL..
   END
   ELSE DO
       FREE F(X)
       IF &LENGTH(&CAT) > 8 THEN DO
          SET OUT = &STR(&CAT )
          PUTFILE OUT
       END
   END
LOOPEND:  -
END
SET OUT = &STR(/* )  /* END OF SYSIN STREAM */
PUTFILE OUT
SET OUT = &STR(//IDCAMS   EXEC PGM=IDCAMS )
PUTFILE OUT
SET OUT = &STR(//SYSIN    DD DISP=(OLD,DELETE),DSN=&&TEMP )
PUTFILE OUT
SET OUT = &STR(//SYSPRINT DD SYSOUT=* )
PUTFILE OUT
SET OUT = &STR(//SYSUDUMP DD SYSOUT=* )
PUTFILE OUT
CLOSFILE OUT
IF &PREFIX ¬= THEN SET NOTIFY = &STR(NOTIFY)
IF &TEST = THEN DO
   THEN SUBMIT &STR(&T) &NOTIFY
   WRITE BATCH JOB SUBMITTED
   END
ELSE DO
     WRITE BATCH JOB GENERATED INTO &T
     WRITE REVIEW FOR ANY NECESSARY CHANGES
     END
LEAVE:    -
CONTROL NOFLUSH NOMSG
FREE F(OUT)
IF &LASTCC > Ø THEN DO
   CLOSFILE OUT
   FREE F(OUT)
   END
CONTROL MSG
```

```
EXIT CODE(&CODE)
END
```

SCRUNCA CSECT

```
SCRUNCA  CSECT
SCRUNCA  AMODE 24
SCRUNCA  RMODE 24
         YREGS
         STM   R14,R12,12(R13)
         LR    R12,R15
         USING SCRUNCA,R12
         ST    R13,SAVE+4
         LA    R13,SAVE
         OPEN  (PUTDCB,OUTPUT)
         OPEN  (INDCB,INPUT)
NEXTCATN EQU   *
         GET   INDCB,DATAREC               GET CATALOG NAME
         MVC   CSICATNM,DATAREC            MOVE CATALOG NAME
NEXTRCD  EQU   *
         LA    R1,PARMLIST
         CALL  IGGCSIØØ
         LTR   R15,R15                     TEST RETURN CODE
         BZ    NORTCODE                    IF ZERO BYPASS CONVERSION
         B     NEXTCATN
NORTCODE EQU   *
         USING DATARET,R5
         LA    R5,DATAAREA                 LOAD DSECT REG
         L     R1,DRETCD                   GET RETURN CODE
         LTR   R1,R1                       TEST RETURN CODE
         BZ    NEXTFLD                     CONTINUE IF NO ERRORS
         B     NEXTCATN
NEXTFLD  EQU *
         LA    R4,DATAEND                  GET BEGINNING OF INFO
         LA    R7,64                       LENGTH OF ENTRY DATA
         USING ENTRY,R4
NEXTENT  EQU   *
         TM    EFLAG,EERROR                DID ERROR OCCUR FOR ENTRY
         BO    ERRDET                      YES
         B     NVSAMENT                    NO
ERRDET   EQU   *                           ERROR DETECTED
         LA    R1,5Ø                       ADD ENTRY HDR LENGTH
         AR    R7,R1                       ADD ENTRY DATA LEN
         AR    R4,R1                       ADD ENTRY DATA LEN
         C     R7,DUSEDLEN                 COMPARE USED TO CALC LEN
         BNM   NEXTRESM                    IF GT OR EQ  RESUME TEST
         B     NEXTENT                     NEXT ENTRY
NVSAMENT EQU   *
         CLI   ETYPE,ENONVSAM              IF NONVSAM TYPE
```

```
         BE    NVSMCONT                    YES
         CLI   ETYPE,EGDS                  IF GDS COUNTS AS NONVSAM
         BNE   NEXTKEY                     NO
NVSMCONT EQU   *
         MVC   DSNAME,ENAME                SAVE DSNAME
         CLC   EVOLSER(3),SCRLIT           IS IT A SCRATCH VOLSER
         BE    DOVOL                       YES
         CLC   EVOLSER+1(3),SCRLIT         IS IT A SCRATCH VOLSER
         BNE   NEXTKEY                     NO
DOVOL    EQU   *
         MVC   VOLCOM,EVOLSER              SAVE VOLSER
         MVC   CATCOM,CSICATNM             SAVE CATALOG NAME
SKIPVOL  BAL   R3,PRINTNAM                 PRINT NAME OF ENTRY
         B     NEXTKEY                     CONTINUE
NEXTKEY  EQU   *                           CALCULATE NEXT KEY POSITION
         SR    R1,R1                       CLEAR REG 1
         LH    R1,EDATALN                  INSERT ENTRY DATA LEN
         LA    R1,46(R1)                   ADD ENTRY HDR LENGTH
         AR    R7,R1                       ADD ENTRY DATA LEN
         AR    R4,R1                       ADD ENTRY DATA LEN
         C     R7,DUSEDLEN                 COMPARE USED TO CALC LEN
         BNM   NEXTRESM                    IF GT OR EQ  RESUME TEST
         B     NEXTENT                     NEXT ENTRY
NEXTRESM CLI   CSIRESUM,C'Y'               IF MORE ENTRIES
         BE    NEXTRCD                     NEXT RECORD
         B     NEXTCATN                    ELSE FINISH
CLOSEEND EQU   *
         CLOSE (PUTDCB,DISP)
         CLOSE (INDCB,DISP)
         L     R13,SAVE+4
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         BR    R14
PRINTNAM EQU   *
         PUT   PUTDCB,OUTREC1              PUT LINE
         PUT   PUTDCB,OUTREC2              PUT LINE
         PUT   PUTDCB,OUTREC3              PUT LINE
         BR    R3
*********************************************************************
* PARAMETER LIST FOR IGGCSIØØ INVOCATION                            *
*********************************************************************
PARMLIST DS    ØD
         DC    A(MODRSNRT)                 MODULE/REASON/RETURN
         DC    A(CSIFIELD)
         DC    A(DATAAREA)
*********************************************************************
* MODULE ID/REASON CODE/RETURN CODE                                 *
*********************************************************************
MODRSNRT DS    ØF
PARMRC   DS    ØCL4
```

```
MODID    DC    XL2'0000'      MODULE ID
RSNCODE  DC    XL1'00'        REASON CODE
RTNCODE  DC    XL1'00'        RETURN CODE
*********************************************************************
* PARAMETER FIELDS FOR CATALOG SEARCH INTERFACE (CSI)               *
*********************************************************************
CSIFIELD DS    0F
CSIFILTK DC    CL44'**'       FILTER   KEY
CSICATNM DC    CL44' '        CATALOG NAME OR BLANKS
CSIRESNM DC    CL44' '        RESUME NAME OR BLANKS
CSIDTYPD DS    0CL16          ENTRY TYPES
CSIDTYPS DC    CL16'AH              '
CSIOPTS  DS    0CL4           CSI OPTIONS
CSICLDI  DC    CL1'Y'         RETURN D&I IF C A MATCH Y OR BLNK
CSIRESUM DC    CL1' '         RESUME FLAG         Y OR BLANK
CSIS1CAT DC    CL1'Y'         SEARCH CATALOG      Y OR BLANK
CSIRESRV DC    XL1'00'        RESERVED
CSINUMEN DC    H'1'           NUMBER OF ENTRIES FOLLOWING
CSIENTS  DS    0CL8           VARIABLE NUMBER OF ENTRIES FOLLOW
CSIFLDNM DC    CL8'VOLSER  '  FIELD NAME
*
DATAREC  DS    CL80                    INPUT DATA RECORD
SAVE     DS    18F
PUTDCB   DCB   MACRF=PM,DSORG=PS,DDNAME=SYSOUT,RECFM=FB,              X
               LRECL=80
INDCB    DCB   MACRF=GM,DSORG=PS,DDNAME=SYSIN,RECFM=FB,               X
               LRECL=80,EODAD=CLOSEEND
*
         DS    0F
OUTREC1  DS    0CL80                   FIRST RECORD
         DC    CL8' DELETE '
DSNAME   DC    CL44' '
         DC    CL20' NOSCRATCH -       '
         DC    CL8' '
OUTREC2  ds    0cl80                   SECOND RECORD
         DC    CL8' '
         DC    CL10'PURGE CAT('
CATCOM   DC    CL44' '
         DC    CL4')   '
         DC    CL14' '
OUTREC3  DS    0CL80                   THIRD RECORD
         DC    CL12'  /* VOLSER='
VOLCOM   DC    CL6' '
         DC    CL8' */'
         DC    CL54' '
*
SCRLIT   DC    CL3'SCR'
DATAAREA DS    0F
         DC    F'65535'                AREA LENGTH
         DS    XL65535                 AREA PROPER
```

