223

MVS update

April 2005

In this issue

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £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 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Printed in England.

Recovery of HSM migration control dataset after an error ARC0744E

We recently experienced an issue with the HSM migration control dataset. Unfortunately the actual error was reported several days after it first occurred and this meant we did not have a good back-up of the dataset.

The message that was issued was ARC0744E with a return code of 12 and a reason code of 8.

We initially reviewed IBM's problem databases and came up with a number of APARs describing these errors. There were none that indicated exactly the issue we had, but from what we read we surmised that the dataset had been updated by more than one system.

After running IDCAMS Examine against the data and index components we were nearly 100% certain this was the issue. On further investigation we found that some of our systems did not have certain toleration PTFs installed. We addressed the latter and then we built the JCL below to perform a recovery of the damaged migration control dataset:

```
//JXB7884A   JOB  (JXB),'JXB,J.BRADLEY',CLASS=A
/*JOBPARM SYSAFF=JXB1
//*
//*       ************************************************************
//*      *                                                            *
//*      * REPRO BROKEN MCDS OUT TO FLAT FILE.                        *
//*      *                                                            *
//*       ************************************************************
//*
//STEP1      EXEC PGM=IDCAMS
//DDIN1      DD   DISP=SHR,DSN=HSM.MCDS.DATA
//DDOUT1     DD   DISP=(NEW,CATLG,DELETE),UNIT=SYSDA,
//            DCB=(LRECL=2052,BLKSIZE=20524,RECFM=VB),
//            SPACE=(CYL,(3000,100),RLSE),DSN=HSM.MCDS.RECOVER
//SYSPRINT   DD   SYSOUT=*
//SYSIN      DD   *
  REPRO INFILE(DDIN1) OUTFILE(DDOUT1) FADDR(0)
/*
//*
```

```
//*      ************************************************************
//*      *                                                          *
//*      * RENAME DAMAGED MCDS TO APPEND .OLD SUFFIX AND BACK IT UP. *
//*      *                                                          *
//*      ************************************************************
//*
//STEP2      EXEC PGM=IDCAMS,COND=(Ø,NE)
//SYSPRINT   DD   SYSOUT=*
//SYSIN      DD   *
  ALTER HSM.MCDS NEWNAME(HSM.MCDS.OLD)
  ALTER HSM.MCDS.DATA NEWNAME(HSM.MCDS.DATA.OLD)
  ALTER HSM.MCDS.INDEX NEWNAME(HSM.MCDS.INDEX.OLD)
/*
//STEP3       EXEC  PGM=FDRDSF,REGION=5M,COND=(Ø,NE)
//SYSUDUMP    DD    SYSOUT=*
//SYSPRINT    DD    SYSOUT=*
//DISK1       DD    UNIT=SYSDA,DISP=SHR,VOL=SER=HSM1Ø2
//TAPE1       DD    DSN=JXB7884.HSMDUMP,DISP=(,CATLG,DELETE),
//            UNIT=349Ø,RETPD=1Ø
//SYSIN    DD  *
 DUMP TYPE=DSF
 SELECT DSN=HSM.MCDS.OLD
/*
//*
//*      ************************************************************
//*      *                                                          *
//*      * DELETE THE OLD DATASET.                                  *
//*      *                                                          *
//*      ************************************************************
//*
//STEP4      EXEC PGM=IDCAMS,COND=(Ø,NE)
//SYSPRINT   DD   SYSOUT=*
//SYSIN      DD   *
  DEL HSM.MCDS.OLD
/*
//*
//*      ************************************************************
//*      *                                                          *
//*      * ALLOCATE A NEW MCDS.                                     *
//*      *                                                          *
//*      ************************************************************
//*
//STEP5      EXEC PGM=IDCAMS,COND=(Ø,NE)
//SYSPRINT   DD   SYSOUT=*
//SYSIN      DD   *
  DEFINE CLUSTER (NAME(HSM.MCDS) VOLUMES(HSM1Ø2) -
       CYLINDERS(3ØØØ) -
       STORCLAS(GSPACE) -
       RECORDSIZE(435 2Ø4Ø) FREESPACE(Ø Ø) -
```

```
       INDEXED KEYS(44 Ø) SHAREOPTIONS(3 3) -
       SPEED BUFFERSPACE(53Ø432) -
       UNIQUE NOWRITECHECK) -
       DATA(NAME(HSM.MCDS.DATA) -
       CONTROLINTERVALSIZE(12288)) -
       INDEX(NAME(HSM.MCDS.INDEX) -
       CONTROLINTERVALSIZE(2Ø48)) -
       CATALOG(ICFCAT.HSM)
/*
//*
//*      ************************************************************
//*      *                                                          *
//*      * SORT RECORDS AND PLACE IN A FILE THEN SECOND SORT STEP    *
//*      * WILL REMOVE ANY DUPLICATES THAT EXIST.                   *
//*      *                                                          *
//*      ************************************************************
//*
//STEP6       EXEC  PGM=SORT,COND=(Ø,NE)
//SORTIN      DD    DSN=HSM.MCDS.RECOVER,DISP=SHR
//SORTOUT     DD    DISP=(,PASS),DSN=&&TFILE1,
//            UNIT=SYSDA,SPACE=(CYL,(5ØØ,5Ø),RLSE),
//            DCB=(LRECL=2Ø52,BLKSIZE=Ø,RECFM=VB,DSORG=PS)
//SORTWKØ1    DD    UNIT=339Ø,SPACE=(CYL,(1Ø,1Ø),RLSE)
//SORTWKØ2    DD    UNIT=339Ø,SPACE=(CYL,(1Ø,1Ø),RLSE)
//SYSOUT      DD    SYSOUT=*
//SYSIN       DD    *
  SORT FIELDS=(5,44,CH,A,53,8,BI,D)
/*
//STEP7       EXEC  PGM=SORT,REGION=4Ø96K,COND=(Ø,NE)
//SYSOUT      DD    SYSOUT=*
//SORTIN      DD    DISP=SHR,DSN=&&TFILE1
//SORTOUT     DD    DISP=OLD,DSN=HSM.MCDS
//SYSIN      DD *
 SORT FIELDS=(5,44,CH,A),EQUALS
 SUM FIELDS=(NONE)
/*
//*
//*      ************************************************************
//*      *                                                          *
//*      * EXAMINE THE RECOVERED FILE TO ENSURE STRUCTURALLY CORRECT.*
//*      *                                                          *
//*      ************************************************************
//*
//STEP8    EXEC  PGM=IDCAMS,COND=(Ø,NE)
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
  EXAMINE  -
  NAME(HSM.MCDS) -
  INDEXTEST NODATATEST
```

```
  EXAMINE  -
  NAME(HSM.MCDS) -
  NOINDEXTEST DATATEST
/*
//*
//*       ************************************************************
//*      *                                                            *
//*      * RUN FDR MCDS REPORT TO CHECK ALL DATA LOOKS CORRECT.      *
//*      *                                                            *
//*       ************************************************************
//*
//STEP9        EXEC  PGM=FDREPORT,COND=(Ø,NE)
//SYSUDUMP     DD    SYSOUT=*
//MCDSDD       DD    DSN=HSM.MCDS,DISP=SHR
//SYSPRINT     DD    SYSOUT=*
//ABRMAP       DD    SYSOUT=*
//ABRSUM       DD    SYSOUT=*
//SYSIN     DD  *
  TITLE LINE='HSM MIGRATED DATASET REPORT'
  DEFAULT MCDSCLUSTER=HSM.MCDS
   REPORT FIELD=(DSN,VOL,DSORG,ADATE,ATIME,ADAYS,SIZE,NTMIGRAT)
  SUMMARY FIELD=(DSORG,DSN,SIZE,VOL,NTMIGRAT,ADAYS)
  PRINT DATATYPE=MCDS
/*
```

The job actually ran for 20 minutes and 10 minutes of this was taken up by the dump step to ensure that we had a back-up of the file prior to the reorganization. The dataset contained around 700,000 records and we found after the reorganization that five of these were duplicates – and these were the offending records that were causing problems with the back-up. After the reorganization we immediately ran a back-up of the dataset and it successfully backed up.

Also, we have since added automation to MVS to trap the ARC0744E message and to flag this as a major issue requiring immediate attention.

Note: comments in the JCL detail what each step does. Condition code checking is used to ensure that if anything other than a 0 is returned from a previous job step subsequent job steps do not run.

John Bradley
Systems Programmer
Meerkat Computer Services (UK)

Innovation Data Processing's RTC=YES parameter

A new feature added to Innovation's FDRABR product is the RTC=YES parameter. This parameter can dramatically improve performance of FDR dump processing. RTC is short for READTRACKCCW.

Hidden away in the release documentation, this parameter is often overlooked even though Innovation strongly recommends its usage. If added to FDRDSF, FDRABR, and FDR jobs, dumps can be speeded up and the following advantages gained:

- Disk and tape buffers are moved above the line, allowing more back-ups to run concurrently in a single FDR or ABR JCL step.
- The Read Track CCW is used to read disk data.
- In most cases, RTC=YES will provide a reduction in back-up elapsed time compared with the default of RTC=NO. This reduction is more noticeable when FICON channels to the disk and tape are being utilized.

One negative of using RTC=YES is that it does not support software compression, and the COMPRESS= parameter will be ignored if coded. If you are dumping to a disk dataset such as ABR Archive TAPE*x* on disk, RTC=YES will usually take up more disk space but the elapsed time will be reduced.

For back-ups specifying RTC=YES, FDR requires some additional memory. It needs approximately 500KB of storage below the 16MB line for programs and control blocks and another 2MB (2048KB) of above-the-line storage for each concurrent back-up. If COMPRESS= is specified, about 100KB of below-the-line storage for each concurrent back-up is required.

Innovation also provides a new DD statement, the FDRSUMM

statement, which, if coded, allows FDR to write one-line messages for each volume dumped or restored (giving result codes, elapsed time, and byte counts). This is usually coded as a SYSOUT dataset and will come into effect if RTC=YES is coded on the DUMP statement.

An example of FDRSUMM output from this statement is shown below:

```
FDR001 FAST DUMP RESTORE - FULL VOLUME - FDR  VER. 5.4/30P

         COMP  ELAPSED     VOLUME  DASD BYTES      BYTES ON
VOLSER   CODE  TIME (MIN)  SIZE    READ FROM VOL   BACKUP FILE
JXB001    0     2.7        3,339   1,453,760,616   1,453,789,800
```

The RTC=YES keyword when coded will do the following:

- Use READ TRACK CCWs to read disk data tracks.
- Read up to one cylinder of data at a time.
- Move FDR buffers above the 16MB line.
- Allow more concurrent back-ups to take place in one step.
- Improve back-up elapsed time.

Remember the default for this parameter is NO, and therefore many of the benefits that can be achieved are potentially not being achieved until you manually modify your back-up routines.

An FDR dump JCL deck with the RTC=YES coded is shown below:

```
//JXB7884D   JOB  (JXB),'J.BRADLEY',CLASS=L,USER=JXB7884
//*
//*      ************************************************************
//*      *                                                          *
//*      * DUMP FULL VOLUME.                                        *
//*      *                                                          *
//*      ************************************************************
//*
//STEP1       EXEC PGM=FDR
//SYSPRINT    DD   SYSOUT=*
//SYSPRIN1    DD   SYSOUT=*
//FDRSUMM     DD   SYSOUT=*
```

```
//SYSUDUMP    DD   SYSOUT=*
//DISK1       DD   VOL=SER=JXBØØ1 ,UNIT=339Ø,DISP=SHR
//TAPE1       DD   DSN=JB.JXBØØ1 A,UNIT=349Ø,RETPD=2,
//            DISP=(,CATLG,DELETE),VOL=(,,,1Ø),
//            DCB=UCC1.DSCB,TRTCH=COMP,LABEL=(,SL)
//TAPE11      DD   DSN=JB.JXBØØ1 B,UNIT=349Ø,RETPD=2,
//            DISP=(,CATLG,DELETE),VOL=(,,,1Ø),
//            DCB=UCC1.DSCB,TRTCH=COMP,LABEL=(,SL)
//*
//*      ************************************************************
//*      *                                                          *
//*      * BUFNO=MAX - USE MAXIMUM BUFFERS.                        *
//*      * COMPRESS=COPY2 - COMPRESS THE SECOND OUTPUT TAPE.       *
//*      * DATA=ALL - BACKUP ALL DATA EVEN EMPTY TRACKS.           *
//*      * DSNENQ=NONE - DONT FAIL IT IF ENQ FAILS.                *
//*      * ENQERR=NO - SUPPRESS UØ888 ABEND.                       *
//*      * FORMAT=NEW - WRITE TAPE USING 56K BLOCKS.               *
//*      * RTC=YES - BACKUP A CYLINDER AT A TIME.                  *
//*      *                                                          *
//*      ************************************************************
//*
//SYSIN       DD    *
  DUMP TYPE=FDR,
       BUFNO=MAX,
       COMPRESS=COPY2,
       DATA=ALL,
       DSNENQ=NONE,
       ENQERR=NO,
       FORMAT=NEW,
       RTC=YES
/*
//
```

An example of the output produced is shown below:

```
1FDRØØ1 FAST DUMP RESTORE - FULL VOLUME - FDR  VER. 5.4/3ØP  -
INNOVATION DATA PROCESSING         DATE=2ØØ4.267  PAGE   1
ØFDR3Ø3  CARD IMAGE --   DUMP TYPE=FDR,
 FDR3Ø3  CARD IMAGE --        BUFNO=MAX,
 FDR3Ø3  CARD IMAGE --        COMPRESS=COPY2,
 FDR3Ø3  CARD IMAGE --        DATA=ALL,
 FDR3Ø3  CARD IMAGE --        DSNENQ=NONE,
 FDR3Ø3  CARD IMAGE --        ENQERR=NO,
 FDR3Ø3  CARD IMAGE --        FORMAT=NEW,
 FDR3Ø3  CARD IMAGE --        RTC=YES
 FDRØØ7  STARTING TIME OF ACTUAL DUMP -- 11.17.Ø6 -- UNIT=339Ø-3
,IN=DISK1   ,OUTPUT=TAPE1 TAPE11
ØFDR122  OPERATION STATISTICS FOR 339Ø VOLUME.................JXBØØ1
 FDR122                        CYLINDERS ON VOLUME.............3,339
 FDR122                        DATASETS PROCESSED..................5
```

```
 FDR122                          BYTES READ FROM DASD.....1,453,760,616
 FDR122                          BYTES ON BACKUP..........1,453,789,800
 FDR122                          DASD TRACKS BACKED UP...........29,515
 FDR122                          BACKUP BLOCKS WRITTEN...........29,514
 FDR122                          DASD EXCPS.....................1,988
 FDR122                          BACKUP FILE EXCPS..............3,947
 FDR122                          CPU TIME (SECONDS)..............0.924
 FDR122                          ELAPSED TIME (MINUTES).............2.7
 FDR122                          BACKUP TIME(EXCLUDING MOUNTS)......2.3
 FDR122                          BACKUP COPY 1 ON TAPE DSN=JB.DJXB001A
 FDR122                                                VOL=054891
 FDR122                          BACKUP COPY 2 ON TAPE DSN=JB.DJXB001B
 FDR122                                                VOL=040005
 FDR007  ENDING   TIME OF ACTUAL DUMP -- 11.19.24 -- UNIT=3390-3
,IN=DISK1    ,OUTPUT=TAPE1 TAPE11
 FDR002 FDR DUMP SUCCESSFULLY COMPLETED FROM VOL=JXB001
 FDR999   FDR SUCCESSFULLY COMPLETED
1FDR001 FAST DUMP RESTORE - FULL VOLUME - FDR  VER. 5.4/30P  -
INNOVATION DATA PROCESSING         DATE=2004.267  PAGE   1
0        COMP ELAPSED         VOLUME     DASD BYTES       BYTES ON COMP-
TRACKS
 VOLSER  CODE TIME(MIN)         SIZE  READ FROM VOL    BACKUP FILE PRESS
DUMPED
0JXB001    0     2.7          3,339  1,453,760,616  1,453,789,800    0%
29,515
```

The example below shows a dump JCL deck without RTC=YES coded:

```
//JXB7884D   JOB  (JXB),'J.BRADLEY',CLASS=L,USER=JXB7884
//*
//*      ************************************************************
//*      *                                                          *
//*      * DUMP FULL VOLUME - NO RTC=YES CODED.                     *
//*      *                                                          *
//*      * EXCP'S INCREASED 10 FOLD WHEN RTC=YES NOT CODED.         *
//*      *                                                          *
//*      ************************************************************
//*
//STEP1       EXEC PGM=FDR
//SYSPRINT    DD   SYSOUT=*
//SYSPRIN1    DD   SYSOUT=*
//FDRSUMM     DD   SYSOUT=*
//SYSUDUMP    DD   SYSOUT=*
//DISK1       DD   VOL=SER=JXB001,UNIT=3390,DISP=SHR
//TAPE1       DD   DSN=JB.JXB001C,UNIT=3490,RETPD=2,
//            DISP=(,CATLG,DELETE),VOL=(,,,10),
//            DCB=UCC1.DSCB,TRTCH=COMP,LABEL=(,SL)
//TAPE11      DD   DSN=JB.JXB001D,UNIT=3490,RETPD=2,
//            DISP=(,CATLG,DELETE),VOL=(,,,10),
```

```
//            DCB=UCC1.DSCB,TRTCH=COMP,LABEL=(,SL)
//*
//*      ************************************************************
//*      *                                                          *
//*      * BUFNO=MAX - USE MAXIMUM BUFFERS.                        *
//*      * COMPRESS=COPY2 - COMPRESS THE SECOND OUTPUT TAPE.       *
//*      * DATA=ALL - BACKUP ALL DATA EVEN EMPTY TRACKS.           *
//*      * DSNENQ=NONE - DONT FAIL IT IF ENQ FAILS.                *
//*      * ENQERR=NO - SUPPRESS U0888 ABEND.                       *
//*      * FORMAT=NEW - WRITE TAPE USING 56K BLOCKS.               *
//*      *                                                          *
//*      ************************************************************
//*
//SYSIN       DD    *
  DUMP TYPE=FDR,
       BUFNO=MAX,
       COMPRESS=COPY2,
       DATA=ALL,
       DSNENQ=NONE,
       ENQERR=NO,
       FORMAT=NEW
/*
//
```

An example of the output associated with this job is shown below:

```
IEF376I  JOB/JXB7884/STOP  2004267.1133 CPU    0MIN 13.04SEC SRB    0MIN
01.11SEC
1FDR001 FAST DUMP RESTORE - FULL VOLUME - FDR  VER. 5.4/30P  -
INNOVATION DATA PROCESSING         DATE=2004.267  PAGE   1
0FDR303  CARD IMAGE --   DUMP TYPE=FDR,
 FDR303  CARD IMAGE --        BUFNO=MAX,
 FDR303  CARD IMAGE --        COMPRESS=COPY2,
 FDR303  CARD IMAGE --        DATA=ALL,
 FDR303  CARD IMAGE --        DSNENQ=NONE,
 FDR303  CARD IMAGE --        ENQERR=NO,
 FDR303  CARD IMAGE --        FORMAT=NEW
 FDR007  STARTING TIME OF ACTUAL DUMP -- 11.29.25 -- UNIT=3390-3
,IN=DISK1   ,OUTPUT=TAPE1 TAPE11
 FDR007  ENDING   TIME OF ACTUAL DUMP -- 11.33.00 -- UNIT=3390-3
,IN=DISK1   ,OUTPUT=TAPE1 TAPE11
 FDR122      BYTES DSK TRK  T BLKS  RESTART  STIMERS ERRS ACT DSK  LOW
HGH DEXCP NUMDS COMP BYTES
 FDR122N 1453760616 029515  029514  0000000  0000000  000  029515  000
000 03956 00005 0732683937
 FDR002 FDR DUMP SUCCESSFULLY COMPLETED FROM VOL=JXB001
 FDR999   FDR SUCCESSFULLY COMPLETED
```

Notice in the output that there is a ten-fold increase in EXCPs when RTC=NO is used.

Elizabeth Bradley
Systems Programmer
Meerkat Computer Services Ltd (UK)

An interactive ANTRQST alternative to PPRC CQUERY commands

BACKGROUND

This is the accompanying article to 'A background ANTRQST alternative to PPRC CQUERY commands' published in *MVS Update*, March 2005, issue 222. It is hoped that you have read that article, started using the PPRCSTAT program, and appreciate the flexibility and simplicity it provides. PPRCSTAT was originally written to facilitate the storage management job function, but I quickly realized that there was a requirement to provide the same functionality interactively.

THE DISPPPRC PROGRAM

To provide interactive functionality, I wrote DISPPPRC, which has very similar input and output to that of PPRCSTAT. Again, DISPPPRC will accept a single device number, or a range of device numbers, and will route the output back to the user's TSO screen. The DISPPPRC program should be assembled and link-edited, with AC(1), to an APF-authorized library in the normal manner. Please ensure that this APF-authorized library is available to the user's TSO session via the normal search sequence. The program *must* run authorized because it uses the UCBLOOK macro and, as such, an AUTHCMD entry for it should be added to SYS1.PARMLIB(IKJTSO00) and implemented via IPL, the TSO PARMLIB command, or, for z/