```
DATARET  DSECT
DWORKLEN DS    F
DREQLEN  DS    F
DUSEDLEN DS    F
DPFPLS   DS    H
DCATFLGS DS    CL1
DCATTYPE DS    CL1
DCATNAME DS    CL44
DRETCD   DS    ØCL1
DMODID   DS    CL2
DRSNCOD  DS    CL1
DRETCOD  DS    CL1
DATAEND  DS    ØF
ENTRY    DSECT
EFLAG    DS    XL1
EERROR   EQU   X'4Ø'
ETYPE    DS    XL1
ENONVSAM EQU   C'A'
EGDS     EQU   C'H'
ENAME    DS    CL44
EERRCOD  DS    ØXL4
EDATALN  DS    XL2
EFLD1LN  DS    XL2
EFLD2LN  DS    XL2
EVOLSER  DS    XL6
ENTEND   DS    ØXL1
         END
```


PSF exit to insert new record

THE PROBLEM

It is sometimes necessary to modify existing records or insert new records when the Print Service Facility (PSF) prints Advanced Function Presentation (AFP) data or Line mode data on any AFP intelligent printers in OS390. To do this, IBM supplies two exit routines – APSUX04X and APSUX04Y. You can find these exits in SYS1.SAMPLIB. However, there are no single exits available, that can handle both AFP and line mode data streams.

A SOLUTION

The following exit has been constructed to alleviate these problems. It is able to insert records for selected jobs and selected forms. These features can be achieved as per the operational requirement with only slight modification in the Assembler exit routine.

PROGRAM DESCRIPTION

In our example, the page segment (S1B00000) will be inserted for the selected job name and form name of the current print job in PSF. The source code for page segment S1B00000 is shown below.

PAGE SEGMENT SOURCE CODE (S1B00000)

```
SETUNITS 1 IN 1 IN ;

OVERLAY S1B00000 SIZE 8.5 11 IN
OFFSET 0 0 ;
ORIENT 0 ;
font f3 a0557i;

CONTROL REPLACE;
POSITION 1 IN 1 IN ;
drawbox 4 2
      withtext 0 top right
      line f3 'SAMPLE TEXT'
      withtext 0 bottom left
      line f3 'SAMPLE TEXT'
      withtext 0
      line f3 'SAMPLE TEXT'
      withtext 0 bottom right
      line f3 'SAMPLE TEXT'
      withtext 0 top left
      line f3 'SAMPLE TEXT';
```

This exit will be executed for each record transmitted to PSF. This exit routine runs in the same address space as PSF. Whenever this routine gets executed, it checks for the job name and if it matches with the specified job name(s) in the routine it proceeds further. Extracting the currently printing job name is difficult because there are no PSF manuals, sample Assembler exits, or even macro library guides that clearly show how to extract the job name. But the job name can be extracted by tracing the address pointers provided in ‘Job Separator

Page Data Area'. On entry to the APSUX04 exit routine, Register 1 points to the address of parameter area APSGEXTP. At offset X'4' of APSGEXTP, we can get the address of the IAZJSPA. At offset X'8' of IAZJSPA, we can get the JOBNAME as provided by JES in the variable JSPAJBNM. In our example, the page segment will be inserted only if the job name matches JOBABCDE.

After the job name check, the program is ready to check for the form name. The form name of the currently printing job will be in the exit communication area variable ECAFORM. The program logic can be twisted based on the value of the form name. In our example the page segment will be inserted only if the form name matches STRD.

This exit is capable of differentiating line mode data and AFP data streams. If the current job contains line mode data then our routine checks for form feed (X'F1') in the first position of each record. If it matches then the exit will include page segments. If it finds an AFP data stream, the exit routine will look for the end of text presentation record (X'D3A99B') for each record. If it matches, then it will include page segment data (X'D3AF5F').

APSUX04 ASSEMBLER EXIT

```
APSUXØ4  START Ø
         TITLE 'INSTALLATION EXIT PARAMETER AREA'
         APSGEXTP LIST=YES
         TITLE 'EXIT COMMUNICATION AREA'
         APSUECA LIST=YES
         TITLE 'INDEX TABLE TO HOLD TEMPORARY DATA'
         IAZIDX LIST=YES
         TITLE 'JOB SEPARATOR PAGE DATA AREA - TO EXTRACT JOB NAME'
         IAZJSPA LIST=YES
APSUXØ4  CSECT ,
APSUXØ4  AMODE 31
APSUXØ4  RMODE ANY
         USING *,15
         B     START
         DC    AL1(16)                 LENGTH OF FOLLOWING FIELDS
         DC    CL8'APSUXØ4 '           NAME OF THIS ROUTINE
         DC    CL8'&SYSDATE'           DATE OF THIS ASSEMBLY
         DROP  15
START    DS    ØH
         STM   14,12,12(13)            SAVE CALLERS REGISTERS
         LR    BASEREG,15              SWITCH BASE REGISTER
         USING APSUXØ4,BASEREG         REGISTER 12
         USING APSGEXTP,XTPPTR         REGISTER 4
```

```
         USING APSUECA,ECAPTR          REGISTER 5
         USING IAZJSPA,7               REGISTER 7
         USING IAZIDX,IDXPTR           REGISTER 1Ø
         USING IDXENTRY,IDXEPTR        REGISTER 11
         USING NEWRECX,RECPTR          REGISTER 6
         L     XTPPTR,Ø(,1)            LOAD ADDRESS OF APSGEXTP
         L     7,XTPJSPAP              AT OFFSET 4 EXTRACT ADDRESS
*                                      IAZJSPA(JOB SEPARATOR PAGE DATA)
         MVC   JOBNAME,JSPAJBNM        AT OFFSET 8 EXTRACT JOBNAME
         DROP  7                       DROP BASE REGISTER 7
         L     ECAPTR,XTPECAP          LOAD ADDRESS OF APSUECA
         LR    2,13                    LOAD ADDRESS OF CALLERS SAVE
         LA    13,ECAUSAVE             ADDRESS OF APSUXØ4 SAVE AREA
         ST    2,4(,13)                SAVE CALLERS SAVE AREA ADDRESS
         ST    13,8(,2)                SAVE APSUXØ4 SAVE AREA ADDRESS
         L     IDXEPTR,XTPRIXP         SET PTR TO PSF INDEX ENTRY
         L     CURPTR,IDXRADR          LOAD ADDRESS OF PSF RECORD
*********************************************************************
*      TEST FOR JOBNAME, IF IT IS JOBABCDE, THEN CHECK FOR FORM NAME
*********************************************************************
         CLC   JOBNAME,JNAMECK         CHECK FOR JOBNAME JOBABCDE
         BZ    CHKFORM                 IF YES BRACH TO CHECK FORM TYPE
         BNZ   FINISH                  IF NOT BRACH TO FINISH THIS EXIT
*********************************************************************
*      TEST FOR FORM IF IT IS STRD THEN SKIP ADDING ISP
*********************************************************************
CHKFORM  CLC   ECAFORM,FORM            CHECK FOR FORM NAME STRD
         BNZ   FINISH                  IF NOT BRACH TO FINISH THIS EXIT
*********************************************************************
*      TEST PSF RECORD FOR DATA STREAM AND DATA TYPE
*********************************************************************
         TM    IDXFLAG1,IDXDSR         CHECK FOR AFP DATA STREAM RECORD
         BNZ   LINEDATA                IF NOT BRACH TO LINEDATA
         DS    ØH
         USING NEWRECX,RECPTR
         USING CURREC,CURPTR
*********************************************************************
*      INTIALIZE WORK POINTERS
*********************************************************************
         MVC   XTP4FLAG,XTP4CONV
         LA    IDXPTR,ECAWKBUF         SET ADDRESS OF NEW INDEX
         LA    IDXEPTR,IDXSIZ(IDXPTR)  SET PTR TO 1ST INDEX ENTRY
*********************************************************************
*      CLEAR WORK AREA FOR INDEX AND RECORDS
*********************************************************************
         SLR   8,8                     RESET WORK REGISTER
         SLR   9,9                     RESET WORK REGISTER
         L     7,WORKLEN               SET LENGTH TO CLEAR
         LR    6,IDXPTR                SET ADDR OF TO FIELD
         MVCL  6,8                     MOVE FROM FIELD TO THE TO FIELD
*********************************************************************
*      BUILD THE INDEX HEADER
```