OS 1.3.0 and above, via the SET IKJTSO=00 command.

```
         TITLE 'DISPPPRC: ISSUE ANTRQST PQUERY COMMAND'
DISPPPRC CSECT ,
DISPPPRC AMODE 31
DISPPPRC RMODE 24
         PRINT GEN
         SAVE  (14,12),,DISPPPRC_&SYSDATE._&SYSTIME SAVE REGISTERS
         USING DISPPPRC,R12            ESTABLISH ADDRESSABILITY
         LR    R12,R15                 LOAD BASE REG WITH ENTRY POINT
         LR    R15,R13                 LOAD CALLER'S SAVE AREA ADDRESS
         ST    R13,SAVE+4              SAVE CALLER'S SAVE AREA ADDRESS
         LA    R13,SAVE                LOAD OUR SAVE AREA ADDRESS
         ST    R13,8(R15)              SAVE OUR SAVE AREA ADDRESS
         USING CPPL,R1                 ESTABLISH ADDRESSABILITY
         MVC   CMDCBUF,CPPLCBUF        COPY ADDRESS OF COMMAND BUF
         MVC   CMDUPT,CPPLUPT          COPY UPT ADDRESS
         MVC   CMDECT,CPPLECT          COPY ECT ADDRESS
         DROP  R1                      DROP ADDRESSABILITY
         ICM   R1,15,CMDCBUF           PICK UP BUFFER ADDR
         XR    R2,R2                   CLEAR R2
         ICM   R2,3,2(R1)              LENGTH OF COMMAND NAME
         LA    R1,CMDUPT               PROVIDE LIST FOR PARSE
         LINK  EP=IKJPARS              PARSE COMMAND
         L     R2,CMDANSR              GET POINTER TO ANSWER AREA
         USING IKJPARMD,R2             PROVIDE ADDRESSABILITY
         L     R1,IKJOPT               GET POINTER TO OPERAND
         LH    R2,IKJOPT+4             GET SIZE OF OPERAND
         DROP  R2                      DROP ADDRESSABILITY
         CH    R2,=H'8'                CHECK LENGTH OF OPERAND
*
* NEXT LINE (BH PARMERR) NOT REQUIRED AS CHECKED BY IKJIDENT MAXLNTH
*
*        BH    PARMERR                 LENGTH GT 8, SHOW ERROR
         BE    PARMOK                  LENGTH EQ 8, CONTINUE
         CH    R2,=H'4'                LENGTH EQ 4 ?
         BNE   PARMERR                 N, SHOW ERROR
PARMOK   DS    ØH
         BCTR  R2,RØ                   DECREMENT FOR EXECUTE
         EX    R2,MOVEDEVS             SAVE DEVICE NUMBERS
         CLC   DEVS+4(4),BLANK4        GOT 2ND DEV NUM ?
         BNE   GOT2ND                  Y, PROCESS DEV NOS
         MVC   DEVS+4(4),DEVS          COPY 1ST DEV NUM TO 2ND DEV NUM
GOT2ND   DS    ØH
         LA    R3,DEVS+4               LOAD 2ND DEV NUM
         TR    Ø(4,R3),TRTAB           TRANSLATE A...F TO X'FA...FF'
         PACK  OUTPUT(3),Ø(4,R3)       PACK END DEVICE NUMBER
         L     R4,OUTPUT               GET PACKED DATA
         SRL   R4,12                   DROP LAST 3 NIBBLES
         LR    R7,R4                   SAVE END HEX DEVICE NUMBERS
```

```
         LA    R3,DEVS                 LOAD 1ST DEV NUM
         TR    Ø(4,R3),TRTAB           TRANSLATE A...F TO X'FA...FF'
         PACK  OUTPUT(3),Ø(4,R3)       PACK START DEVICE NUMBER
         L     R4,OUTPUT               GET PACKED DATA
         SRL   R4,12                   DROP LAST 3 NIBBLES
         CR    R4,R7                   START DEVICE NUMBER BIGGER ?
         BH    DEVN1BIG                Y, SHOW ERROR
         LINK  EP=CLEAR                CLEAR THE SCREEN
         LA    R6,1(RØ)                SET INCREMENT TO 1
ANTLOOP  DS    ØH
         STH   R4,XDEVN                SAVE START HEX DEVICE NUMBER
         LA    RØ,L'XQRYINFO           GET SIZE OF XQRYINFO
         STH   RØ,XQRYSIZE             AND SAVE IT
         ANTRQST ILK=PPRC,             PPRC FUNCTION                    *
               REQUEST=PQUERY,         PQUERY                           *
               DEVN=XDEVN,             DEVICE NUMBER                    *
               QRYSIZE=XQRYSIZE,       QRYINFO SIZE                     *
               QRYINFO=XQRYINFO,       OUTPUT INFO AREA                 *
               RETINFO=XRETINFO,       RET CODE INFO AREA               *
               RETCODE=RTNCD,          RETURN CODE                      *
               RSNCODE=RSNCD,          REASON CODE                      *
               MF=(E,P_LIST)           EXECUTE FORM
         LTR   R15,R15                 ANTRQST PARM ERROR ?
         BNZ   ANTRQ_PARMERR           Y, SHOW ERROR
         L     R15,RTC                 LOAD RETINFO RETURN CODE
         L     RØ,RSN                  LOAD RETINFO REASON CODE
         C     R15,=A(RQST_PQUERY_QRYSIZE_BIG_ENOUGH) BIG ENOUGH ?
         BE    BIGENUFF                Y, CONTINUE
         C     R15,=A(RQST_PC_NUMBER_ZERO) ANTASØØØ STARTED ?
         BE    NOTSTART                N, SHOW ERROR
         C     R15,=A(RQST_PQUERY_ERROR) SDM ERROR 724Ø ?
         BNE   ANTRQ_QRYERR            N, SHOW ERROR
         CLC   =CL4'213I',ANTMSG       ESS PAV DEVICE ?
         BE    LOOPCNTL                Y, IGNORE IT AND PROCESS NEXT
         B     ANTRQ_ERROR             N, SHOW ERROR
BIGENUFF DS    ØH
         MVI   BUFFER,C' '             BLANK OUT FIRST CHARACTER
         MVC   BUFFER+1(79),BUFFER     PROPAGATE THROUGH BUFFER
         LA    R11,XQRYINFO            POINT TO ANTRQST INFO
         CLI   XQRYINFO+39,C','        IS IT AN SVA ? IE NO LSSID
         BE    MOVESVA                 Y, MOVE IN SVA DETAILS
         USING EQUERYD,R11             ESTABLISH ADDRESSABILITY
         MVC   BDEVN,EDEVN             MOVE IN DEVICE NUMBER
         MVC   BLEVEL,ELEVEL           MOVE IN LEVEL
         MVC   BSTATE,ESTATE           MOVE IN PAIR STATE
         MVC   BPTHSTAT,EPTHSTAT       MOVE IN PATH STATUS
         MVC   BPSSID,EPSSID           MOVE IN PRIMARY SSID
         MVC   BPCCA,EPCCA             MOVE IN PRIMARY CCA
         MVC   BPLSSID,EPLSSID         MOVE IN PRIMARY LSSID
         MVC   BSSSID,ESSSID           MOVE IN SECONDARY SSID
```

```
         MVC   BSCCA,ESCCA             MOVE IN SECONDARY CCA
         MVC   BSLSSID,ESLSSID         MOVE IN SECONDARY LSSID
         B     GETVOL                  GO AND GET VOLSER FROM UCB
MOVESVA  DS    ØH
         USING SQUERYD,R11             ESTABLISH ADDRESSABILITY
         MVC   BDEVN,SDEVN             MOVE IN DEVICE NUMBER
         MVC   BLEVEL,SLEVEL           MOVE IN LEVEL
         MVC   BSTATE,SSTATE           MOVE IN PAIR STATE
         MVC   BPTHSTAT,SPTHSTAT       MOVE IN PATH STATUS
         MVC   BPSSID,SPSSID           MOVE IN PRIMARY SSID
         MVC   BPCCA,SPCCA             MOVE IN PRIMARY CCA
         MVC   BSSSID,SSSSID           MOVE IN SECONDARY SSID
         MVC   BSCCA,SSCCA             MOVE IN SECONDARY CCA
GETVOL   DS    ØH
         MODESET MODE=SUP              GET INTO SUPERVISOR MODE
         UCBLOOK DEVN=XDEVN,           GET UCB FOR BINARY DEVICE NO.   *
               UCBPTR=UCBPTR,          --> UCB COMMON SECTION          *
               DYNAMIC=YES,            INCLUDE DYNAMIC UCBS            *
               LOC=ANY,                INCLUDE UCBS ABOVE 16MB         *
               RANGE=ALL,              INCLUDE 4DIGIT UCBS             *
               NOPIN,                  DON'T PIN UCB                   *
               RETCODE=RTNCD,          RETURN CODE                     *
               RSNCODE=RSNCD           REASON CODE
         MODESET MODE=PROB             GET INTO PROBLEM MODE
         MVC   BVOLSER,NOVOLSER        PRIME WITH UNKNOWN VOLSER
         LTR   R15,R15                 RCØ FROM UCBLOOK ?
         BNZ   PUT                     N, GO AND SHOW INFO
         L     R5,UCBPTR               GET POINTER TO UCB COMMON AREA
         USING UCBCMSEG,R5             ESTABLISH ADDRESSABILITY
         MVC   BVOLSER,OFFLINE         PRIME WITH OFFLINE VOLSER
         CLC   UCBVOLI,HEXZERO6        OFFLINE ?
         BE    PUT                     Y, GO AND SHOW INFO
         MVC   BVOLSER,UCBVOLI         MOVE IN VOLSER
PUT      DS    ØH
         TPUT  BUFFER,8Ø               SHOW INFO
LOOPCNTL DS    ØH
         BXLE  R4,R6,ANTLOOP           LOOP FOR ALL DEV NOS IN RANGE
         B     RETURN                  AND EXIT
*
RETURN   DS    ØH
         L     R13,4(,R13)             RESTORE CALLER'S SAVE AREA ADDR
         RETURN (14,12),RC=Ø           AND RETURN
*
ANTRQ_PARMERR  EQU  *
         CVD   R15,WA                  CONVERT RET CODE TO DECIMAL
         MVC   MSGARQP+£I1(L'EDMASKT),EDMASKT  MOVE IN EDIT PATTERN
         ED    MSGARQP+£I1(L'EDMASKT),WA+5     EDIT RETURN CODE
         CVD   RØ,WA                   CONVERT RSN CODE TO DECIMAL
         MVC   MSGARQP+£I2(L'EDMASKT),EDMASKT  MOVE IN EDIT PATTERN
         ED    MSGARQP+£I2(L'EDMASKT),WA+5     EDIT REASON CODE
```

```
         TPUT  MSGARQP,LMSGARQP        INFORM USER
         B     RETURN                  AND EXIT
*
ANTRQ_QRYERR   EQU  *
         CVD   R15,WA                  CONVERT RET CODE TO DECIMAL
         MVC   MSGARQQ+£I1(L'EDMASKT),EDMASKT  MOVE IN EDIT PATTERN
         ED    MSGARQQ+£I1(L'EDMASKT),WA+5     EDIT RETURN CODE
         CVD   RØ,WA                   CONVERT RSN CODE TO DECIMAL
         MVC   MSGARQQ+£I2(L'EDMASKT),EDMASKT  MOVE IN EDIT PATTERN
         ED    MSGARQQ+£I2(L'EDMASKT),WA+5     EDIT REASON CODE
         TPUT  MSGARQQ,LMSGARQQ        INFORM USER
         B     RETURN                  AND EXIT
*
ANTRQ_ERROR    EQU  *
         ICM   R9,B'1111',ANTMSGL      MOVE IN MSG LENGTH
         SRL   R9,24                   MOVE TO LAST BYTE OF R9
         LTR   R9,R9                   ZERO MSG LENGTH ?
         BZ    RETURN                  Y, EXIT
         BCTR  R9,Ø                    REDUCE LENGTH FOR EXECUTE
         LA    R8,MSGARQE+5            POINT PAST ANTPØ IN MSG
         EX    R9,MVCANT               MOVE IN ANTMSG
         TPUT  MSGARQE,LMSGARQE        INFORM USER
         B     RETURN                  AND EXIT
*
PARMERR  DS    ØH
         TPUT  MSGPARM,LMSGPARM        INFORM USER
         B     RETURN                  AND EXIT
*
DEVN1BIG DS    ØH
         TPUT  MSGBIG,LMSGBIG          INFORM USER
         B     RETURN                  AND EXIT
*
NOTSTART DS    ØH
         TPUT  MSGNS,LMSGNS            INFORM USER
         B     RETURN                  AND EXIT
*
         DS    ØD
TRTAB    DC    XL256'ØØ'
         ORG   TRTAB+X'81'        ORG TO LOWERCASE 'A'
         DC    X'FAFBFCFDFEFF'    TRANSLATE LOWERCASE 'ABCDEF'
         ORG   TRTAB+C'A'         ORG TO UPPERCASE 'A'
         DC    X'FAFBFCFDFEFF'    TRANSLATE UPPERCASE 'ABCDEF'
         ORG   TRTAB+C'Ø'
         DC    C'Ø123456789'
         ORG
         DS    ØD
*
ECB      DC    F'Ø'               ECB FOR PARSE
RETCODE  DC    F'4'               RETURN CODE (AND MSG SWITCH)
CMDANSR  DC    F'Ø'               PARSE ANSWER AREA POINTER
```

```
CMDUPT   DC    F'Ø'               USER PROFILE TABLE POINTER
CMDECT   DC    F'Ø'               ENVIRONMENT CONTROL TABLE
CMDECB   DC    A(ECB)             ECB POINTER
CMDPCL   DC    A(IKJPCL)          ADDRESS OF IKJPARM
CMDANS   DC    A(CMDANSR)         PLACE TO PUT ANSWER
CMDCBUF  DC    F'Ø'               POINTER TO COMMAND BUFFER
*
         LTORG
*
         REGEQU ,                 REGISTER EQUATES
*
£I1      EQU   26                 OFFSET FOR RTN CODE IN TPUT
£I2      EQU   37                 OFFSET FOR RSN CODE IN TPUT
MVCANT   MVC   Ø(Ø,R8),ANTMSG     MOVE IN ANTMSG
MOVEDEVS MVC   DEVS(Ø),Ø(R1)      SAVE DEVICE NUMBERS
*
MSGARQP  DS    ØF
         DC    C'DISPPPRC:  PARM ERROR: RC=Ø12345 RSN=Ø12345'
LMSGARQP EQU   *-MSGARQP
*
MSGARQQ  DS    ØF
         DC    C'DISPPPRC: QUERY ERROR: RC=Ø12345 RSN=Ø12345'
LMSGARQQ EQU   *-MSGARQQ
*
MSGARQE  DS    ØF
         DC    CL128'ANTPØ'
LMSGARQE EQU   *-MSGARQE
*
MSGPARM  DS    ØF
         DC    C'DISPPPRC: PARM MUST BE 4 OR 8 BYTES IN LENGTH'
LMSGPARM EQU   *-MSGPARM
*
MSGBIG   DS    ØF
         DC    C'DISPPPRC: START DEV NO GREATER THAN END DEV NO'
LMSGBIG  EQU   *-MSGBIG
*
MSGNS    DS    ØF
         DC    C'DISPPPRC: ANTASØØØ MUST BE STARTED'
LMSGNS   EQU   *-MSGNS
*
BLANK4   DC    CL4' '                  BLANK DEVICE NUMBER
EDMASKT  DC    X'4Ø2Ø2Ø2Ø212Ø'         EDIT MASK FOR RTN/RSN CODE
SAVE     DS    18F                     SAVE AREA
OUTPUT   DS    F                       WORK AREA FOR PACK
WA       DS    D                       WORK AREA FOR CVD
DEVS     DC    CL8' '                  DEVICE NUMBERS
UCBPTR   DS    CL4                     POINTER TO UCB COMMON SECTION
NOVOLSER DC    CL6'??????'             USED ON UCBLOOK FAILURE
OFFLINE  DC    CL6'*OFFL*'             OFFLINE VOLUME INDICATOR
HEXZERO6 DC    XL6'ØØØØØØØØØØØØ'       OFFLINE VOLUME IN UCBVOLI
```

```
RTNCD    DS    F                        RETURN CODE
RSNCD    DS    F                        REASON CODE
DEVN1    DS    H                        START HEX DEVICE NUMBER
DEVN2    DS    H                        END HEX DEVICE NUMBER
XDEVN    DS    H                        ANTRQST INPUT HEX DEVICE NUMBER
XQRYSIZE DS    H                        ANTRQST SIZE OF OUTPUT INFO AREA
XQRYINFO DS    CL512                    ANTRQST OUTPUT INFO AREA
         DS    ØF
XRETINFO DS    CL1ØØ                    ANTRQST RETURN CODE INFO
         ORG   XRETINFO
RTC      DS    F                        ANTRQST RETURN CODE
RSN      DS    F                        ANTRQST REASON CODE
ANTMSGL  DS    XL1                      ANTRQST MSG LENGTH
ANTMSG   DS    ØX                       ANTRQST MSG
         ORG   XRETINFO+L'XRETINFO
         ANTRQSTL NAME=P_LIST,BASE=ØF
*
BUFFER   DS    ØCL8Ø                    PRINT LINE
BDEVN    DS    CL4                      DEVICE NUMBER
         DS    CL2
BLEVEL   DS    CL9                      LEVEL (PRIMARY/SECONDARY)
         DS    CL2
BSTATE   DS    CL1Ø                     PAIR STATE (SIMPLEX/DUPLEX/
*                                                  COPYING/SUSPENDED)
         DS    CL3
BPTHSTAT DS    CL8                      PATH STATUS (ACTIVE/INACTIVE)
         DS    CL3
BPSSID   DS    CL4                      PRIMARY SSID
         DS    CL1
BPCCA    DS    CL2                      PRIMARY CCA
         DS    CL1
BPLSSID  DS    CL2                      PRIMARY LSSID
         DS    CL3
BSSSID   DS    CL4                      SECONDARY SSID
         DS    CL1
BSCCA    DS    CL2                      SECONDARY CCA
         DS    CL1
BSLSSID  DS    CL2                      SECONDARY LSSID
         DS    CL3
BVOLSER  DS    CL6                      VOLSER
         DS    CL7
*
IKJPCL   IKJPARM ,                      PARSE PARM CONTROL LIST
IKJOPT   IKJIDENT 'DEV NUMBERS',MAXLNTH=8,FIRST=ALPHANUM,              X
               OTHER=ALPHANUM,PROMPT='DEVICE NUMBER RANGE EG 12345678'
         IKJENDP ,                      PARSE END PARM CONTROL LIST
         IKJCPPL ,                      CMD PROCESSOR PARM LIST
         IEFUCBOB ,                     UCB MAPPING MACRO
*
*  STK SVA ANTRQST PQUERY DATA AREA (SEE ANTPØ91I FOR LAYOUT)
*
```

```
SQUERYD  DSECT
SDEVN    DS    CL4                    DEVICE NUMBER
         DS    CL1
SLEVEL   DS    CL9                    LEVEL (PRIMARY/SECONDARY)
         DS    CL1
SSTATE   DS    CL1Ø                   PAIR STATE (SIMPLEX/DUPLEX/
*                                                  COPYING/SUSPENDED)
         DS    CL1
SPTHSTAT DS    CL8                    PATH STATUS (ACTIVE/INACTIVE)
         DS    CL1
SPSSID   DS    CL4                    PRIMARY SSID
         DS    CL1
SPCCA    DS    CL2                    PRIMARY CCA
         DS    CL1
SPSERIAL DS    CL12                   PRIMARY SERIAL #
         DS    CL1
SSSSID   DS    CL4                    SECONDARY SSID
         DS    CL1
SSCCA    DS    CL2                    SECONDARY CCA
         DS    CL1
SSSERIAL DS    CL12                   SECONDARY SERIAL #
SQUERYDL EQU   *-SQUERYD
*
*  IBM ESS ANTRQST PQUERY DATA AREA (SEE ANTPØ91I FOR LAYOUT)
*
EQUERYD  DSECT
EDEVN    DS    CL4                    DEVICE NUMBER
         DS    CL1
ELEVEL   DS    CL9                    LEVEL (PRIMARY/SECONDARY)
         DS    CL1
ESTATE   DS    CL1Ø                   PAIR STATE (SIMPLEX/DUPLEX/
*                                                  COPYING/SUSPENDED)
         DS    CL1
EPTHSTAT DS    CL8                    PATH STATUS (ACTIVE/INACTIVE)
         DS    CL1
EPSSID   DS    CL4                    PRIMARY SSID
EPLSSID  DS    CL2                    PRIMARY LSS ID
         DS    CL1
EPCCA    DS    CL2                    PRIMARY CCA
         DS    CL1
EPSERIAL DS    CL12                   PRIMARY SERIAL #
         DS    CL1
ESSSID   DS    CL4                    SECONDARY SSID
ESLSSID  DS    CL2                    SECONDARY LSS ID
         DS    CL1
ESCCA    DS    CL2                    SECONDARY CCA
         DS    CL1
ESSERIAL DS    CL12                   SECONDARY SERIAL #
EQUERYDL EQU   *-EQUERYD
*
         END   DISPPPRC
```

DISPPPRC USAGE

The DISPPPRC program has been tested with IBM's Enterprise Storage Server (ESS) and StorageTek's Shared Virtual Array (SVA) DASD subsystems, but should, if required, be tailorable to other manufacturers' DASD. The program can be executed using the following commands:

```
TSO DISPPPRC mmmm
```

for one device, or:

```
TSO DISPPPRC mmmmnnnn
```

for a range of devices, where *mmmm* and *nnnn* are the 4-digit hexadecimal addresses of the devices to be queried.

Entering TSO DISPPPRC 12051208 returns the following to the user's TSO screen:

```
1205  PRIMARY   DUPLEX     ACTIVE    1004 05 04   2004 05 04   AB1205
1206  PRIMARY   SUSPEND(3) ACTIVE    1004 06 04   2004 06 04   AB1206
1207            SIMPLEX    INACTIVE  1004 07 04                AB1207
1208            SIMPLEX    INACTIVE  1004 08 04                *OFFL*
```

From this you can see that device number 1205 is the primary volume in a duplexed pair, 1206 is the primary volume in a suspended duplex pair, 1207 is not PPRCed and on-line, and 1208 is not PPRCed and off-line.

Again, I hope that you will agree that DISPPPRC provides more flexibility than CQUERY, while retaining the interactive process.

Iain McArthur
Systems Programmer (UK)

An EXEC to list all DLI, SQL, MQ, and CICS calls

FUNCTION

This EXEC will scan an entire PDS (either fully or selectively)

or a sequential file for all DLI, SQL, MQ, and/or CICS calls in COBOL programs. Every call is then listed in a new output file. For DLI and MQ series calls, it will list all the parameters that were passed via the CALL statement along with the CALL statement, and for DB2 and CICS it will list all the lines within EXEC and END-EXEC statements. The program name is listed in columns 1–8.

FORMAT

The format of the EXEC to call is:

```
SCANALL  InputFile {DLI='Y|N'} {SQL='Y|N'} {MQ='Y|N'} {CICS='Y|N'}
```

The input can be a PDS or a sequential file. If you want to scan a whole PDS, just give the PDS name. If you want to scan a subset of a PDS, submit the name with a pattern.

Follow the TSO rules, ie if the input dataset does not start with a quote, your TSO userid will be prefixed to the input dataset name. If you do not pass the input file name, you will be prompted for it.

By default this EXEC will scan for DLI, SQL, MQ, and CICS calls. If you want to override these defaults, pass them as parameters. For example DLI='N' will ignore all DLI calls, while SQL='N' will ignore all SQL calls. If all calls have been set to 'N', the EXEC will abort with a message.

CUSTOMIZATION

There are a number of parameters you can customize to suit your installation standards as well as your needs. These parameters are in the BldDefaults subroutine.

Whenever there is a call statement that matches the criteria, the EXEC will write the source statements from column 7 for a length of 61 characters to the output file. If you want to change these – maybe you need to see from column 1 or column 12 – set nStartCol and nLength to suitable values.

I have defaulted the search to be for all IMS and MQ calls, and SQL and CICS EXEC statements. You can override this by modifying the sScanSQL, sScanDLI, sScanCICS, and sScanMQ values.

The EXEC will display ‘Analyzing’ MemberName on your terminal during execution. If you do not want to see this message on the terminal, set sWatch to N.

Modify the output dataset name to your installation standards.

Finally, in the ClearScreen subroutine, modify the module name to one in existence at your installation that clears the terminal screen.

The EXEC identifies DLI calls whenever the second parameter of the CALL statement is either CBLTDLI or has CEETDLI (see the cDliMod variables in the EXEC). This EXEC will assemble all the CALL and EXEC statements on one line before analysing. xSource in AssembleDLI and xSql in AssembleSQL subroutines show the complete CALL and EXEC statements. Should you want to do further analysis of the CALL or EXEC statements, you can add suitable logic.

Cheatsheet: you can scan for any CALL statement in a PDS or flat file. Just add the CALL='Y' parameter when calling this program. If you have called the program with CALL='Y', every CALL statement is listed (ie the EXEC ignores the values set by other parameters).

SCANALL

```
/*                        Rexx                                    */
/* If this character is not | logical OR, please make a    */
/* a global change to modify this character to logical OR. */
/* ------------------------------------------------------- */
/* REXX EXEC to list IMS, DB2, CICS, MQ, or CALL statements*/
/* from a file or PDS                                      */
/* Format is                                               */
/*  IMSDB2  FileName | PdsName{(Pattern)}                  */
/*   {DLI=Y|N} {DB2=Y|N} {MQ=Y|N} {CICS=Y|N} {CALL=Y|N}    */
/* ------------------------------------------------------- */
/* Naming conventions used:                                */
```

```
/* REXX commands & functions start with a capital letter   */
/* TSO  commands are in upper case                         */
/* User data (variables/constants/labels) is a combination */
/* of upper & lower case and                               */
/*  Alphanumeric user data starts with x  (lower x)        */
/*  Numeric      user data starts with n  (lower n)        */
/*  Switches / Flags       start  with s  (lower s)        */
/*  Literals / Constants   start  with c  (lower c)        */
/*  REXX labels start with a capital letter                */
/* ------------------------------------------------------- */
/* Author: Moyeen Ahmed Khan      moyeenkhan@gmail.com     */
/* ------------------------------------------------------- */
Arg xFromDsn xRest

Call BldDefaults                          /* Defaults are here !!!    */
Call InitVars
If xFromDsn='' Then Call GetInput
Call CheckInput
If xType='Pds' Then Call AnalyzePds
  Else Call Analyze
If sHit='Y' Then Call ViewHitsDsn
  Else Call Nothing2Disp
Exit
/*-------------------------  Start Of Subroutines ------------------*/
GetInput:
/*******/
Call ClearScreen
Parse Source . .  xExec .
xMsg.=''
xMsg.1='Welcome To' xExec 'Exec'
xMsg.2='This EXEC will scan and list DLI calls, DB2 SQL statements,'
xMsg.3='CICS EXEC statements, MQ calls or CALL statements'
xMsg.4='Input to this EXEC can be a PDS or a sequential file'
xMsg.5='Type the input dataset name to scan -- TSO rules apply'
xMsg.6='ie if the name is not in quotes,' Userid() 'will be suffixed'
xMsg.7=Copies('-',Length(xMsg.2))
xMsg.8='For a PDS, to analyse all members just type the PDS name OR'
xMsg.9='Choose a pattern for partial selection - use * ? as wild
characters'
xMsg.1Ø='Example of pattern: your selection can be UP*E OR NB* OR *ASK
OR S?KH*'
Do I=1 By 1 While xMsg.I<>''
 Say Center(xMsg.I,74)
 End
Pull xFromDsn .
If xFromDsn='' Then Do
  Say 'Dataset cannot be blank'
  Say 'Input has to be a PDS, PDS with pattern or flat file.'
  Exit
  End
```

```
xFromDsn=Strip(Translate(xFromDsn,"'",'"'))
Return
CheckInput:
/*********/
If Left(xFromDsn,1)<>"'" Then xFromDsn="'"||UserId()"."xFromDsn"'"
If Left(xFromDsn,1)="'" & Right(xFromDsn,1)<>"'" Then
xFromDsn=xFromDsn"'"
If Pos('(',xFromDsn)>Ø Then Do
  Parse Var xFromDsn xFromDsn '(' xMemPat ')' .
  If Verify(xMemPat,'*')=Ø Then xMemPat=''
  xFromDsn=xFromDsn"'"
  End
xAvail=SYSDSN(xFromDsn)
If xAvail<>'OK' Then Do
  zedsmsg=xFromDsn 'Is Not Found'
  zedlmsg='Your Input File does not exist -- Did you type correctly?'
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
xAvailRc=LISTDSI(xFromDsn NORECALL)
If Rc<4 Then Do
  Select
   When Left(SYSDSORG,2)='PO' Then Do
    xDsName=xFromDsn
    xType='Pds'
    End
   When Left(SYSDSORG,2)='PS' Then Do
    xDsName=xFromDsn
    xType='Seq'
    End
   When Left(SYSDSORG,2)='VS' & sVsamSupport='Y' Then Do
    xDsName=xFromDsn
    xType='Vsam'
    End
   Otherwise Do
    zedsmsg='Unsupported DSN type'
    zedlmsg='REXX EXEC cannot read your input dataset'
    Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
    Exit
    End
   End
  End
 Else Do
  zedsmsg='Error accessing' xFromDsn
  zedlmsg=zedsmsg
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
Return
AnalyzePds:
```

```
/*********/
Select
 When Pos('*',xMemPat)>Ø | Pos('?',xMemPat)>Ø Then Do
  xPattern=",PATTERN("Strip(xMemPat)")"
  End
 When xMemPat<>'' Then Do
  xPattern=",PATTERN("Strip(xMemPat)")"
  sOneMember='Y'
  End
 Otherwise xPattern=''
 End
Address ISPEXEC 'LMINIT DATAID(xNameInp) DATASET('xFromDsn') ENQ(SHR)'
If Rc<>Ø Then Do
  zedsmsg='Unable to Access' xFromDsn
  zedlmsg=zedsmsg 'Rc='Rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
Address ISPEXEC 'LMOPEN DATAID('xNameInp') OPTION(INPUT)'
If Rc<>Ø Then Do
  zedsmsg='Unable to Open' xFromDsn
  zedlmsg=zedsmsg 'Rc='Rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
Call ClearScreen
Do Forever
 Address ISPEXEC 'LMMLIST DATAID('xNameInp') OPTION(LIST)
MEMBER(xNextMem) STATS(YES)' xPattern
 If Rc<>Ø Then Leave
 xNextMem=Strip(xNextMem)
 xDsName=Strip(xFromDsn,'T',"'")'('xNextMem")'"
 Call Analyze
 End
Address ISPEXEC 'LMCLOSE DATAID('xNameInp')'
Return
Analyze:
/******/
nMem=nMem+1
If nMem>2Ø Then Do
  nMem=Ø
  Call ClearScreen
  End
Say Center('Analyzing' Left(xNextMem,8),7Ø)
sMemPresent='Y'
nOut=Ø
xOut.=''
xOut.Ø=Ø
Address 'TSO'
xSamFir=MSG('OFF')
```

```
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
'ALLOC DD(xInpDsn) DSN('xDsName') SHR'
If Rc<>Ø Then Do
  zedsmsg='Unable To Allocate' xDSName
  zedlmsg=zedsmsg 'Rc=' rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
'EXECIO * DISKR xInpDsn (STEM xRec.'
If Rc<>Ø Then Do
  zedsmsg='Unable To Read' xDSName
  zedlmsg=zedsmsg 'RC=' Rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit
  End
'EXECIO Ø DISKR xInpDsn (FINIS'
xSamFir=MSG('OFF')
'FREE DD(xInpDsn)'
xAskNb=MSG(xSamFir)
nRec.Ø=xRec.Ø
Do I=1 By 1 Until I=nRec.Ø
 If Substr(xRec.I,7,1)<>'' Then Iterate I
 xRec=Substr(xRec.I,7,66)
 xRec=Space(xRec)
 If Pos('CALL',xRec)=Ø & Pos('EXEC',xRec)=Ø Then Iterate I
 Select
  When sScanSql='Y'  & WordPos('EXEC',xRec)>Ø       Then Call AsmbleExec
  When sScanDli='Y'  & Pos("CALL 'CBLTDLI'",xRec)>Ø Then Call AsmbleCall
  When sScanDli='Y'  & WordPos('CALL',xRec)>Ø       Then Call AsmbleCall
  When sScanCICS='Y' & WordPos('EXEC',xRec)>Ø       Then Call AsmbleExec
  When sScanMQ='Y'   & Pos("CALL 'MQ",xRec)>Ø       Then Call AsmbleCall
  When sScanMQ='Y'   & WordPos('CALL',xRec)>Ø       Then Call AsmbleCall
  When sScanCall='Y' & WordPos('CALL',xRec)>Ø       Then Call AsmbleCall
  Otherwise Nop
  End
 End
If nOut>Ø Then Do
  nOut=nOut+1
  xOut.nOut=Copies('/',8)  Copies('* ',32)     /* Program Separator */
  Call WriteRecs
  End
Return
AsmbleCall:
/**********/
sTime2Exit='N'
xSource=xRec
nStart=I
sSubstract='N'
If Pos('.',xRec)>Ø & xRec<>'.' Then sTime2Exit='Y'
```

```
                            /* ************************************** */
                            /* Let us assemble the Call statements    */
                            /* into a single line                     */
                            /* ************************************** */
Do While sTime2Exit='N'
 I=I+1
 If I=nRec.Ø Then sTime2Exit='Y'
 If Substr(xRec.I,7,1)<>' ' Then Iterate
 xRec=Substr(xRec.I,7,64)
 xRec=Space(xRec)||' '
 Select
  When WordPos('END-CALL',xRec)>Ø Then Do
    xSource=xSource xRec
    sTime2Exit='Y'
    End
  When WordPos(Word(xRec,1),cCblVerbs)>Ø Then Do
    sSubstract='Y'
    sTime2Exit='Y'
    End
  When WordPos(Word(xRec,1),cEndVerbs)>Ø Then Do
    sSubstract='Y'
    sTime2Exit='Y'
    End
  When Pos('.',xRec)>Ø Then Do
    xSource=xSource xRec
    sTime2Exit='Y'
    End
  Otherwise xSource=xSource xRec
  End
 End
                            /* -------------------------------------- */
                            /* Here, xSource has the complete CALL    */
                            /* statement.                             */
                            /* -------------------------------------- */
nPos=WordPos('CALL',xSource)+1
If nPos<>2 Then Return
xWord=Word(xSource,nPos)
Select
 When sScanCall='Y' Then Nop
 When Pos(xWord,cDliMods)>Ø & sScanDli='Y' Then Nop
 When Left(xWord,3)="'MQ" & sScanMQ='Y' Then Nop
 Otherwise Return
 End
If sSubstract='N' Then nEnd=I
 Else nEnd=I-1
Do J=nStart By 1 While J<=nEnd
 If Substr(xRec.J,7,1)=' ' Then Do
   nOut=nOut+1
   xOut.nOut=Left(xNextMem,8)  Substr(xRec.J,nStartCol,nLength)
   End
```

```
 End
Return
AsmbleExec:
/**********/
sEndExc='N'
nStart=I
xExec=Space(xRec,1)
If Pos('END-EXEC',xRec)>Ø Then sEndExc='Y'
Do While sEndExc='N'
 I=I+1
 If I=nRec.Ø Then sEndExc='Y'
 xRec=Substr(xRec.I,7,66)
 Select
  When Word(xExec,2)='SQL' Then Nop
  When Word(xExec,2)='CICS' Then Nop
  Otherwise Return
  End
 xExec=xExec Space(xRec,1)
 If Pos('END-EXEC',xRec)>Ø Then sEndExc='Y'
 End
                           /* ------------------------------------- */
                           /* Here, xExec has the complete EXEC     */
                           /* EXEC statement                        */
                           /* ------------------------------------- */
nEnd=I
xSecondWord=Word(xExec,2)
Select
 When xSecondWord='SQL' & sScanSql='Y' Then sWrite='Y'
 When xSecondWord='CICS' & sScanCICS='Y' Then sWrite='Y'
 Otherwise sWrite='N'
 End
If sWrite='Y' Then Do
  Do J=nStart By 1 While J<=nEnd
   If Substr(xRec.J,7,1)=' ' Then Do
     nOut=nOut+1
     xOut.nOut=Left(xNextMem,8) Substr(xRec.J,nStartCol,nLength)
     End
   End
  End
Return
WriteRecs:
/********/
If sFirstTime='Y' Then Do
  Call AllocOutput
  End
'EXECIO * DISKW xOutDsn  (STEM xOut.'
If Rc<>Ø Then Do
  xSamFir=MSG('OFF')
  'FREE DD(xOutDsn)'
  xAskNb=MSG(xSamFir)
```

```
  zedsmsg='Write Failure On' xFileOut
  zedlmsg=zedsmsg 'RC='rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit 16
  End
nOutRecs=nOutRecs+xOut.Ø
sHit='Y'
Return
AllocOutput:
/**********/
xOutName="'"xOutName"'"
xAvail=SYSDSN(xOutName)
If xAvail='OK' Then Do
  xSamFir=MSG('OFF')
  'DELETE' xOutName
  xAskNb=MSG(xSamFir)
  If Rc<>Ø Then Do
    zedsmsg=xOutName 'Could Not Be Deleted'
    zedlmsg=zedsmsg 'RC=' Rc
    Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
    Exit
    End
  End
xDfltDisp='NEW UNIT(SYSDA) LRECL('nLrecl') SPACE(4Ø) DSORG(PS)
RECFM(F,B) TRACKS RELEASE'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
'ALLOCATE DSN('xOutName') DD(xOutDsn)' xDfltDisp
If Rc<>Ø Then Do
  zedsmsg='Alloc Failed' xOutName
  zedlmsg=zedsmsg 'RC='Rc
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Exit 16
  End
sFirstTime='N'
Return
ViewHitsDsn:
/**********/
'EXECIO Ø DISKW xOutDsn (FINIS'
xSamFir=MSG('OFF')
'FREE DD(xOutDsn)'
xAskNb=MSG(xSamFir)
Call ClearScreen
Address ISPEXEC 'VIEW DATASET ('xOutName')'
Return
Nothing2Disp:
/**********/
If sMemPresent='N' Then Do
  zedsmsg='No Members To Match'
```

```
  zedlmsg='No Members Found In Pds' xFromDsn 'That Could Match' xMemPat
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  Return
  End
If nOutRecs=Ø Then Do
  zedsmsg='No Hits For Your Statements'
  xAnam=''
  If sScanSql='Y'  Then xAnam=xAnam 'DB2'
  If sScanDli='Y'  Then xAnam=xAnam 'IMS'
  If sScanMq='Y'   Then xAnam=xAnam 'MQ'
  If sScanCICS='Y' Then xAnam=xAnam 'CICS'
  If sScanCall='Y' Then xAnam=xAnam 'CALL'
  nTotal=Words(xAnam)
  If nTotal>1 Then Do
    xAfsh=Subword(xAnam,1,nTotal-1)
    xAysh='Or' Word(xAnam,nTotal)
    xAnam=xAfsh xAysh
    End
  zedlmsg='There were no' xAnam 'statements found'
  Address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
  End
Return
ClearScreen:
/**********/
/* **************
"CALL 'PROD.LINKLIB(OPJCLEAR)'"
 ************** */
'CLRSCRN'
Return
InitVars:
/**********/
nMem=Ø
nOutRecs=Ø
sFirstTime='Y'
sHits='N'
sMemPresent='N'
xMsg=''
xSrchDsn=''
xMemPat=''
xSara=''
cCblVerbs=''
cVerbs.='*'
cVerbs.1='ACCEPT ADD ALTER CALL CANCEL CLOSE COMPUTE CONTINUE'
cVerbs.2='DELETE DISPLAY DIVIDE ELSE ENTER EVALUATE EXIT'
cVerbs.3='GOBACK GO IF INITIALIZE INSPECT MERGE MOVE MULTIPLY'
cVerbs.4='OPEN PERFORM READ RELEASE RETURN REWRITE SEARCH SET'
cVerbs.5='SORT START STOP STRING SUBSTRACT UNSTRING WRITE'
Do I=1 By 1 While cVerbs.I<>'*'
 cCblVerbs=cCblVerbs  cVerbs.I
 End
```

```
cEndVerbs=''
Do I=1 By 1 Until I=Words(cCblVerbs)
 cEndVerbs=cEndVerbs 'END-'||Word(cCblVerbs,I)
 End
cDliMod.1="'CBLTDLI'"
cDliMod.2="WS-CEETDLI MODULE-CEETDLI DYN-CEETDLI 'CEETDLI' LIT-CEETDLI"
cDliMod.3="L-CEETDLI LT-CEETDLI W-CEETDLI W-C-CEETDLI"
cDliMods=cDliMod.1 cDliMod.2 cDliMod.3
If xRest<>'' Then Call OverRideDflts
If sScanSQl||sScanDLI||sScanCICS||sScanMQ||sScanCall='NNNNN' Then Do
  zedsmsg='Nothing To Scan For!'
  zedlmsg='Pl supply / override at least one Search Argument'
  Address ISPEXEC 'SETMSG MSG(ISRZ001)'
  Exit
  End
Select
 When nLength=0 Then nLength=72
 When nLength+10<80 Then nLrecl=80
 Otherwise nLrecl=100
 End
Return
OverRideDflts:
/************/
xSql=''
xDli=''
xCICS=''
xMQ=''
xCall=''
xRest=Translate(xRest,'',",'")
If Pos('DB2=',xRest)>0  Then Parse Var xRest . 'DB2='  xSql .
If Pos('SQL=',xRest)>0  Then Parse Var xRest . 'SQL='  xSql .
If Pos('IMS=',xRest)>0  Then Parse Var xRest . 'IMS='  xDlI .
If Pos('DLI=',xRest)>0  Then Parse Var xRest . 'DLI='  xDlI .
If Pos('CICS=',xRest)>0 Then Parse Var xRest . 'CICS=' xCics .
If Pos('MQ=',xRest)>0   Then Parse Var xRest . 'MQ='   xMq .
If Pos('CALL=',xRest)>0 Then Parse Var xRest . 'CALL=' xCall .
If xSql<>'' Then Do
  xSql=Strip(xSql)
  If xSql='Y' Then sScanSql='Y'
    Else sScanSql='N'
 End
If xDli<>'' Then Do
  xDli=Strip(xDli)
  If xDli='Y' Then sScanDli='Y'
   Else sScanDli='N'
 End
If xCICS<>'' Then Do
  xCICS=Strip(xCICS)
  If xCICS='Y' Then sScanCICS='Y'
   Else sScanCICS='N'
```

```
 End
If xMQ<>'' Then Do
  xMQ=Strip(xMQ)
  If xMQ='Y' Then sScanMQ='Y'
   Else sScanMQ='N'
 End
If xCall<>'' Then Do
  xCall=Strip(xCall)
  If xCall='Y' Then sScanCall='Y'
   Else sScanCall='N'
 End
Return
/* ------Note----------------Note----------------------Note-------- */
/* The defaults are defined here. Modify them to suit your needs    */
/* or standards.                                                    */
/* ---------------------------------------------------------------- */
BldDefaults:
/**********/
nStartCol=7                              /* Display data from this column*/
nLength=61                               /* for this length from source  */
sScanCICS='Y'                            /* Y = Scan CICS    statements  */
sScanDLI='Y'                             /* Y = Scan IMS/DLI statements  */
sScanMQ='Y'                             /* Y = Scan MQ      statements  */
sScanSQl='Y'                             /* Y = Scan DB2 SQL statements  */
sScanCall='N'                            /* Y = Scan CALL    statements  */
sWatch='Y'                               /* Y = Show PDS membrs on screen*/
sVsamSupport='N'                         /* Y = ISPF can read VSAM File  */
xOutName=Userid()'.ASKNB.IMSDB2.LIST' /* Output dataset name          */
Return
```