```
*********************************************************************
         MVC   IDXID,INDEXID           SET INDEX HEADER ID
         LA    14,NUMIDXE
         STH   14,IDXNUM               SET NUMBER OF ENTRIES
*********************************************************************
*      COPY THE 1ST INDEX ENTRY AND MODIFY THE CURRENT RECORD.
*      THIS ENTRY POINTS TO THE MODIFIED CURRENT RECORD.
*
*      Set the record address and length in the IDX.  This is
*      necessary for 'spanned' records.  For 'spanned' records,
*      the original IDX points to the last section of the
*      'spanned' record.  This IDX must point to the entire
*      record.
*********************************************************************
*
         L     14,XTPRIXP              SET PTR TO PSF INDEX ENTRY
         MVC   IDXENTRY(IDXESIZ),Ø(14) COPY OLD ENTRY TO NEW ENTRY
         MVC   IDXRECL(2),XTPRECL+2    SET RECORD LENGTH
         MVC   IDXRADR,XTPRECP         SET RECORD ADDRESS
         L     14,XTPRECP              SET PTR TO CURRENT RECORD
         CLI   Ø(14),X'F1'             CHECK FOR FORM FEED IN LINE DATA
         BZ    DONE1                   IF YES BRANCH TO DONE1
         B     FINISH                  IF NOT BRANCH TO FINISH
*********************************************************************
*      INITIALIZE WORK POINTERS
*********************************************************************
LINEDATA DS    ØH
         USING NEWREC,RECPTR           6
         USING CURREC,CURPTR           7
         CLC   DSTYPE,TYPDATA          CHECK FOR DATA TYPE RECORD
         BNE   FINISH                  IF NOT BRANCH TO EPILOGUE
         DROP  CURPTR
         LA    IDXPTR,ECAWKBUF         SET PTR TO NEW INDEX
         LA    IDXEPTR,IDXSIZ(IDXPTR)  SET PTR TO 1ST INDEX ENTRY
*********************************************************************
*      CLEAR WORK AREA.  THE WORK AREA WILL CONTAIN THE INDEX HEADER,
*      3 INDEX ENTRIES FOLLOWED BY 2 ADDED RECORDS.
*********************************************************************
         SLR   8,8                     RESET WORK REGISTER
         SLR   9,9                     RESET WORK REGISTER
         L     7,WORKLEN               SET LENGTH TO CLEAR
         LR    6,IDXPTR                SET ADDR OF TO FIELD
         MVCL  6,8                     MOVE FROM FIELD TO THE TO FIELD
*********************************************************************
*      BUILD THE INDEX HEADER
*********************************************************************
         MVC   IDXID,INDEXID           SET INDEX HEADER ID
         LA    14,NUMIDXE
         STH   14,IDXNUM               SET NUMBER OF ENTRIES
*********************************************************************
*      COPY THE 1ST INDEX ENTRY FROM THE PSF INDEX ENTRY.
*      THIS ENTRY POINTS TO THE CURRENT PSF RECORD.
```

```
*
*      Set the record address and length in the IDX.  This is
*      necessary for 'spanned' records.  For 'spanned' records,
*      the original IDX points to the last section of the
*      'spanned' record.  This IDX must point to the entire
*      record.
*********************************************************************
         L     14,XTPRIXP              SET PTR TO PSF INDEX ENTRY
         MVC   IDXENTRY(IDXESIZ),Ø(14) COPY PSF ENTRY TO NEW ENTRY
         MVC   IDXRECL(2),XTPRECL+2    SET RECORD LENGTH
         MVC   IDXRADR,XTPRECP         SET RECORD ADDRESS
*********************************************************************
*      BUILD THE 2ND INDEX ENTRY AND 1ST ADDED RECORD
*********************************************************************
DONE1    LA    RECPTR,IDXESIZ*NUMIDXE(IDXEPTR)
*                                     SET PTR TO 1ST ADDED RECORD
         LA    14,IDXESIZ              SET LENGTH OF ENTRY
         ALR   IDXEPTR,14              SET PTR TO 2ND ENTRY
         STH   14,IDXENTRL             SET ENTRY LENGTH
         ST    RECPTR,IDXRADR          SET RECORD ADDRESS
         MVI   IDXFLAG1,IDXDSR+IDXANSI SET RECORD TYPE TO DATA STREAM
*                                     AND SET ANSI OR MACHINE CONTROL
*                                     TO INDICATE THE 1ST BYTE ('5A')
*                                     IS A CONTROL CHARACTER.
         MVC   IDXRECL,RECLEN          SET RECORD LENGTH
         MVC   IDXRECID,RECID          SET RECORD ID
         MVC   RECTEXT,DSCON           COMPLETE NEW RECORD TEXT
*********************************************************************
*      UPDATE THE INSTALLATION EXIT PARAMETER LIST
*********************************************************************
         MVI   XTPPIND,XTWRTIX         SET PSF PROCESSING INDICATOR
*                                     TO WRITE RECORDS IN INDEX
         ST    IDXPTR,XTPRIXP          UPDATE POINTER TO NEWLY
*                                     CREATED INDEX HEADER
*********************************************************************
*                         EPILOGUE
*********************************************************************
FINISH   SLR   15,15                   PSF EXPECTS ZERO RETURN CODE
         L     13,4(,13)               RESTORE CALLERS SAVE AREA ADDR.
         L     14,12(,13)              RESTORE CALLERS RETURN ADDRESS
         LM    Ø,12,2Ø(13)             RESTORE CALLERS REGISTERS
         BR    14                      RETURN TO CALLER
         SPACE 2
WRITE    EQU   X'Ø9'                   WRITE WITH SPACE CONTROL
*                                     THAN NECESSARY)
NUMIDXE  EQU   2                       NUMBER OF INDEX ENTRIES
RECLENX  DC    Y(L'RECCC+L'RECTEXTX)   LENGTH OF ADDED RECORD FOR DATA
RECLEN   DC    Y(L'RECTEXT)            LENGTH OF ADDED RECORD FOR AFP
INDEXID  DC    C'IDX '                 USED FOR INDEX HEADER ID
WORKLEN  DC    F'3ØØ'                  LENGTH OF WORK AREA (LENGTH IS
```

```
*                                       LONGER THAN NECESSARY)
RECID    DC    C'APSUX04 '              USED FOR RECORD ID
JNAMECK  DC    C'JOBABCDE'              JOBNAME FOR WHICH DATA STREAM
*                                       TO BE INSERTED
FORM     DC    C'STRD'                  FORM NAME FOR WHICH DATA STREAM
*                                       TO BE INSERTED
TYPDATA  DC    X'D3A99B'                END OF TEXT PRESENTATION
DSCON    DS    CL23                     ADDED DATA STREAM RECORD
         ORG   DSCON
         DC    X'5A'                    DATA STREAM CONTROL CHARACTER
         DC    X'0017'                  LENGTH EXCLUDING THE '5A'
TYPEDATA DC    X'D3AF5F'                INCLUDE PAGE SEGMENT
         DC    X'00'                    FLAGS
         DC    X'0000'                  SEQUENCE
         DC    C'S1B00000'              TESTING MESSAGE
         DC    X'000300'
         DC    X'000300'
         SPACE 2
XTPPTR   EQU   4                        POINTER TO APSGEXTP
ECAPTR   EQU   5                        POINTER TO APSUECA
RECPTR   EQU   6                        POINTER TO APSUECA
CURPTR   EQU   7                        POINTER TO CURRENT PSF RECORD
IDXPTR   EQU   10                       POINTER TO NEW IAZIDX
IDXEPTR  EQU   11                       POINTER TO NEW INDEX ENTRY
BASEREG  EQU   12                       BASE REGISTER
         SPACE 2
NEWREC   DSECT                          ADDED RECORD DESCRIPTION
RECTEXT  DS    CL39                     NEW RECORD INFORMATION