Moyeen A Khan (moyeenkhan@gmail.com)
(USA)

Integrated Catalog Facility Recovery utility

The Integrated Catalog Recovery utility is a special suite of programs that can be utilized to recover Integrated Catalog Facility catalogs. It can be used to read SMF records and to repair a broken catalog. The utility requires a valid back-up of the catalog to be repaired from an IDCAMS Export, and access to the various SMF records to do a forward recovery. Clear processes and procedures are required to use the utility correctly, and one of the key factors affecting its usage is that

some effort is required upfront to develop adequate catalog diagnosis and back-ups.

This article highlights processes that should be put into place to allow the utility to be used correctly and then goes on to show how the utility should be used to perform a recovery. The article's aim is to provide you with a structure for developing similar processes at your installation.

BACKING UP CATALOGS

Your catalog environment is one of the most critical parts of your MVS system. A failed catalog can result in many datasets being inaccessible, or, if the catalog is system related, an outage can result in the total loss of your system. In the light of this, it is interesting just how many people do not perform any maintenance or review of their environment, and, in many cases, just how many sites do not perform standard catalog back-ups – relying on full volume dumps to recover them. If you share master catalogs in a Sysplex you can even lose the whole of your Sysplex if the master catalog becomes damaged.

The first thing to consider is how often to back up catalogs. My recommendation would be certainly to back-up all catalogs once daily. For more critical catalogs, or ones that fluctuate frequently, you may want to initiate regular hourly or two-hourly back-ups. It really does depend on your recovery objectives. The key thing is that you are backing them up.

As you would expect there are a number of different ways to back up catalogs. The available catalog back-up utilities include:

- IDCAMS Export – supplied as part of IDCAMS.
- DSS Physical Dump – requires the DFSMSdss product.
- DSS Logical Dump – requires the DFSMSdss product.
- HSM BACKDS command – requires the DFSMShsm product.

- Third-party products – there are several products on the market that specialize in catalog recovery.

IDCAMS Export is by far the most widely used back-up utility. It is easy to use and easily built. It is possible to back up only one catalog per job step. The job below is an IDCAMS Export of two catalogs:

```
//EXB788R     JOB   (JXB),'J.BRADLEY',CLASS=A
//*
//*       ************************************************************
//*      * CATALOG BACKUP JOB.                                        *
//*      *                                                            *
//*      * STEP1 - DIAGNOSE CATALOGS FOR ERRORS.                      *
//*      * STEP2 - BACKUP FIRST CATALOG. STORE ON FILE 1 OF TAPE.     *
//*      * STEP3 - BACKUP SECOND CATALOG. STORE ON FILE 2 OF TAPE.    *
//*      * STEP4 - BACKUP LISTCAT OUTPUT. STORE ON FILE 3 OF TAPE.    *
//*      * STEP5 - DIAGNOSE CATALOGS FOR ERRORS AS PRECAUTION.        *
//*      *                                                            *
//*       ************************************************************
//*
//STEP1            EXEC  PGM=IDCAMS
//SYSPRINT    DD       SYSOUT=*
//SYSIN            DD      *
  EXAMINE NAME(CATALOG.ACAT) INDEXTEST NODATATEST
  EXAMINE NAME(CATALOG.ACAT) NOINDEXTEST DATATEST
  EXAMINE NAME(CATALOG.CCAT) INDEXTEST NODATATEST
  EXAMINE NAME(CATALOG.CCAT) NOINDEXTEST DATATEST
  DIAGNOSE ICFCATALOG INDATASET(CATALOG.ACAT)
  DIAGNOSE ICFCATALOG INDATASET(CATALOG.CCAT)
/*
//STEP2             EXEC  PGM=IDCAMS
//SYSPRINT    DD        SYSOUT=*
//BACKCAT1  DD        DSN=EXB7884.CATBACK.ACAT,DISP=(,CATLG,DELETE),
//                        UNIT=(349Ø,,DEFER),LABEL=(1,SL),VOL=(,RETAIN)
//SYSIN            DD       *
  EXPORT CATALOG.ACAT TEMPORARY OUTFILE(BACKCAT1)
/*
//STEP3            EXEC  PGM=IDCAMS
//SYSPRINT    DD      SYSOUT=*
//BACKCAT2  DD      DSN=EXB7884.CATBACK.CCAT,DISP=(,CATLG,DELETE),
//                        UNIT=(349Ø,,DEFER),LABEL=(2,SL),
//                        VOL=(,RETAIN,REF=*.STEP2.BACKCAT1)
//SYSIN            DD      *
  EXPORT CATALOG.CCAT TEMPORARY OUTFILE(BACKCAT2)
/*
//STEP4            EXEC  PGM=IDCAMS
//SYSPRINT    DD    DSN=EXB7884.CATBACK.LISTCAT,DISP=(,CATLG,DELETE),
//                        UNIT=(349Ø,,DEFER),LABEL=(3,SL),
```

```
//                      VOL=(,RETAIN,REF=*.STEP2.BACKCAT1)
//SYSIN          DD        *
  LISTC ENT(CATALOG.ACAT) ALL
  LISTC ENT(CATALOG.CCAT) ALL
/*
//STEP5            EXEC  PGM=IDCAMS
//SYSPRINT    DD       SYSOUT=*
//SYSIN           DD       *
  EXAMINE NAME(CATALOG.ACAT) INDEXTEST NODATATEST
  EXAMINE NAME(CATALOG.ACAT) NOINDEXTEST DATATEST
  EXAMINE NAME(CATALOG.CCAT) INDEXTEST NODATATEST
  EXAMINE NAME(CATALOG.CCAT) NOINDEXTEST DATATEST
  DIAGNOSE ICFCATALOG INDATASET(CATALOG.ACAT)
  DIAGNOSE ICFCATALOG INDATASET(CATALOG.CCAT)
/*
```

The example shows how separate steps are used. Also note I have included the IDCAMS EXAMINE and DIAGNOSE commands prior to the back-up and also after the back-up. This ensures that I am aware of any structural issues that may exist in the catalog before and after back-up. The output below shows what I would expect to see if all is well:

```
  EXAMINE NAME(CATALOG.ACAT ) INDEXTEST NODATATEST
IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.ACAT ) NOINDEXTEST DATATEST
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 49 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 2715 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 1276 BYTES
IDC01722I 88 PERCENT FREE SPACE
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.CCAT) INDEXTEST NODATATEST
IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.CCAT) NOINDEXTEST DATATEST
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 9 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 380 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
```

```
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 534 BYTES
IDC01722I 97 PERCENT FREE SPACE
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  DIAGNOSE ICFCATALOG INDATASET(CATALOG.ACAT )
IDCAMS  SYSTEM SERVICES
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  DIAGNOSE ICFCATALOG INDATASET(CATALOG.CCAT)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

IDCAMS  SYSTEM SERVICES                                    TIME: 09:16:54

  EXPORT CATALOG.ACAT  TEMPORARY OUTFILE(BACKCAT1)
IDC0005I NUMBER OF RECORDS PROCESSED WAS 2714
IDC0594I PORTABLE DATA SET CREATED SUCCESSFULLY ON 11/03/04 AT 09:17:49
IDC1147I IT IS RECOMMENDED THAT DIAGNOSE AND EXAMINE BE RUN BEFORE
IDC1147I IMPORT OF CATALOG
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

IDCAMS  SYSTEM SERVICES                                    TIME: 09:17:53

  EXPORT CATALOG.CCAT TEMPORARY OUTFILE(BACKCAT2)
IDC0005I NUMBER OF RECORDS PROCESSED WAS 380
IDC0594I PORTABLE DATA SET CREATED SUCCESSFULLY ON 11/03/04 AT 09:17:54
IDC1147I IT IS RECOMMENDED THAT DIAGNOSE AND EXAMINE BE RUN BEFORE
IDC1147I IMPORT OF CATALOG
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

   IDCAMS  SYSTEM SERVICES

  EXAMINE NAME(CATALOG.ACAT ) INDEXTEST NODATATEST
IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.ACAT ) NOINDEXTEST DATATEST
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 49 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 2714 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 1276 BYTES
IDC01722I 88 PERCENT FREE SPACE
```

```
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.CCAT) INDEXTEST NODATATEST
IDC01700I INDEXTEST BEGINS
IDC01724I INDEXTEST COMPLETE - NO ERRORS DETECTED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  EXAMINE NAME(CATALOG.CCAT) NOINDEXTEST DATATEST
IDC01701I DATATEST BEGINS
IDC01709I DATATEST COMPLETE - NO ERRORS DETECTED
IDC01708I 9 CONTROL INTERVALS ENCOUNTERED
IDC01710I DATA COMPONENT CONTAINS 380 RECORDS
IDC01711I DATA COMPONENT CONTAINS 0 DELETED CONTROL INTERVALS
IDC01712I MAXIMUM LENGTH DATA RECORD CONTAINS 534 BYTES
IDC01722I 97 PERCENT FREE SPACE
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  DIAGNOSE ICFCATALOG INDATASET(CATALOG.ACAT )
IDCAMS  SYSTEM SERVICES                                    TIME: 09:18:01
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  DIAGNOSE ICFCATALOG INDATASET(CATALOG.CCAT)
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

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

It is vital that any issues are flagged. It is also critical that you get a successful creation of portable dataset message, as shown in the example above, to highlight that the catalog has backed up successfully. Note that the aliases related to the catalog are also exported with it. As an extra precaution, I also run a LISTCAT in STEP5 of the back-up job to produce output of the aliases associated with each catalog and write these to the third file of the output tape. This can then be used if required as part of the recovery. Note that on the back-up I have coded TEMPORARY, which specifies that the catalog is not to be deleted. The catalog is marked ‘temporary’ to indicate that another copy exists and that the original copy can be replaced. This is a required parameter when exporting an Integrated Catalog Facility catalog that cannot be exported with the PERMANENT parameter.

You can also use the DFSMSdss utility to back up catalogs. If you do this, more than one catalog can be backed up in a single back-up step and all can be stored on one back-up

dataset. The back-up will also contain any aliases associated with that catalog and the lock attribute of the catalog. The latter is important to note because, if the catalog was locked originally when dumped, it will be restored in a locked status. This might not be what you want.

It is also possible to use DFSMShsm to perform a back-up. This is achieved using management classes and can be a good method of automating back-ups. DFSMShsm actually invokes the IDCAMS EXPORT command in the background to perform the back-up. Aliases are also included in this back-up copy.

When backing up, do consider where the dataset being created will be catalogued. Obviously if it is catalogued in one of the catalogs you are backing up, you may not be able to access the back-up. Ideally, and importantly, catalog the back-ups in a separate catalog and also back that up and catalog its back-ups in a different catalog. Caution is the key factor to success should you have to recover the catalogs. Always assume the worst can happen!

OTHER BACK-UP FACTORS TO TAKE INTO ACCOUNT

One factor that will be vital as part of any forward recovery is the availability of the required SMF data. You must make certain that the required SMF data is being collected. This can be changed dynamically if required, but the key factor is that as a minimum the records detailed below are being created and backed up.

The critical SMF record type is shown, followed by its purpose:

- 36 – BCS successfully exported.
- 60 – VVR or NVR inserted, updated, or deleted.
- 61 – BCS entry defined.
- 65 – BCS entry deleted.
- 66 – BCS entry altered.

Master catalogs should not only be backed up, but there should also be an alternative master catalog and you should set up the necessary system members and processes to keep it in sync with the live master catalog. This operation is normally performed by the systems programmers at your site. If you do not have a shared system and utilize only a single MVS image, then a process to recover using stand-alone restore should be created.

Lastly on back-ups, you must take into consideration the VSAM volume dataset. The back-up of your catalog contains only data that resided in the BCS. It is the VVDS that contains detail about the structure of the dataset. The back-up and recovery of VVDSs is a major subject and not covered here. However, note that when recovering catalogs, in some cases the VVDS does need to be considered.

RECOVERY CONSIDERATIONS

Obviously, once you are successfully backing up catalogs, you then need to consider various factors relating to recovery. For example, if you need to recover, what is the latest back-up? If you use a generation dataset for the back-up copies, then it should be relatively easy to find out. Another option is to run a program against SMF to extract the type 36 record. This details exactly when a catalog was last backed up. How you recover depends on how you backed up. If you used IDCAMS Export to back up the catalog then you will use IDCAMS Import to get it back. Figure 1 shows the restore process with its associated back-up process. Also consider how, once you have restored, you will forward recover to ensure that any lost datasets are recovered.

As part of the recovery, you need to ensure that no-one can access the catalog during initial restore and forward recovery. A command to lock the catalog is provided and can be initiated using IDCAMS ALTER or via a catalog modify command. When a catalog is locked, any access to it will fail with a message saying the catalog is temporarily unavailable. Certain

Back-up	*Restore*	*Alias situation*
IDCAMS Export	IDCAMS Import	Defined if ALIAS coded on the IMPORT
DSS Logical Dump	DSS logical restore	Aliases redefined
DSS Full Volume Dump	DSS Full Volume or physical dataset restore	Aliases have to be manually recreated
HSM BACKDS	HSM Recover	Aliases redefined

Figure 1: Back-up, restore, and alias considerations

security requirements are required to issue the lock command successfully. The IGG.CATLOCK RACF profile must be set up and you require READ access to it to successfully initiate a LOCK. You also require ALTER access to the catalog being locked.

INTEGRATED CATALOG RECOVERY UTILITY

The Integrated Catalog Recovery utility requires a back-up of the catalog taken using Export. For forward recovery, it requires that SMF type 61, 65, and 66 records are being collected. The utility works by creating a file that is error free, and this can be used to actually replace the catalog that is in error. This is achieved using two programs. The ICFRRSV program extracts the required SMF records based on the CATALOG name that is being recovered. It also uses a start date and time and an end date and time to ensure that only the required records are extracted. The extracted data is then sorted and used as input to the second utility program, ICFRRAP. ICFRRAP also uses the exported copy of the catalog to produce a file containing the various updated ICF catalog records.

The file produced can then be used by IDCAMS Import to create a recovered catalog with all required entries.

The whole process is relatively simple and as long as you put into place the processes and procedures, along with a test of recovery for each catalog, you should feel relatively comfortable that you can get back to where you need to be.

The Integrated Catalog Recovery utilities do require you to do some manual work, such as locking the catalog being recovered, but overall it is useful and removes much of the complexity required to process the SMF records manually that normally results when a catalog is recovered.

After a successful recovery you do need to ensure the following is done:

1 Issue:

```
LISTCAT ENT(CATALOG.RECOVERED) ALL CAT(CATALOG.RECOVERED)
```

If this completes with a 0 condition code, you have successfully read the catalog record from the recovered catalog.

2 Unlock the recovered catalog. Use:

```
ALTER CATALOG.RECOVERED UNLOCK
```

3 Run an EXAMINE against both the INDEX and DATA components using:

```
EXAMINE NAME(CATALOG.RECOVERED) INDEXTEST -
                    NODATATEST
EXAMINE NAME(CATALOG.RECOVERED) NOINDEXTEST -
                    DATATEST
```

4 Back up the recovered catalog.

CONCLUSION

Catalog recovery is not something that just happens. It is essential that careful planning is put into place to ensure that the appropriate processes exist to back up the required environments and then to initiate recovery quickly if required.

To achieve this a reasonable understanding is required not only of catalogs but also of the various system utilities that can be used to assist in the processes.

IBM provides several base utilities to assist you and, if required, third-party products can also be utilized.

Further recommended reading before actually starting a catalog recovery project would be:

- SG24-5644: *IBM Redbook ICF Catalog Backup and Recovery*.
- SC26-7409: *DFSMS: Managing Catalogs*.
- SC26-7394: *DFSMS: Access Method Services for Catalogs*.
- SA22-7630: *Systems Management Facilities*.

Many people believe that their catalogs are safe yet have no back-up or recovery in place; or they do have processes and jobs that run but are not checking them for validity.

Elizabeth Bradley
Systems Programmer
Meerkat Computer Services (UK)

Measuring TSO command use

As you already know, TSO is conversational. Conversational, in a shared computing context among other things, means that each TSO user who is logged on to an MVS system has their own address space. Present within each TSO session's internal memory structure will be every component of a standard task. This includes things like TCBs (Task Control Blocks), an entry in the ASVT (Address Space Vector Table), an ASCB (Address Space Control Block), and JCL. Each TSO session also has its own unique mnemonic designator within

the scheme of MVS dispatching: TSU (Time-Sharing User). So, one can say that TSO has a direct one-to-one relationship – one user, one address space. Because of this, even on a large MVS mainframe, TSO consumes far too many finite resources to be a viable application platform for a large group of users. Therefore it was not a surprise that in the early 1990s there was a large push to off-load mainframe development to far less expensive personal computers. Perhaps, like so many other things in the distributed world of PCs, it has become clear that off-loading development is not quite as economical as first envisioned. In the meantime, mainframe time has become less expensive and mainframe hardware continues to modernize. Neither IBM nor the user community appreciated the extent to which TSO would be used. TSO has, in fact, gained unexpected acceptance as a development environment and production tool in MVS installations. There are few MVS installations without a significant TSO user community, ranging from systems programmers, to operations monitors, to application developers, to end users. A few years ago IBM reported that “most MVS installations see at least a 30% increase in TSO CPU use each year. This rate of growth appears to be continuing or even increasing as more TSO applications and 4GL products are installed”. If that is so, then understanding the status of TSO resources and applications processing for a healthy TSO environment is critical.

TSO response time is often the significant and only metric by which the efficiency of an installation is gauged. From the perspective of a TSO user, installation performance is often based on how well TSO responds to user interactions. There are several commercially-available tools that report TSO response times. It seems, however, that most of the problems with TSO performance are not caused by TSO. Most TSO response problems are caused by the interaction between TSO and MVS or between TSO and the rest of the system, or by other workloads on the system. The following are typical causes of poor TSO response time:

- Swap-in delaying TSO users.

- TSO users being denied access to a processor because of system overhead.
- TSO users being denied access to a processor because of workload with a higher priority.
- Excessive paging delaying TSO users.
- Waiting for I/O to complete delaying TSO users.
- Inappropriate WLM/SRM parameters delaying TSO users.
- Contention with other work in the system delaying TSO users.
- Contention for system resources among TSO users causing response delays.
- Inadequate processor power delaying TSO users.

It was observed that tuning TSO subsystem can often provide a reduction in TSO response time of 50% or more. What is noticeably absent from the standard TSO tuning approach is the changing and/or modifying of TSO applications themselves.

If you want to measure the effect and the system impact of a change on a given TSO application and see the service consumed (broken down into the four standard measurable categories: CPU, SRB, MSO, and IOC), a different tuning approach is needed. The simplest method is to use a third-party software package to measure the effect of a change. However, I'd like to describe a method that you can use with standard SMF facilities for collecting command response and resource usage so that you can monitor the TSO/E commands users issue and record the number of times a user issues a specific command or subcommand. For example, you can:

- Keep track of and compare how frequently certain commands at your installation are used and how many resources are consumed by these commands. You may want to provide better performance for the more commonly-used commands by placing them in LPALIB.

- Keep track of the number of times users issue TSO/E commands so you can bill users for their computer use.
- Audit the commands users issue to ensure that they do not violate security practices at your installation.

Before proceeding any further let us remember that RMF and SMF can identify only TSO commands or command processors included in the IEEMB846 module. The IBM-supplied module IEEMB846 contains a partial list of the TSO/E commands, prefixed subcommands, and aliases that are counted. Commands not specified in the IEEMB846 fall into a category of ***OTHER. Called programs all show up under the command 'CALL' and CLISTs show up under the command 'EXEC'. Also, some TSO/E products or, possibly, user applications currently do not count TSO/E commands. For example, the Interactive Problem Control System (IPCS) does not count TSO/E commands, but TSO/E subcommands are counted. The TSO/E command interface lets a user application avoid this problem. Therefore, if you want to get some information about a specific CLIST, you can write a command processor to invoke the CLIST. Then let users execute the command processor. For example, I've had several requests asking how to measure SPSS usage under TSO. The SPSS program counts under the CALL command and the SPSS CLIST counts under the EXEC command. You can write a command processor called SPSSTSO, ask users to execute SPSSTSO, then collect command usage on SPSSTSO. Note that the SMFTSOCM member of SYS1.SAMPLIB is provided so that the user can add or delete commands for the installation. The SMFTSOCM member contains the source code for the IBM-supplied IEEMB846.

How can one collect SMF command measurements? The following steps are needed:

- Create command processors for programs or CLIST/EXECs that you want to monitor.
- Define the commands that you want to monitor in CSECT IEEMB846.

- Modify SMF to collect type 32 records in member SMFPRM*xx* in SYS1.PARMLIB.
- Collect type 32 data.
- Process the type 32 records to get average CPU and I/O consumption for the commands.

Optionally, you can get more resource information by specifying 'DETAIL' in the SUBSYS(TSO) option in SMFPRM*xx*. This additional data includes TCB time, SRB time, the number of TGETs and TPUTs, SRM transaction count, EXCP count, and device connect time associated with the command. Obviously, DETAIL requires more CPU time and more virtual storage space since the data for the type 32 record is kept in CSA. Since the command segment itself is 40 bytes per command, a large TSO installation may require a significant amount of virtual storage for this data. Naturally, fewer commands executed take less overhead. On the other hand, you may want to collect and analyse one day's type 32 (DETAIL) data before deciding whether or not to collect type 32 (DETAIL) records every day.

In order to provide a starting point from which one can begin to measure and analyse TSO command use, I have coded a sample TSO command report writer. The code is a four-part stream. The first part (DEL32) is a clean-up step that deletes the files to be used in later steps. In the second step (DUMPTSO), SMF records 32 subtype 4 and type 34 are extracted from the SMF dataset to a file, which can be used as a base of archived records. In the next part (SORTTSO), previously extracted records (selection being defined by INCLUDE's condition) are sorted and copied to a file, which is the input to TSOCMD EXEC invoked in REXTSO step.

Record type 34 is written when the TSO/E logoff function processes a step termination. This record identifies the job by job name, log-on time and date, user identification, program name, and performance group number. The record contains operating information such as initiator start time, number of

TPUTs issued, number of TGETs satisfied, termination status, device allocation start time, problem program start time, components of CPU time, transaction active time (calculated as total transaction active time minus the accumulated transaction active time before this step's initialization), transaction residency time (which is the amount of time the transaction was in real storage), number of address space swap sequences (a swap sequence consists of a swap-out and swap-in of an address space), and storage related data (storage protect key, largest amount of storage used from top of private area, largest amount of storage used from bottom of private area and region size established). CPU time includes CPU time under SRBs and TCBs. CPU time under SRBs includes the CPU time for various supervisory routines that are dispatched via SRBs: locking routines, page resolution, swap control, cross-memory communications (WAIT, POST, I/O POST), and TQE scheduling. CPU time under TCBs (Task Control Blocks) includes the CPU time for all tasks that are dispatched via TCBs below the level of RCT.

CODE

```
//DEL32     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE hlq.R324.DATA
    SET MAXCC=Ø
/*
//DUMPTSO  EXEC PGM=IFASMFDP,REGION=ØM
//INDD     DD DSN=your.weekly.smf.dataset,DISP=SHR
//OUTDD    DD DSN=&&SMF32OUT,DISP=(NEW,PASS),
//         SPACE=(CYL,(25,5)),DCB=(your.weekly.smf.dataset)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    DATE(yyyyddd,yyyyddd)
    INDD(INDD,OPTIONS(DUMP))
    OUTDD(OUTDD,TYPE(32(4),34))
/*
//SORTTSO  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=&&SMF32OUT,DISP=SHR
//SMF      DD DSN=hlq.R324.DATA,
```

```
//          SPACE=(CYL,(6,1)),
//          DISP=(NEW,KEEP),
//          DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN    DD *
  COPY FROM(RAWSMF) TO(SMF) USING(SMFI)
//SMFICNTL DD *
*
* ELIMINATE HEADER AND TRAILER RECORDS
* SORT BY SESSION END DATE AND TIME
*
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,32,OR,6,1,BI,EQ,34)
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//REXTSO    EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50
//SYSEXEC   DD DISP=SHR,DSN=your.rexx.lib
//SMF       DD DSN=hlq.R324.DATA,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN   DD *
prof nopref
%TSOCMD
/*
```

TSOCMD EXEC

```
/* REXX EXEC to read and format SMF 32.4 and 34 records         */
/* These records are written at normal or abnormal termination  */
/* of a TSO session. The record length is variable.             */
/* Source :SYS1.MACLIB(IFASMFR3)                                */
ADDRESS TSO
numeric digits 16
*-------------------------------------------------------------*/
/* SUs/SEC and MIPS calculations                               */
/*-------------------------------------------------------------*/
cvt     = c2d(storage(10,4))                /* point to CVT         */
rmct    = c2d(storage(d2x(cvt+604),4))      /* point to RMCT        */
csd     = c2d(storage(d2x(cvt+660),4))      /* point to CSD         */
numcpu  = c2d(storage(d2x(csd+10),2))       /* point to # of CPUS   */
su      = c2d(storage(d2x(rmct+64),4))      /* cpu rate adjustment  */
susec   = format((16000000/su),7,2)         /* SUs per second       */
mips    = format((susec/48.5) * numcpu,6,2) /* SRM mips calculation */
/*-------------------------------------------------------------*/
/* Find TSO/E level - aka Sysvar("SYSTSOE")                    */
/*  -------------------------------------------------------------*/
cvttvt   = c2d(storage(d2x(cvt + 156),4))   /* point to TSO vect tbl*/
tsover   = storage(d2x(cvttvt+100),1)       /* point to TSO version */
tsorel   = storage(d2x(cvttvt+101),2)       /* point to TSO release */
tsomod   = storage(d2x(cvttvt+103),1)       /* point to TSO mod lvl */
tsorel   = format(tsorel)
```