         SPACE 2
CURREC   DSECT                          CURRENT PSF DATA STREAM RECORD
DSCONC   DS    CL1                      CONTROL CHARACTER
         DS    CL2                      LENGTH
DSTYPE   DS    CL3                      RECORD TYPE
NEWRECX  DSECT
JOBNAME  DS    CL8                      CURRENT JOBNAME PRINTING IN PSF
RECCC    DS    CL1
RECTEXTX DS    CL35



         END   APSUX04
```

Muthukumar Kannaiyan
R Systems Inc (USA)

SMP/E alias to convert Assembler H to High Level Assembler

INTRODUCTION

OS/390, previously known as MVS, is legendary for its downward compatibility. Unlike other operating systems, the chances are that programs developed on a System/370 machine twenty or more years ago can still run unchanged on one of the newer releases of OS/390. It is an exception if something like the name of the main load module of a Program Product (PP) changes from release to release. Nevertheless this is exactly what happened with the OS/390 Assembler, and not for the first time. The Assembler F load module was called IFOX00, it became IEV90 with Assembler H, and finally ASMA90 with the arrival of the high-level Assembler.

System programmers often keep the assembly and link-edit JCL together with the source code in one and the same PDS(E) member. This facilitates the assembly because unusual macro libraries, like the JES2 one for instance, are regularly needed. As a result, a large number of systems coding contains the name of the Assembler in the 'EXEC PGM=' JCL line of the source member. Being a system programmer, and therefore more inclined to find a difficult but fool-proof solution for a relatively small problem than to adapt JCL, the most evident solution that springs to my mind is to define an alias using the linkage-editor or the DFSMS Binder. This method equates the old name with the new one while only one physical copy exists. The problem with this conclusion is that the ASMA90 module is managed by SMP/E. This implies that after a PTF did something with the load module, the alias would not be automatically adapted by SMP/E. In fact, after a new linkage-edit by SMP/E, the TTR pointer of the PDS directory would still point to the old location of ASMA90 on the disk. A PDS compress almost certainly would prove to turn out disastrous. The only way to avoid this situation is to modify SMP/E to redefine the alias as well.

Junior system programmers do not seem to be too comfortable with SMP/E. That is why I am convinced that the publication of this

technique could be useful, even if it resolves only a minor problem. In the JCL beneath UCLIN, replace changes the way SMP/E will behave next time after applying maintenance to ASMA90. Since SMP/E will not actually do anything at the moment, a second step is necessary to modify the existing ASMA90. The assembly and linkage-edit parameters were captured from SMP/E.

Notes: The linkage-edit parameters in the two steps must correspond to each other.

On this system all datasets are located by DDDEFs. Check out the DDDEF for the ASMA90 LMOD.

SOURCE

```
//JEDSP4X JOB ('JED:SP'),'JAN DE DECKER',CLASS=A,MSGCLASS=X,
//             NOTIFY=&SYSUID,REGION=ØM
//*
//*
//* DOC: THIS JOB IS NOT A REAL USERMOD. IT CHANGES THE DEFINITION OF
//*      THE WAY SMP/E WILL RELINK-EDIT ASMA9Ø IN CASE OF APPLIED
//*      MAINTENANCE. IT DOES THIS BY UCLIN.
//*
//*      IN A SECOND STEP THE LOADMODULE ASMA9Ø IS CHANGED TO REFLECT
//*      THE CHANGE ALREADY (BEFORE SMP/E MAINTENANCE).
//*
//*      THE BIG CHANGE IS THAT WE DEFINE AN ALIAS WITH THE NAME OF
//*      THE OLD ASSEMBLER H (IEV9Ø) TO BE THE EQUIVALENT OF THE
//*      NEW HIGH-LEVEL ASSEMBLER (ASMA9Ø).
//*
//* NOTE: THE SYSLIB STATEMENT OF THE SECOND STEP SHOULD REFER TO THE
//*       DDDEF KNOWN BY THE CSI USED IN THE FIRST STEP.
//*
//*       ALWAYS CHECK THE LINK-EDIT PARAMETERS IN SMP/E.
//*       IF NECESSARY ADAPT THE IEWL PARAM STATEMENT.
//*
//SMPE     EXEC PGM=GIMSMP
//SMPCSI   DD  DSN=SPB1.MVS.V24Ø.GLOBAL.CSI,DISP=SHR
//SMPCNTL  DD  *
  SET BDY(MVST1ØØ) .
  UCLIN .
  REP LMOD(ASMA9Ø)
      RENT REUS REFR AMODE=ANY RMODE=24 NCAL
++LMODIN
      ORDER    ASMA9Ø
      ENTRY    ASMA9Ø
      NAME     ASMA9Ø(R)
```

```
      ALIAS    IEV90
++ENDLMODIN  .
  ENDUCL .
/*
//S1       EXEC PGM=IEWL,
//            PARM='RENT,REUS,REFR,AMODE=ANY,RMODE=24,NCAL'
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DISP=SHR,DSN=ASMA.V1R2M0.SASMMOD1
//SYSLMOD  DD DISP=SHR,DSN=*.SYSLIB
//SYSUT1   DD UNIT=VIO,SPACE=(CYL,(1,1))
//SYSLIN   DD *
      INCLUDE  SYSLIB(ASMA90)
      ORDER    ASMA90
      ENTRY    ASMA90
      ALIAS    IEV90
      NAME     ASMA90(R)
/*
//*
```

Jan de Decker
Systems Engineer
JED:SP NV (Belgium)

Using overlays

INTRODUCTION

Here is a time saver you can use while EDITing a program (or any other flat file). If you want to save a lot of keystrokes by copying repetitive information, overlays can often save you quite a bit of work.

An overlay is a 'line' command. It is issued at the line level as opposed to the COMMAND line and affects only the line or lines involved.

You can get pretty creative. We will start with a simple example using COBOL code. Let us assume we have existing code that looks like the first example.

I have included the 'BNDS' line to indicate where the boundaries are assumed to be set. We will take a closer look at the 'BNDS' command before we have finished.


```
BNDS <                                    >
00100*      ?
00200     IF X = 2
00300      MOVE X      TO Y.
00400     DISPLAY Y.
00500     CALL 'JULCAL' USING  X
   .
   .
   .
```

Now, suppose we want to comment out lines 200 - 400 (it is obviously easier to do this small amount by hand, but let us imagine that there is a lot of code we want to comment out). The ‘c’ stands for ‘copy’ as usual. ‘o’ means ‘overlay’ and ‘oo’ means overlay a range, just like doubling up other codes indicates a range. The commands can be upper or lower case.

```
c0100*      ?
oo200     IF X = 2
00300        MOVE X     TO Y.
oo400     DISPLAY Y.
00500     CALL 'JULCAL' USING  X
```

This will copy line 100 and overlay it on lines 200 - 400. Only spaces (or nulls) in the target lines will be overlaid. Therefore, the ‘?’ in line 100 will not be overlaid in the target lines. The results will be:

```
00100*      ?
00200*    IF X = 2
00300*     MOVE X       TO Y.
00400*    DISPLAY Y.
00500     CALL 'JULCAL' USING  X
```

Another simple one. Combine two lines into one:

```
m0200     IF X = 2

o0300                  MOVE X  TO Y.
```

result:

```
00300     IF X = 2      MOVE X  TO Y.
```

Let us consider a more complex one:

```
c0100                  TO X-TABLE (01).
oo200     MOVE IN-1
00300     MOVE IN-2
00400     MOVE IN-3
```

```
00500     MOVE IN-4
oo500     MOVE IN-4-THE-COUNT
```

Result:

```
00100                     TO X-TABLE (01).
00200     MOVE IN-1       TO X-TABLE (01).
00300     MOVE IN-2       TO X-TABLE (01).
00400     MOVE IN-3       TO X-TABLE (01).
00500     MOVE IN-4       TO X-TABLE (01).
00500     MOVE IN-4-THE-COUNT-TABLE (01).
```

Note the error in line 500. We should have stopped at line 400. If we had done it correctly, we could now manually change the subscripts in ‘X-TABLE’ and delete line 100 if appropriate.

Another example:

```
cc100                   TO X-TABLE (01).
cc200                   TO Y-TABLE (01).
oo300    MOVE IN-1
00400    MOVE IN-2
00500    MOVE IN-3
00600    MOVE IN-4
oo700    MOVE IN-5
```

Result:

```
00100                   TO X-TABLE (01).
00200                   TO Y-TABLE (01).
00300    MOVE IN-1 TO X-TABLE (01).
00400    MOVE IN-2 TO Y-TABLE (01).
00500    MOVE IN-3 TO X-TABLE (01).
00600    MOVE IN-4 TO Y-TABLE (01).
00700    MOVE IN-5 TO X-TABLE (01).
```

Note that 100 and 200 were repeated until we ran out of places to overlay them. Since there is an odd number of overlay lines, we got three ‘X-TABLE’s and only two ‘Y-TABLE’s.

Now, back to the ‘BNDS’ command. ‘BNDS’ restricts the columns that are involved in many line statements.

Using our first example, let us modify it a bit to make it harder:

```
00100*              ?
00200    IF X = 2
00300      MOVE X      TO Y.
00400    DISPLAY Y.
```

```
00500     CALL 'JULCAL' USING  X
   .
   .
   .
```

Notice that the question mark has moved. It is now in a position to be a problem.

First of all, we can issue a 'BNDS' command to restrict the columns involved.

```
BNDS < >
00100*                 ?
00200     IF X = 2
00300        MOVE X      TO Y.
00400     DISPLAY Y.
00500     CALL 'JULCAL' USING  X
   .
   .
   .
```

After we hit <ENTER>, we can issue our overlay commands as before:

```
BNDS < >
c0100*                ?
oo200     IF X = 2
00300       MOVE X       TO Y.
oo400     DISPLAY Y.
00500     CALL 'JULCAL' USING  X
```

The results will now be correct, because only columns 7 and 8 will be copied and overlaid. The question mark will be ignored.

Do not forget to reset the BNDS to normal before going on to your next task. Only your imagination can restrict your use of these capabilities.

Alan Kalar
Systems Programmer (USA)

Cursor-sensitive ISPF

INTRODUCTION

Have you ever been looking at some job output and wanted to check a dataset whose name was shown in the messages? Normally, you would need to swap to another split screen, choose an option (eg 3.2) then cut and paste the dataset name using your PC software.

This 'DS' command offers an alternative cursor-sensitive method; and it can do much more as well. Here are some examples:

- A user is browsing job output. They use a single click of the mouse to put the cursor on a dataset-name then presses a PF key. The dataset information is then displayed. After that display it returns immediately to showing the job output again.
- A user is browsing some JCL. He types 'DS B' on the command line then does a double-click on a dataset name in the JCL. That dataset is immediately browsed.
- A user is editing a PDS and tries to save an updated member but the dataset is full. Therefore he types 'DS Z' on the command line, moves the cursor up one line to the dataset-name and presses ENTER. The dataset is then compressed. Now it is possible to save the member.
- A user is in a panel of an ISPF application, and PANELID is on. She types 'PNL' on the command line, moves the cursor onto the panel name (at top left) and presses a PF key. DS finds the active panel definition member and invokes a browse of it.

HOW IT WORKS

This command is used from anywhere in ISPF when a dataset name (or volume serial number) is displayed. It works by getting the ISPF screen buffer and finding the dataset name where the cursor is located, then it takes some action on that dataset.

With ISPF Version 4.5 (which came with OS/390 Version 2 Release 5) or later, the code can use some new undocumented ISPF variables

for the screen buffer and cursor position (called ZSCREENI and ZSCREENC). For earlier versions of ISPF, it must use REXX to find ISPF's register 1 from ISPTASK's SAVEAREA, then use undocumented control blocks to find the screen buffer and cursor position. This screen image appears before updates (such as adding PANELID or adding an ISPF message).

The ZSCREENI variable provides the FINAL screen image. Unfortunately, the ZSCREENC variable (cursor position) is not correct when either:

- DS is invoked from inside a pop-up window and the 'SUSPEND' option is used.
- The command line is at the bottom.

These bugs will not be fixed until ISPF Version 4.10 (when the variables officially exist). Hence, the code uses variables ZSCREENI and ZSCREENC only when appropriate.

It is also possible to pass dataset names (and volumes) as parameters. Then it does not look for the screen buffer and cursor position. Here are a couple of examples:

- 'DS B SYS1.PARMLIB VOL001' to invoke a recursive BROWSE of 'SYS1.PARMLIB' library on volume VOL001.
- 'DS MO TEST.MYLIB(NEW*)' to invoke a MOVE of userid.TEST.MYLIB library members with names starting 'NEW'.

DS checks that the dataset exists before invoking any action (except for CMD, DD, L, LC, MSG, PNL, SKL, or VOL), in the following order:

1 It checks for the dataset name specified, treating it as fully qualified. If it is found the action is invoked.

2 If the dataset is not found and the dataset name is not in quotes, DS adds the user's TSO prefix to the start of the dsname then checks again. (If there is no TSO prefix it adds the user-id.) If the dataset is found the action is invoked.

3 If the dataset is still not found an ERROR panel is shown, enabling the user to correct the dataset name (or volser) and try again.

ACTIONS

DS can invoke many different types of dataset action. The available actions include browse, list catalog information, delete, edit, show dataset information, list datasets matching a mask, list members of a PDS, list all datasets on a volume, compress, free unused space, catalog/uncatalog, copy, execute a CLIST/EXEC. The full list can be seen on the DSHELP panel.

Many of these actions are done using the standard facilities of ISPF option 3.4, which normally shows a list of datasets. The DS command supplies an exact dataset name, hence option 3.4 lists only one dataset. Then the user's specified action is automatically entered for that dataset and the ENTER key is simulated – invoking the action. (This requires modified versions of the IBM panels ISRUDLP and ISRDULS0.)

When the user finishes their selected action, the two modified panels are again not displayed because they automatically simulate the END key. Thus the user sees only their desired panels for their action and never the interim (ISRUDLP and ISRDULS0) panels. Some of these DS actions use ISPF 3.4 line commands that were introduced in Version 4.2 of ISPF. 'Action 'DD' invokes the (undocumented) ISRDDN program, which has been a part of ISPF since Version 4.2, for example, 'DS DD SYSEXEC' will list the libraries allocated to SYSEXEC. Hence, this dialog should run OK on any system with ISPF Version 4.2 or later, with all of the defined actions valid. There are also some actions that have a member name as input instead of a dataset name. They use the FINDMEM EXEC to locate the member in a dataset concatenation.

Here are a couple of examples:

- 'DS CMD MYEXEC' will search for MYEXEC EXEC then BROWSE it.
- 'DS MSG ISPYB035' will find member ISPYB03 in ISPMLIB and BROWSE it.

Unlike all the others, the 'BOOK' action is not a dataset function. It provides a cursor-sensitive search for a character string in the BookManager bookshelf of your choice. An example would be looking up a message description directly from a display of some job output.

You are not restricted to just the actions defined in the DS EXEC. For example, you could have a program 'BR' which browses VSAM or BDAM datasets; then DS can be used with action 'BR' and it will invoke the TSO command, 'BR dataset-name'. Similarly, you could specify action 'DSLIST' or 'LCAT' and it would invoke those EXECs, exactly as if you specified the defined actions 'L' or 'LC'.

DEFAULTS

If DS is used with no action parameters specified, it defaults to the last-used action. But you can change it to permanently use the default action of your choice. The following commands control this:

- 'DS DEF action' to set your permanent default action.
- 'DS DEF' to display the current default action.
- 'DS NODEF' to reset it to default to the last-used action.

This setting of the default can be most useful when you have a favourite action that you use most of the time, and especially if a PF key is defined for the DS command.

There are also actions 'SHELF' and 'MEM', which set defaults. They are explained in the HELP panel which follows.

All defaults for DS are stored in a table (called DSVARS) in your ISPPROF dataset, rather than in any ISPF xxxxPROF profile member, so that they can remain consistent even though DS runs in many different ISPF applids.

DSHELP PANEL

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR DEFAULT(%+_)
  ¬TYPE(PT)
  $ TYPE(NT)
  ¬ TYPE(ET)
  # TYPE(CT)
? AREA(SCRL) EXTEND(ON)
)BODY
#HELP$----------------------% DS Command $--------------------------------
#HELP
$Command ==>_ZCMD                                              $Version 8.4
?INFO
```

```
?
)AREA INFO
¬FUNCTION  :$DS invokes some action on a specified dataset, then returns to
$            the panel from which it was invoked.
$
$
¬INVOCATION:$The DS command can be called from anywhere within ISPF by
$            enteringıDS$in the command line and putting the cursor on
$            any character of a dataset-name (or volser or member-name).
$            Then pressıENTER$to invoke it.
$            (It can also be useful to put DS command into a PF-key.)
$
$                     ¬Command ==>ıDS (action)
$
$            Alternatively, enter the command with parameters for action,
$            dataset name and (optionally) volume-serial.
$
$                     ¬Command ==>ıDS action dsname (volser)
$
$            Before any action, DS checks that the dataset exists.  If the
$            dataset is not found (and the dataset-name is not in quotes) it
$            adds your user-id to the start of the dsname then checks again.
$            If the dataset is not found an ERROR panel is shown, enabling you
$            to correct the dsname (or volser) and try again.
$
$
¬DEFAULT   :$IfıDS$is typed WITHOUT PARAMETERS the default action will be
$            used, except when the cursor is also on a blank - then this HELP
$            will be shown. That default action is your last-used action,
$            unless you have set a permanent default action.
$
$       ¬---------------------------------------------------------------------
¬ACTIONS:
$
# DEF      -$Display the DEFAULT action
$
# DEF act  -$Define a permanent DEFAULT action (eg 'D DEF B' set default to B)
$
# NODEF    -$Remove permanent DEFAULT action, so it defaults to last-used
$           action
$
# A        -$Display dataset allocation information (invoking ISPF 3.2)
$
# B        -$BROWSE the dataset
$
# BOOK     -$Search for string in BookManager ('SHELF' action defines
$           bookshelf)
$
# C        -$CATALOG the dataset
$
```

```
# CMD      -$Display a CLIST or EXEC (with BROWSE, EDIT or VIEW)
$
# CO       -$COPY dataset (members)
$
# D        -$DELETE of a member or dataset, confirm-screen is given.
$
# DD       -$List DDNAMES (using standard ISPF program ISRDDN)
$
# DI       -$Display Dataset Information (non-VSAM, on a standard IBM panel)
$
# E        -$EDIT the dataset
$
# F        -$FREE unused space
$
# H        -$This HELP panel is displayed.
$
# HELP     -$This HELP panel is displayed.
$
# I        -$Dataset Information is displayed (using LISTDSI or LISTCAT)
$
# L        -$LIST datasets (invoking ISPF 3.4)
$
# LC       -$Browse dataset CATALOG information (using TSO LISTCAT)
$
# M        -$List dataset MEMBERS
$
# MEM      -$Set DS member display to BROWSE, VIEW or EDIT (for
$          CMD,MSG,PNL,SKL)
$
# MO       -$MOVE dataset member(s)
$
# MSG      -$Display ISPF message definition member
$
# PNL      -$Display ISPF panel definition member
$
# R        -$RENAME the dataset
$
# RS       -$Reset Statistics of dataset members
$
# S        -$Short dataset information (LRECL, BLKSIZE, DSORG, VOLUME, RECFM)
$
# SHELF    -$Set bookshelf for BookManager search ('BOOK' action)
$
# SKL      -$Display ISPF skeleton definition member
$
# U        -$UNCATALOG the dataset
$
# V        -$VIEW the dataset
$
# VOL      -$List all datasets on the VOLUME (invoking ISPF 3.4)
```

```
$
# X (parm) -$EXECUTE a CLIST or EXEC, (with optional parameters)
$
# Z        -$COMPRESS library
$
# ?        -$This HELP panel is displayed.
$
$
$       ¬-------------------------------------------------------------------
$
¬NOTES: $If an UNDEFINED action is entered, DS will assume it is the name of a
$        CLIST/EXEC or program, and try the command:¬TSO action 'dsname'
$
$        Most actions simply do their function then return immediately to the
$        panel from which they were invoked.  However, the following actions
$        need a bit more explanation:
$
$       #A$-shows NONVSAM dataset information, then shows the ISPF 3.2 panel
$           so you can easily allocate a new dataset with the same attributes.
$
$    #BOOK$-is not a dataset action.  It invokes a Bookmanager search for a
$           text string, on the other side of a split screen.  For example,
$           this is typically used to find a message description (eg ¬DS BOOK
$           IEC161I$). Then you could easily flip between the explanation in
$           BookManager and the message text, via the ISPF^SWAP$command.
$           See action:#SHELF
$
$     #CMD$-searches for a CLIST or REXX EXEC then displays the member using
$           BROWSE, EDIT or VIEW as set by the#MEM$action (default is BROWSE).
$           It checks in the search order shown by^TSO ALTLIB DISPLAY$command.
$
$      #DD$-lists libraries and their DDNAMES, using a standard program in
$          ISPF. ¬DS DD *$lists ALL ddnames;¬DS DD ISP$lists all ddnames which
$           include characters 'ISP';^DS DD ISPPLIB MY*$lists all libraries
$           allocated to ddname ISPPLIB and searches for members starting with
$           'MY'.
$
$       #I$-shows dataset information. For VSAM datasets the information
$           comes from the system catalog. The panel for NONVSAM dataset
$           information looks similar to IBM panels, but it additionally
$           allows you to display different datasets or to choose any other DS
$           action.
$
$       #L$-shows a dataset list, then shows the ISPF 3.4 panel so that you
$           can easily change the dataset-name mask to see a new dataset list.
$
$     #MEM$-sets the DS display to BROWSE, VIEW or EDIT for the member actions
$           (ie actions CMD, MSG, PNL, SKL). ¬DS MEM$shows the current
$           default,
$          ¬DS MEM E$will set it to EDIT, and¬DS MEM B$will set it to BROWSE.
```

```
$
$     #MSG$-searches for the message definition member then displays it using
$           BROWSE, EDIT or VIEW as set by the#MEM$action (default is BROWSE).
$           It first checks any LIBDEFs for ISPMLIB, then the ISPMLIB
$           libraries. The message-id is input, and this action truncates it
$           to know which member to browse (eg ¬DS MSG ISREO51$will browse
$           member ISREO5).
$
$     #PNL$-searches for the panel definition member then displays it using
$           BROWSE, EDIT or VIEW as set by the#MEM$action (default is BROWSE).
$           It first checks any LIBDEFs for ISPPLIB, then the ISPPLIB
$           libraries. This is typically used with¬PANELID ON$to display the
$           id of the current panel, then using this action to see its panel
$           definition.
$
$   #SHELF$-sets the bookshelf for the#BOOK$action (eg.^DS SHELF$will display
$           the currently selected shelf and¬DS SHELF MESSAGES$will set it to
$           the 'MESSAGES' shelf).  Your shelf selection is saved in the
$           ISPPROF dataset.  Set your bookshelf before you start using
$           the#BOOK$action, otherwise it will use the initial default 'ALL',
$           which only lists all the bookshelves and does not search them.
$           See action:#BOOK
$
$     #SKL$-searches for the skeleton definition member then displays it using
$           BROWSE, EDIT or VIEW as set by the#MEM$action (default is BROWSE).
$           It first checks any LIBDEFs for ISPSLIB, then the ISPSLIB
$           libraries. This is typically used when browsing program source
$           which uses the FTINCL service, then using this action to see a
$           skeleton member.
$
$     #VOL$-shows a dataset list, then shows the ISPF 3.4 panel so that you
$           can easily change the volser to list datasets on a different
$           volume.
$
$       #X$-executes a CLIST/EXEC using an explicit command:¬TSO EXEC
$           'dsname'$ DS can get the dsname only from the cursor position. You
$           can also add parameter(s) after the action:¬DS X parm$, then it
$           would get the dsname from the cursor and invoke:^TSO EXEC 'dsname'
$           'parm'
$
$       #Z$-compresses a library.  It needs exclusive use of the library, and
$           if that is not possible it displays message:#Dataset in use$.
$           Then you can press PF1 twice to get a^Data Set Contention$panel
$           which shows all the contending users.
$
$       ¬---------------------------------------------------------------------
$
$        To work correctly, this command should be defined in an ISPF command
$        table: ¬ Verb   Trunc     Action
$               #  DS      0     SELECT CMD(%DS &&ZPARM) NEWPOOL
```

```
$
$        Otherwise it must be invoked:#'TSO %DS ........'
$
$        A PF key can also be defined with:#'TSO %DS'$for easier invocation.
$
$        (Note that in all cases the '%' before the 'DS' is necessary.)
$
$       ¬----------------------------------------------------------------
)INIT
 IF (&HELP = YES)            /* &HELP is set to YES by panel DSIHELP */
     &HELP = NO
     &ZERRSM = ''
     &ZERRLM = '              *** use PF10 or PF11 to scroll UP or DOWN ***'
     &ZERRALRM = NO
     .MSG = ISRZ002
     &ZCONT = DSIHELP
)PROC
)END
# Q        -$Show MVS Enqueues for the dataset
$
# ST       -$Invoke StarTools
$
```