```
tsolev   = tsover || '.' || tsorel || '.' || tsomod
userid=SYSVAR(SYSUID)
tsor = userid||'.cmd.rpt'       /* TSO Command Use report file       */
sesr = userid||'.ses01.rpt'     /* TSO Session summary report part 1 */
serr = userid||'.ses02.rpt'     /* TSO Session summary report part 2 */
ser3 = userid||'.ses03.rpt'     /* TSO Session summary report part 3 */
x = MSG('ON')
IF SYSDSN(tsor) = 'OK'
   THEN "DELETE "tsor" PURGE"
IF SYSDSN(sesr) = 'OK'
   THEN "DELETE "sesr" PURGE"
IF SYSDSN(serr) = 'OK'
   THEN "DELETE "serr" PURGE"
IF SYSDSN(ser3) = 'OK'
   THEN "DELETE "ser3" PURGE"
"ALLOC FILE(TSOCMD) DA("tsor")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(70,9) CATALOG",
   " REUSE RELEASE LRECL(172) RECFM(F B)"
"ALLOC FILE(SESSP1) DA("sesr")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(70,9) CATALOG",
   " REUSE RELEASE LRECL(235) RECFM(F B)"
"ALLOC FILE(SESSP2) DA("serr")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(70,9) CATALOG",
   " REUSE RELEASE LRECL(183) RECFM(F B)"
"ALLOC FILE(SESSP3) DA("ser3")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(70,9) CATALOG",
   " REUSE RELEASE LRECL(156) RECFM(F B)"
/*-------------------------------------------------------------*/
/* Header for TSO Command Use report                           */
/*-------------------------------------------------------------*/
rptc.1 = left('TSO Command Use report',50)
rptc.2 = left(' ',1,' ')
rptc.3 = left(' ',79)       left('Terminal',11)      left('SRM',6)   ,
         left('Connect',10) left('Terminal',8)
rptc.4 = left('Date',8)     left('User/Command',12)  left('JobID',8) ,
         left('Clock',11)   left('# cmds',9)         left('tcb',6)   ,
         left('srb',6)      left('cpu',6)            left('excp',5)  ,
         left('read',5)     left('write',5)          left('trans.',8),
         left('time',11)    left('i/o',7)            left('cpu/
comm',11),
         left('cpu/trans',11) left('mips/comm',13)   left('mips/
trans',10)
rptc.5 = left('-',170,'-')
 "EXECIO * DISKW TSOCMD (STEM rptc.)"
/*-------------------------------------------------------------*/
/* Header for TSO Session summary report  (part 1)             */
/*-------------------------------------------------------------*/
rpts.1 = left('TSO Session summary report  (part 1)',80)
rpts.2 = left(' ',1)
rpts.3 = left('Session end TOD',24) left('JobID',8) left('User',8) ,
```

```
         left('Terminal',9)         left('Clock',11)     ,
         left('rdr start',12)       left('rdr time',12)  ,
         left('logon enq.',12)      left('logging',12)   ,
         left('init',12)            left('dev. alloc',12),
         left('pgm fetch',11)       left('tot. cpu',9)   ,
         left('# comds',9)          left('cpu/comd',9)   ,
         left('# trans.',9)         left('cpu/trans',1Ø) ,
         left('excps',8)            left('connect',8) ,
         left('term. i/o',9)
rpts.4 = left('-',233,'-')
"EXECIO * DISKW SESSP1 (STEM rpts.)"
/*---------------------------------------------------------------*/
/* Header for TSO Session summary report  (part 2)               */
/*---------------------------------------------------------------*/
rptx.1 = left('TSO Session summary report  (part 2)',8Ø)
rptx.2 = left(' ',128,' ')           left('Delay',14) ,
         left('--      Service units    --',3Ø)
rptx.3 = left('Session end TOD',24)  left('User',8)   ,
         left('Terminal',11)         left('CC',5)     ,
         left('Clock',11)            left('excps',6)  ,
         left('term.i/o',9)          left('tcb',6)    ,
         left('srb',6)               left('cpu',7)    ,
         left('active time',13)      left('residency',9) ,
         left('duration',1Ø)         left('session',1Ø),
         left('cpu',9)               left('srb',8)     ,
         left('i/o ',4)              left('main stg.  ',14)

rptx.4 = left('-',18Ø,'-')
"EXECIO * DISKW SESSP2 (STEM rptx.)"
/*---------------------------------------------------------------*/
/* Header for TSO Session summary report  (part 3)               */
/*---------------------------------------------------------------*/
rptw.1 = left('TSO Session summary report  (part 3)',8Ø)
rptw.2 = left(' ',57,' ')            left('Memory used (K):',17),
         left('Page  &  Swap: ',22)  left('  Vio pages:',15)    ,
         left('Comm',5)              left('Lpa',4)             ,
         left('Hsp:',8)              left('Pages',6)           ,
         left('pg',7)                left('Page',4)
rptw.3 = left('Session end TOD',24)  left('User',8)   ,
         left('Terminal',11)         left('Clock',11) ,
         left('sys',4)               left('used',5)   ,
         left('rgn.',6)              left('in',4)     ,
         left('out',3)               left('swaps',5)  ,
         left('in',4)                left('out',4)    ,
         left('in',3)                left('out',3)    ,
         left('reclm',6)             left('in',4)     ,
         left('in',4)                left('in',3)     ,
         left('out',4)               left('stolen',6) ,
         left('miss',8)              left('sec.',4)
rptw.4 = left('-',155,'-')
```

```
 "EXECIO * DISKW SESSP3 (STEM rptw.)"
 'EXECIO * DISKR SMF ( STEM x. FINIS'
     do i = 1 to x.Ø
k = 1
smfxxrty = c2d(substr(x.i,2,1))              /* record type          */
 select
  when smfxxrty = 34  Then call type34
  otherwise do
smf32tme = smf(c2d(substr(x.i,3,4)))         /* record written time. */
                                    /* this is the time the session*/
smf32tme1= c2d(substr(x.i,3,4))              /* ended.               */
smf32dte = substr(c2x(substr(x.i,7,4)),3,5)  /* record written date  */
smf32sid = substr(x.i,11,4)                  /* system identification*/
smf32wid = substr(x.i,15,4)                  /* subsystem id         */
smf32stp = c2d(substr(x.i,19,2))             /* record subtype       */
/* ------------------------------------------------------------ */
/*       Self-defining section                                  */
/* ------------------------------------------------------------ */
smf32pof = c2d(substr(x.i,21,4))      /* offset to product segment   */
smf32pln = c2d(substr(x.i,25,2))      /* length of product segment   */
smf32pon = c2d(SUBSTR(x.i,27,2))      /* number of product segment   */
smf32iof = c2d(SUBSTR(x.i,29,4))           /* offset to id. segment  */
smf32iln = c2d(SUBSTR(x.i,33,2))           /* length of id. segment  */
smf32ion = c2d(substr(x.i,35,2))      /* number of id. segments      */
smf32cof = c2d(substr(x.i,37,4))      /* offset to tso comm. segment */
smf32cln = c2d(substr(x.i,41,2))      /* length of tso comm. segment */
smf32con = c2d(substr(x.i,43,2))      /* number of tso comm. segments*/
smf32cos = c2d(substr(x.i,45,4))      /* number of tso comm. segments*/
                                           /* in subsequent records  */
  if smf32pof >Ø & smf32pon > Ø  then do
     pof = smf32pof - 3
/* ------------------------------------------------------------ */
/*       Product section                                        */
/* ------------------------------------------------------------ */
smf32typ = c2d(substr(x.i,pof,2))        /* sub type id for type 32  */
                                    /* 1 - user interval, no detail*/
                                    /* 2 - user sess.end, no detail*/
                                    /* 3 - user interval, detail   */
                                    /* 4 - user session end, detail*/
smf32rvn = substr(x.i,pof+2,2)        /* record version number c'Ø1' */
smf32pnm = substr(x.i,pof+4,8)        /* product name 'tso for tso'  */
smf32osl = substr(x.i,pof+12,8)       /* mvs product name            */
smf32syn = substr(x.i,pof+2Ø,8)       /* system name                 */
smf32syp = substr(x.i,pof+28,6)       /* sysplex name                */
end
/* ------------------------------------------------------------ */
/*       User / session identification segment                  */
/*       Job  / session identification                          */
/* ------------------------------------------------------------ */
  if smf32iof >Ø & smf32ion > Ø  then do
```

```
     iof = smf32iof - 3
smf32jbn = substr(x.i,iof,8)          /* job / session name          */
smf32pgm = substr(x.i,iof+8,8)        /* program name                */
smf32stm = substr(x.i,iof+16,8)             /* step name            */
smf32uif = substr(x.i,iof+24,8)             /* user id. field       */
smf32jnm = substr(x.i,iof+32,8)             /* jes job number       */
smf32stn = c2d(substr(x.i,iof+40,2))        /* step  #              */
smf32pgn = c2d(substr(x.i,iof+44,2))    /* job performance group no. */
                                      /*  - valid only if workload   */
                                      /*    management compatibility */
                                      /*    mode in effect.          */
                                      /*  - zero if workload         */
                                      /*    management goal mode in  */
                                      /*    effect when the type 32  */
                                      /*    is generated.            */
smf32jpt = c2d(substr(x.i,iof+46,2))         /* jes input priority   */
smf32ast = smf(c2d(substr(x.i,iof+48,4)))  /* device alloc start time*/
smf32pps = smf(c2d(substr(x.i,iof+52,4))) /*problem program start time*/
smf32pps1= c2d(substr(x.i,iof+52,4))         /*                     */
smf32sit = smf(c2d(substr(x.i,iof+56,4)))    /* step initiation time */
smf32std =  substr(c2x(substr(x.i,iof+60,4)),3,5)
                                       /* step initiation date    */
smf32rst = smf(c2d(substr(x.i,iof+64,4)))     /* reader start time   */
smf32rst1= c2d(substr(x.i,iof+64,4))             /*                  */
smf32rsd = substr(c2x(substr(x.i,iof+68,4)),3,5) /* reader start date*/
smf32ret = smf(c2d(substr(x.i,iof+72,4)))     /* logon enqueue time  */
smf32ret1= c2d(substr(x.i,iof+72,4))             /*                  */
smf32red = substr(c2x(substr(x.i,iof+76,4)),3,5) /*logon enqueue date*/
smf32usr = substr(x.i,iof+80,20)             /* programmers name     */
smf32grp = substr(x.i,iof+100,8)             /* racf group id        */
smf32rud = substr(x.i,iof+108,8)             /* racf user id         */
smf32tid = substr(x.i,iof+116,8)      /* if racf is not active then 0*/
                                        /* also if this is not a  */
                                      /* terminal user then this  */
                                        /* field is zero          */
elaps    = cross(smf32tme1,smf32rst1)
logging  = smf(smf32pps1-smf32rst1)
readertm = smf(smf32ret1-smf32rst1)
end
/* ------------------------------------------------------------ */
/*       TSO command segment                                    */
/* ------------------------------------------------------------ */
totcmd = 0 ; totcpu = 0 ; totexcp = 0 ; totget = 0 ; totput = 0 ;
totdasd= 0 ; tottcb = 0 ; totsrb  = 0 ; totsrm = 0 ; totio  = 0
  do j = 0 to smf32con -1
    cof = (smf32cof + (j*smf32cln)) - 3
/* ------------------------------------------------------------ */
/*      Basic command section                                   */
/* ------------------------------------------------------------ */
smf32cmd = substr(x.i,cof,8)                 /* tso command name    */
```

```
smf32cnt = c2d(substr(x.i,cof+8,4))          /* count of commands    */
totcmd   = totcmd + smf32cnt
/* ------------------------------------------------------------ */
/*      Detail command section                                  */
/* ------------------------------------------------------------ */
smf32tcb = c2d(substr(x.i,cof+12,4))*Ø.Ø1    /* tcb time for command */
smf32srb = c2d(substr(x.i,cof+16,4))*Ø.Ø1    /* srb time for command */
smf32tgt = c2d(substr(x.i,cof+2Ø,4))         /* tgets for command    */
smf32tpt = c2d(substr(x.i,cof+24,4))         /* tputs for command    */
smf32trn = c2d(substr(x.i,cof+28,4))    /* transactions for command  */
smf32exp = c2d(substr(x.i,cof+32,4))         /* excps for command    */
smf32tct = c2d(substr(x.i,cof+36,4))*128E-3  /* total device connect */
smf32tct1= c2d(substr(x.i,cof+36,4))*128E-6  /*          time        */
smf32dsf = c2d(substr(x.i,cof+4Ø,4))         /* detail section flags */
/* ------------------------------------------------------------ */
/* Total terminal I/O                                           */
/* ------------------------------------------------------------ */
 select
    when smf32tgt > Ø | smf32tpt> Ø then ,
    termio = smf32tgt + smf32tpt
    otherwise termio =' '
 end
/* ------------------------------------------------------------ */
/* Get totals for a session                                     */
/* ------------------------------------------------------------ */
cpu    = (smf32tcb + smf32srb)
totcpu = totcpu  + cpu
tottcb = tottcb  + smf32tcb
totsrb = totsrb  + smf32srb
totexcp= totexcp + smf32exp
totget = totget  + smf32tgt
totput = totput  + smf32tpt
totsrm = totsrm  + smf32trn
totdasd= totdasd + smf32tct1
totio  = totio   + smf32tgt + smf32tpt
/* ------------------------------------------------------------ */
/*  CPU cost of a transaction                                   */
/* ------------------------------------------------------------ */
select
 when totsrm > Ø then cputran= totcpu  / totsrm
 otherwise            cputan = Ø
end
/* ------------------------------------------------------------ */
/*  CPU cost of a command                                       */
/* ------------------------------------------------------------ */
select
 when totcmd > Ø then cpucmd = totcpu  / totcmd
 otherwise            cpucmd = Ø
end
select
```

```
 when cpu > 0 then cpucm = cpu  / smf32cnt
 otherwise         cpucm  = 0
end
/* ------------------------------------------------------------ */
/*   Get a transaction cpu ratio                                */
/* ------------------------------------------------------------ */
select
 when smf32trn > 0 then ratio  = format(cpu/smf32trn,7,5)
 otherwise              ratio  =  ' '
end
/* ------------------------------------------------------------ */
/*   Get a transaction mips ratio                               */
/* ------------------------------------------------------------ */
select
 when smf32trn > 0 then ratim  = format(smf32trn/mips,9,9)
 otherwise              ratim  =  ' '
end
select
 when totsrm   > 0 then ratimt = format(totsrm/mips,9,9)
 otherwise              ratimt =  ' '
end
/* ------------------------------------------------------------ */
/*   Get a command mips ratio                                   */
/* ------------------------------------------------------------ */
select
 when cpu > 0 then mipss  = format(smf32cnt/mips,12,11)
 otherwise         mipss  =  ' '
end
select
 when totcpu  > 0 then mipsst = format(totcmd/mips,12,11)
 otherwise         mipsst =  ' '
end
/* ------------------------------------------------------------ */
/*  Command detail line is being formated                       */
/* ------------------------------------------------------------ */
det.k = left(' ',11,' ') left(smf32cmd,32), /* tso command name      */
        right(smf32cnt,4)  ,                /* count of commands     */
        right(smf32tcb,6)  ,                /* tcb time for command  */
        right(smf32srb,6)  ,                /* srb time for command  */
        right(cpu,6)       ,                /* cpu time for command  */
        right(smf32exp,6)  ,                /* excps for command     */
        right(smf32tgt,5)  ,                /* tgets for command     */
        right(smf32tpt,5)  ,                /* tputs for command     */
        right(smf32trn,5)  ,             /* srm transactions/command */
        right(smf32tct1,11),             /* total device connect time*/
        right(termio,6)    ,                /* total terminal i/o    */
        format(cpucm,7,5)  ,                /* cpu per command       */
        right(ratio,13)    ,             /* cpu per transaction ratio*/
        right(mipss,13)    ,                /* cpu/mips              */
        right(ratim,13)                  /* mips/transaction ratio   */
```