DSBLANK PANEL

```
)ATTR
/*-------------------------------------------------------------------------*/
/*  Blank (dummy) panel                                                    */
/*  - used by EXEC DS (Cursor-sensitive dataset processing)                */
/*-------------------------------------------------------------------------*/
)BODY
)END
```

DSDEL PANEL

```
)ATTR
/*-------------------------------------------------------------------------*/
/*  Panel for user to confirm a dataset delete operation                   */
/*  - displayed by exec DS (Cursor-sensitive dataset processing)           */
/*-------------------------------------------------------------------------*/
 + TYPE(NT)                /* normal text     GREEN */
 $ TYPE(ET)                /* emphasised text  TURQ */
)BODY WINDOW(55,18)
+
%COMMAND ===>_ZCMD                                    +
%
+DATA SET NAME:%&DSN
%
+VOLUME ......:%  &VOL
+DS-ORG ......:% &DSORG
+RECORD-FORMAT:%&RECFM
+
```

```
+
+
+   Press$ENTER+key to CONFIRM delete request.
+      (The dataset will be deleted)
+
+   Enter$END+command to CANCEL delete request.
)INIT
  &ZWINTTL = 'Confirm DELETE request'  /* heading for Pop-Up window */
  &ZCMD = &Z
  .HELP = ıDSN
)PROC
 IF (&ZCMD = CAN,CANCEL,EXIT)       /* If &ZCMD = 'END' or 'RETURN' */
     .RESP = END                    /* .. then also: .RESP = 'END'  */
)END
```

DSERR PANEL

```
)PANEL KEYLIST(ISRSPBC,ISR)
)ATTR
/*--------------------------------------------------------------------*/
/*  Panel for user to correct dataset name in case of an error        */
/*  - displayed by exec DS (Cursor-sensitive dataset processing)      */
/*--------------------------------------------------------------------*/
 ~ TYPE(INPUT) INTENS(NON)
 + TYPE(NT)               /* normal text     GREEN */
 $ TYPE(ET)               /* emphasised text  TURQ */
)BODY WINDOW(57,12)
+          ~ZCMD                                 +
+DS action:_ACTION                               +
+
+
+Dataset  :_DSN                                        +
+ Volume  :_VOL   +
+
+
+Change it and press%ENTER+to$RETRY
+
+       ... or press%PF3/PF12+to$CANCEL
)INIT
 &ZWINTTL = 'DS ERROR'              /* heading for Pop-Up window */
 .CURSOR = DSN
)PROC
 IF (&ZCMD = CAN,CANCEL,EXIT)       /* If &ZCMD = 'END' or 'RETURN' */
     .RESP = END                    /* .. then also: .RESP = 'END'  */
)END
/*  Note: ZCMD variable will be cleared by ISPF when SWAP of split
/*      screens is done. Without it - the ACTION variable would be lost!
```

DSIEO PANEL

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------*/
/* Display dataset information for DS command (EXEC)   */
/*  derived from IBM panel ISRUAIEO                      */
/*-------------------------------------------------------*/
)ATTR DEFAULT(%+_)
  ¬ TYPE(PT)
  ~ TYPE(FP)
  # TYPE(VOI) PADC(USER)
  * TYPE(CH)
  ı TYPE(NEF) CAPS(ON) PADC(USER)
  ! AREA(SCRL) EXTEND(ON)
)BODY  CMD(ZCMD)
+                          ¬Data Set Information+
+
~DS Action ===>ıZ
+
!SAREA39
!
!
!
)AREA SAREA39
~Data Set Name . . . . :ıZ                                          +

*General Data*                       *Current Allocation*
 ~Management class . . :#Z       +     ~Allocated &SPCUCØ . :#Z
+
 ~Storage class  . . . :#Z       +     ~Allocated extents . :#Z
+
  ~Volume serial . . . :ıZ     +       ~Maximum dir. blocks :#Z
+
  ~Device type . . . . :#Z       +
 ~Data class . . . . . :#Z       +
  ~Organization  . . . :#Z       +    *Current Utilization*
  ~Record format . . . :#Z     +       ~Used &SPCUC1  . . . :#Z
+
  ~Record length . . . :#Z      +      ~Used dir. blocks  . :#Z
+
  ~Block size  . . . . :#Z       +     ~Number of members . :#Z
+
  ~1st extent &SPCUC2  :#Z            +
  ~Secondary &SPCUC3 . :#Z            +
  ~Dataset name type   :#Z         +

  ~Creation date . . . :#Z         +
~  Last Reference date :#Z         +
)INIT
.ZVARS = '(ZCMD DSN  ZALMC TOTA ZALSC EXTA ZALVOL ZALDIR DEVT ZALDC
DSORG +
           ZALRF TOTU ZALLREC DIRU ZALBLK NRMEM ZAL1EX ZAL2EX ZALDSNT +
           CRDT REFDATE)'
.HELP = DSIHELP
```

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

DSIES PANEL

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------*/
/* Display dataset information for DS command (EXEC)   */
/*  derived from IBM panel ISRUAIES                      */
/*-------------------------------------------------------*/
)ATTR DEFAULT(%+_)
  ¬ TYPE(PT)
  ~ TYPE(FP)
  # TYPE(VOI) PADC(USER)
  * TYPE(CH)
  ı TYPE(NEF) CAPS(ON) PADC(USER)
  ! AREA(SCRL) EXTEND(ON)
)BODY  CMD(ZCMD)
+                           ^Data Set Information+
+
```

```
~DS Action ===>ıZ
+
!SAREA39
!
!
!
)AREA SAREA39
~Data Set Name . . . . :ıZ                                              +

*General Data*                          *Current Allocation*
 ~Management class . . :#Z       +       ~Allocated &SPCUCØ . :#Z
+
 ~Storage class  . . . :#Z       +       ~Allocated extents . :#Z
+
  ~Volume serial . . . :ıZ     +
  ~Device type . . . . :#Z       +
 ~Data class . . . . . :#Z       +      *Current Utilization*
  ~Organization  . . . :#Z       +       ~Used &SPCUC1  . . . :#Z
+
  ~Record format . . . :#Z     +
  ~Record length . . . :#Z      +
  ~Block size  . . . . :#Z       +
  ~1st extent &SPCUC2  :#Z            +
  ~Secondary &SPCUC3 . :#Z            +
  ~Data set name type  :#Z         +

  ~Creation date . . . :#Z         +
~  Last Reference date :#Z         +
)INIT
.ZVARS = '(ZCMD DSN  ZALMC TOTA ZALSC EXTA ZALVOL DEVT ZALDC DSORG +
           TOTU ZALRF ZALLREC ZALBLK ZAL1EX ZAL2EX ZALDSNT CRDT
REFDATE)'
.HELP = DSIHELP
&ZCMD = ' '
  if (&zalspc = CYLINDER)
   &SPCUCØ = 'cylinders :'
   &SPCUC1 = 'cylinders  . . :'
   &SPCUC2 = 'cylinders:'
   &SPCUC3 = 'cylinders :'
  if (&zalspc = TRACK)
   &SPCUCØ = 'tracks  . :'
   &SPCUC1 = 'tracks . . . . :'
   &SPCUC2 = 'tracks . :. '
   &SPCUC3 = 'tracks  . :'
  if (&zalspc = BLOCK)
   &SPCUCØ = 'blocks  . :'
   &SPCUC1 = 'blocks . . . . :'
   &SPCUC2 = 'blocks . :'
   &SPCUC3 = 'blocks  . :'
  if (&zalspc = MEGABYTE)
   &SPCUCØ = 'megabytes :'
   &SPCUC1 = 'megabytes  . . :'
```

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

DSILE PANEL

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------*/
/* Display dataset information for DS command (EXEC)   */
/*  derived from IBM panel ISRUAILE                      */
/*-------------------------------------------------------*/
)ATTR DEFAULT(%+_)
  ¬ TYPE(PT)
  ~ TYPE(FP)
  # TYPE(VOI) PADC(USER)
  * TYPE(CH)
  ı TYPE(NEF) CAPS(ON) PADC(USER)
  ! AREA(SCRL) EXTEND(ON)
)BODY  CMD(ZCMD)
+                           ¬Data Set Information+
+
~DS Action ===>ıZ
+
!SAREA39
!
!
!
)AREA SAREA39
~Data Set Name . . . . :ıZ                                          +

*General Data*                        *Current Allocation*
 ~Management class . . :#Z       +     ~Allocated &SPCUCØ . :#Z
+
 ~Storage class  . . . :#Z       +     ~Allocated extents . :#Z
+
  ~Volume serial . . . :ıZ     +       ~Maximum dir. blocks :#Z
+
  ~Device type . . . . :#Z       +
 ~Data class . . . . . :#Z       +
  ~Organization  . . . :#Z       +    *Current Utilization*
```

```
  ~Record format . . . :#Z     +       ~Used pages  . . . . :#Z
+
  ~Record length . . . :#Z      +      ~Used dir. blocks  . :#Z
+
  ~Block size  . . . . :#Z       +     ~Number of members . :#Z
+
  ~1st extent &SPCUC2  :#Z            +
  ~Secondary &SPCUC3 . :#Z            +
  ~Data set name type  :#Z               +

  ~Creation date . . . :#Z         +
~  Last Reference date :#Z         +
)INIT
.ZVARS = '(ZCMD DSN  ZALMC TOTA ZALSC EXTA ZALVOL DIRA DEVT ZALDC DSORG
+
           ZALRF PAGEU ZALLREC DIRU ZALBLK NRMEM ZAL1EX ZAL2EX ZALDSNT
CRDT +
           REFDATE)'
.HELP = DSIHELP
&ZCMD = ' '
  if (&zalspc = CYLINDER)
   &SPCUCØ = 'cylinders :'
   &SPCUC1 = 'cylinders  . . :'
   &SPCUC2 = 'cylinders:'
   &SPCUC3 = 'cylinders :'
  if (&zalspc = TRACK)
   &SPCUCØ = 'tracks  . :'
   &SPCUC1 = 'tracks . . . . :'
   &SPCUC2 = 'tracks . :. '
   &SPCUC3 = 'tracks  . :'
  if (&zalspc = BLOCK)
   &SPCUCØ = 'blocks  . :'
   &SPCUC1 = 'blocks . . . . :'
   &SPCUC2 = 'blocks . :'
   &SPCUC3 = 'blocks  . :'
  if (&zalspc = MEGABYTE)
   &SPCUCØ = 'megabytes :'
   &SPCUC1 = 'megabytes  . . :'
   &SPCUC2 = 'megabytes:'
   &SPCUC3 = 'megabytes :'
  if (&zalspc = KILOBYTE)
   &SPCUCØ = 'kilobytes :'
   &SPCUC1 = 'kilobytes  . . :'
   &SPCUC2 = 'kilobytes:'
   &SPCUC3 = 'kilobytes :'
  if (&zalspc = BYTE)
   &SPCUCØ = 'bytes . . :'
   &SPCUC1 = 'bytes  . . . . :'
   &SPCUC2 = 'bytes  . :'
   &SPCUC3 = 'bytes . . :'
)PROC
)END
```

DSIP PANEL

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------*/
/* Display dataset information for DS command (EXEC)   */
/*  derived from IBM panel ISRUAIP                       */
/*-------------------------------------------------------*/
)ATTR DEFAULT(%+_)
  ¬ TYPE(PT)
  ~ TYPE(FP)
  # TYPE(VOI) PADC(USER)
  * TYPE(CH)
  ı TYPE(NEF) CAPS(ON) PADC(USER)
  ! AREA(SCRL) EXTEND(ON)
)BODY  CMD(ZCMD)
+                           ¬Data Set Information+
+
~DS Action ===>ıZ
+
!SAREA39
!
!
!
)AREA SAREA39
~Data Set Name  . . . :ıZ                                          +

*General Data*                        +*Current Allocation*
 ~Volume serial . . . :ıZ     +         ~Allocated &SPCUCØ . :#Z
+
 ~Device type . . . . :#Z       +       ~Allocated extents . :#Z
+
 ~Organization  . . . :#Z       +
 ~Record format . . . :#Z     +
 ~Record length . . . :#Z      +
 ~Block size  . . . . :#Z       +     *Current Utilization*
 ~1st extent &SPCUC2  :#Z             + ~Used &SPCUC1  . . . :#Z
+
 ~Secondary &SPCUC3 . :#Z             +
+

 ~Creation date . . . :#Z         +
~ Last Reference date :#Z         +
)INIT
.ZVARS = '(ZCMD DSN  ZALVOL TOTA DEVT EXTA DSORG ZALRF ZALLREC +
           ZALBLK ZAL1EX TOTU ZAL2EX CRDT REFDATE)'
.HELP = DSIHELP
&ZCMD = ' '
```

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

Ron Brown
Systems Progammer (Germany)

MVS news

Tivoli has announced Version 1.3 of its NetView for OS/390, along wth NetView Performance Monitor for measuring network response time, network utilization, and traffic statistics. It's also started shipping its previously-announced Tivoli Service Desk for OS/390 Version 1 Release 2.

Using a new NetView Management Console, Version 1.3 manages both TCP/IP and SNA networks from a single console. It reports both TCP/IP and SNA network to the service desk for problem tracking and resolution. Version 1.3 includes an SNMP Management Information Base (MIB) compiler, said to manage any vendor's networking hardware while reducing problem detection time.

Performance Monitor 2.5 combines performance tracking and reporting for both SNA and TCP/IP networks. It has a new GUI and claimed faster installation and depicts performance in real-time graphically, identifying potential problem areas before they can impact business. When response time or utilization thresholds are exceeded, it sends notification to NetView for corrective action. Tivoli NetView for OS/390 1.3 and Tivoli NetView Performance Monitor 2.5 will be available this quarter.

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

Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK.
Tel: 01753 896 896
Fax: 01753 896 899
http://www.tivoli.com

* * *

Candle has unveiled new versions of its OMEGAMON II and Candle Command Center products for OS/390, and expanded products available for ordering via IBM SystemPac. OMEGAMON II Version 500s (MVS, CICS, DB2, IMS, DBCTL, SMS and VTAM) has a range of new features including a new Application Trace Facility in the version for IMS, analysis of various types of Web-based connections in CICS, and new flexible user profile controls in several versions. OMEGAVIEW II for the Enterprise Version 200 and OMEGAVIEW for 3270 Version 300 get a simplified architecture and higher performance.

New cross-product functions are designed to simplify the task of installing and configuring the software. Among these is the Subsystem Logging Facility for single-point-of-control of OS/390 environments, allowing users to tie together multiple message streams from XCF-connected MVS images with or without a Parallel Sysplex.

Candle also announced new functions in its AF/OPERATOR and OMEGACENTER Gateway to support the OS/390 Automatic Restart Manager and OS/390 alerts and variables.

Candle Corp, 2425 Olympic Blvd, Santa Monica, CA 90404, USA.
Tel: 310 829 5800
Fax: 310 582 4287

Candle, 1 Archipelago, Lyon Way, Frimley, Camberley, Surrey, GU16 5ER, UK.
Tel: 01276 414 700
Fax: 01276 414 777
http://www.candle.com

* * *