```
    k = k +1
 end
drop det.k
/* ------------------------------------------------------------ */
/*  Session command totals line is being formatted              */
/* ------------------------------------------------------------ */
 tot.1 = left(date('n',smf32dte,'j'),11),   /* session end date      */
         left(smf32jbn,8)   ,               /* job / session name    */
         left(smf32jnm,8)   ,               /* jes job number        */
         right(elaps,12)    ,               /* session clock time    */
         right(totcmd,6)    ,               /* tot.# of commands     */
         right(tottcb,6)    ,               /* tot. tcb time         */
         right(totsrb,6)    ,               /* tot. srb time         */
         right(totcpu,6)    ,               /* tot. cpu time         */
         right(totexcp,6)   ,               /* tot. # of excp        */
         right(totget,5)    ,            /* tot. # of terminal reads */
         right(totput,5)    ,            /* tot. # of terminal writes*/
         right(totsrm,5)    ,            /* tot. # of srm transaction*/
         right(totdasd,11)  ,            /* tot. device conn. time   */
         right(totio,6)     ,            /* session terminal i/o     */
         format(cpucmd,7,5) ,            /* cpu per commands-session */
         format(cputran,7,5),            /* cpu per transaction - ses*/
         right(mipsst,13)   ,            /* cpu/mips                 */
         right(ratimt,13)                /* mips/transaction ratio   */
/* ------------------------------------------------------------ */
/*   Session summary (part one)                                 */
/* ------------------------------------------------------------ */
sess.1 = left(date('n',smf32dte,'j'),11) ,  /* session end date      */
         left(smf32tme,12)  ,               /* session end time      */
         left(smf32jnm,8)   ,               /* jes job number        */
         left(smf32jbn,8)   ,               /* job / session name    */
         left(smf32tid,8)   ,            /* terminal symbolic name   */
         right(elaps,12)    ,            /* session clock time       */
         left(smf32rst,12)  ,            /* reader start time        */
         left(readertm,12)  ,            /* reader time              */
         left(smf32ret,12)  ,               /* logon enqueue time    */
         left(logging,12)   ,               /* logging on time       */
         left(smf32sit,12)  ,               /* step initiation time  */
         left(smf32ast,12)  ,            /* device alloc start time  */
         left(smf32pps,12)  ,            /* program fetch event      */
         right(totcpu,6)    ,            /* tot. cpu (tcb+srb) time  */
         right(totcmd,6)    ,            /* tot. # of commands       */
         format(cpucmd,7,5) ,            /* cpu per a command        */
         right(totsrm,5)    ,            /* srm transaction count    */
         format(cputran,7,5),            /* cpu/a srm transaction    */
         right(totexcp,6)   ,            /* excps count              */
         right(totdasd,11)  ,            /* total io connect time    */
         right(totio,6)                  /* session terminal i/o     */
"EXECIO * DISKW TSOCMD (STEM tot.)"
"EXECIO * DISKW TSOCMD (STEM det.)"
```

```
"EXECIO * DISKW SESSP1 (STEM sess.)"
k = Ø
totcmd = Ø ; totcpu = Ø ; totexcp = Ø ; totget = Ø ; totput = Ø ;
totdasd= Ø ; tottcb = Ø ; totsrb  = Ø ; totsrm = Ø ; totio  = Ø
   end
  end
 end
n.1=left(' ',1,' ')
n.2=left('Completion code:',8Ø)
n.3=left("X'Øccc' indicates system ABEND where ccc is the system",8Ø)
n.4=left('        ABEND code. (See z/OS MVS System Codes.)',8Ø)
n.5=left("X'8ccc' indicates user ABEND where ccc is the user",8Ø)
n.6=left('        ABEND code.',8Ø)
n.7=left("X'nnn'  indicates normal completion where nnn is the",8Ø)
n.8=left('        contents of the 2 low-order bytes in register 15 at
end.',8Ø)
n.9=left("X'ØØØ'  indicates either: (1) the job step was flushed
(not",8Ø)
n.1Ø=left('        processed) because of an error during allocation, or
(2)',8Ø)
n.11=left('        normal job completion with a return code of Ø.',8Ø)
n.12=left(' ',1,' ')
n.13=left('Use this field in conjunction with the termination',8Ø)
n.14=left('indicator field.',8Ø)
"EXECIO * DISKW SESSP2 (STEM n.)"
 /* Close & free all allocated files  */
  "EXECIO Ø DISKW TSOCMD(FINIS "
  "EXECIO Ø DISKW SESSP1(FINIS "
  "EXECIO Ø DISKW SESSP2(FINIS "
  "EXECIO Ø DISKW SESSP3(FINIS "
  say
  say 'TSO Command Use report file dsn ...:'tsor
  say 'TSO Session summary report part 1..:'sesr
  say 'TSO Session summary report part 2..:'serr
  say 'TSO Session summary report part 3..:'ser3
  say
  "FREE FILE(SMF TSOCMD SESSP1 SESSP2 SESSP3)"
exit
TYPE34:
tivrcdts = smf(c2d(substr(x.i,3,4)))    /*time stamp tod .Ø1 secs    */
endtime  =     c2d(substr(x.i,3,4))
tivrcdte = substr(c2x(substr(x.i,7,4)),3,5) /*        date ØØyydddf  */
tivcpuid = substr(x.i,11,4)                 /*system identification  */
tivuif   = substr(x.i,15,8)              /*user identification field */
tivontme = smf(c2d(substr(x.i,23,4)))       /*logon time tod .Ø1 secs*/
starttme = c2d(substr(x.i,23,4))            /*                       */
tivondte = substr(c2x(substr(x.i,27,4)),3,5)/*       date ØØyydddf   */
tivudata = substr(x.i,31,8)                 /*resv for user          */
tivinvsq = c2d(substr(x.i,39,1))            /*step sequence  #       */
tivsit   = smf(c2d(substr(x.i,4Ø,4)))       /*tod step initiation    */
```

```
tivoutct = c2d(substr(x.i,44,4))            /*line out count         */
tivinct  = c2d(substr(x.i,48,4))            /*line in count          */
tivstat  = c2d(substr(x.i,52,2))          /*step termination status  */
tivpri   = c2d(substr(x.i,54,1))     /*step dispatching priority      */
tivprgnm =     substr(x.i,55,8)      /*name of program invoked        */
tivinvnm =     substr(x.i,63,8)      /*step (proc) name               */
tivrsv5  = c2d(substr(x.i,71,2))     /*reserved                       */
tivsyst  = c2d(substr(x.i,73,2))     /*syst area used, top pri area   */
tivmcre  = c2d(substr(x.i,75,2))     /*core actually used in  1k blks */
tivrvc   = c2d(substr(x.i,77,2))     /*reserved                       */
tivefrgn = c2d(substr(x.i,79,4))     /*effective rgn size in 1k blocks*/
termio   = tivinct + tivoutct        /* terminal i/o                  */
tivspk   = c2d(substr(x.i,83,1))     /*storage protect key            */
tivsti   = x2b(c2x(substr(x.i,84,1))) /*step termination indicators:  */
                                       /*bitØ - reserved             */
bit1 = substr(tivsti,2,1)            /*bit1 - cancelled - exit iefujv */
bit2 = substr(tivsti,3,1)            /*bit2 - cancelled - exit iefuji */
bit3 = substr(tivsti,4,1)            /*bit3 - cancelled - exit iefusi */
                                     /*bit4 - reserved                */
bit5 = substr(tivsti,6,1)            /*bit5 - step is to be restarted */
bit6 = substr(tivsti,7,1)            /*bit6 - Ø=normal completion     */
                                     /*       1=abend                 */
                                     /*bit7 - step flushed            */
bit7 = substr(tivsti,8,1)
 select
  when bit1 = 1 then btt1 ="iefujv"       /* cancelled by exit iefujv */
  otherwise          btt1=""
 end
 select
  when bit2 = 1 then btt2 ="iefuji"       /* cancelled by exit iefuji */
  otherwise          btt2=""           /* sessions cancelled by iefuji*/
 end                                    /* iefusi will not be processed*/
 select                                            /* bit7 will be on */
  when bit3 = 1 then btt3 ="iefusi"       /* cancelled by exit iefuj  */
  otherwise          btt3=""               /* bit7 will be on        */
 end
 select
  when bit5 = 1 then btt5 ="restart"        /* step is to be restarted*/
  otherwise          btt5=""
 end
 select                                  /* if Ø, then normal completion */
  when bit6 = 1 then btt6 ="Abend "  /*if 1, then abnormal end of task*/
  otherwise          btt6=""         /* (abend) will occur. if step   */
 end                               /* completion code equals Ø322 or */
                                   /* Ø522, iefutl caused the abend. */
                                   /* if step completion code equals */
                                   /* Ø722, iefuso caused the abend. */
 select
  when bit7 = 1 then btt7 ="flush"    /* if Ø, then normal completion */
  otherwise          btt7=""              /* if 1, step was flushed  */
```

```
 end
bit=btt1||btt2||btt3||btt5||btt6||btt7
tivrv1   = c2d(substr(x.i,85,2))            /*reserved                */
tivast   = smf(c2d(substr(x.i,87,4)))       /*alloc. start time       */
tivppst  = smf(c2d(substr(x.i,91,4)))      /*problem prog. start time */
tivrv2   = c2d(substr(x.i,95,1))            /*reserved                */
tivsrbt  = c2d(substr(x.i,96,3))*Ø.Ø1  /*step cpu under srb(.Ø1 sec) -*/
                                          /*includes the cpu time for */
                                        /*various supervisory routines*/
                                        /*that are dispatched via srbs*/
/* ------------------------------------------------------------ */
/*   Data validity check                                        */
/* ------------------------------------------------------------ */
tivrin   = x2b(c2x(substr(x.i,99,2)))       /*record indicators       */
                                            /*bitØ-3 reserved         */
                                    /*4 =1 field tivcputm not valid  */
                                    /* for sp41Ø,see type 3Ø record  */
                                    /*     when actjtime is >3 bytes */
                                    /*5 =1 device data not recorded  */
                                    /*6 =1 possible error in         */
                                    /*     device entry data         */
                                    /*7 =Ø stor is virt              */
                                    /*  =1 stor is real              */
                                    /*8-15 reserved                  */

bt4 = substr(tivrin,5,1)
 select
  when bt4 = 1 then btx4 ="not valid tcbtm"
                                    /* field tivcputm is not valid.  */
  otherwise         btx4 ="valid tcbtm / "  /*an overflow condition is*/
 end                                   /* when the length > 3 bytes. */
                                    /* this condition is not recorded*/
                                    /* in the type34 record          */
bt5 = substr(tivrin,6,1)
 select                             /* 1: device data not recorded.  */
  when bt5 = 1 then btx5 ="no dev.data"     /*when there are more than*/
  otherwise         btx5 ="dev.data ok /"   /*1635 DD statements,     */
 end                  /* device data is not collected for type 34 rec.*/

bt6 = substr(tivrin,7,1)
 select
  when bt6 = 1 then btx6 ="wrong excp"  /* 1: excp count may be wrong */
  otherwise         btx6 ="excp ok /"
 end

bt7 = substr(tivrin,8,1)
 select
  when bt7 = 1 then btx7 ="real stor."      /* storage type           */
  otherwise         btx7 ="virt.stor."
 end
```

```
bt=btx4||btx5||btx6||btx7
clock    = cross(endtime,starttme)
tivrlct  = c2d(substr(x.i,101,2))       /*offset to relocate section  */
tivrlct  = tivrlct +1
tivvar   = c2d(substr(x.i,103,2))       /*length of excp count fields */
                                        /*(including these two bytes) */
n = (tivvar -2)/8                           /*  # of excp fields      */
/* ------------------------------------------------------------- */
/*  Execute channel program (EXCP) section                       */
/*                                                               */
/*  dataset access information:                                  */
/*  note: virtual i/o devices are identified by the following:   */
/*        device class 0                                         */
/*        unit type 0                                            */
/*        device  number  x'7fff'                                */
/* ------------------------------------------------------------- */
offset = 105
totex = 0
do j =0 to n-1
   incr = (offset + (j*8))
tivdevc  = c2d(substr(x.i,incr,1))                 /* device class   */
tivutyp  = c2d(substr(x.i,incr+1,1))               /*    unit type   */
tivcuad  = c2d(substr(x.i,incr+2,2))               /*  device number */
tivnexcp = c2d(substr(x.i,incr+4,4))               /*    excp count  */
totex  = totex + tivnexcp                          /*total excp count*/
end
/* ------------------------------------------------------------- */
/*   Accounting section                                          */
/* ------------------------------------------------------------- */
acc      = 103 + tivvar
tivvara  = c2d(substr(x.i,acc,1))   /*length of cpu and acct. section */
                                    /*(not including this byte)       */
tivcputm = c2d(substr(x.i,acc+1,3))*0.01    /* field includes the cpu */
                          /* time for all tasks that are dispatched  */
                                    /* via tcbs below the level of rct*/
tivnbrac = c2d(substr(x.i,acc+4,1))         /* # of accounting fields */
/* ------------------------------------------------------------- */
/*       Relocate section                                        */
/* ------------------------------------------------------------- */
tivpgin  = c2d(substr(x.i,tivrlct,4))       /*     # of page-ins      */
tivpgout = c2d(substr(x.i,tivrlct+4,4))     /*     # of page-outs     */
tivrgns  = c2d(substr(x.i,tivrlct+8,4))     /*     # of swaps         */
tivsin   = c2d(substr(x.i,tivrlct+12,4))    /* # of tso swap page-ins */
tivsout  = c2d(substr(x.i,tivrlct+16,4))    /* # of tso swap page-outs*/
tivvpi   = c2d(substr(x.i,tivrlct+20,4))    /*     vio page ins       */
tivvpo   = c2d(substr(x.i,tivrlct+24,4))    /*     vio page outs      */
tivsst   = c2d(substr(x.i,tivrlct+28,4))    /*     step service time  */
tivact   = c2d(substr(x.i,tivrlct+32,4))    /*     step active time   */
tivact   = format(tivact*1024E-6,5,2)       /*step active time (sec.):*/
                                    /* calculated as total job trans. */
```

```
                                    /* active time - the accumulated  */
                                    /* transaction active time before */
                                    /* this step's initialization.    */
tivpgno  = c2d(substr(x.i,tivrlct+36,2))    /* performance  #         */
                                            /* - valid only if wlm    */
                                            /*   compatibility        */
                                            /*   mode in effect.      */
                                            /* - zero if wlm          */
                                            /*   goal mode in         */
                                            /*   effect.              */
tivtrant = c2d(substr(x.i,tivrlct+38,4)) /*step trans.residency time: */
tivtrant = format(tivtrant*1Ø24E-6,5,2)     /* amount of time the     */
                                 /* transaction  was in real storage. */
tivcpm   = c2d(substr(x.i,tivrlct+42,4))    /* cread page miss count  */
tivrclam = c2d(substr(x.i,tivrlct+46,4))    /*    # of vio reclaims   */
tivcpgin = c2d(substr(x.i,tivrlct+5Ø,4))    /*    # of common page-ins*/
tivhspi  = c2d(substr(x.i,tivrlct+54,4))    /*hiperspace page-in count*/
tivpgstl = c2d(substr(x.i,tivrlct+58,4))    /*    # of pages stolen   */
                                            /*    from this memory    */
tivpgsec = c2d(substr(x.i,tivrlct+62,8))    /* count of page seconds  */
tivpgsec = tivpgsec*Ø.ØØ1                   /* (unit is 1 page ms):   */
                                    /* the number of pages used by    */
                                    /* this step times the processing */
                                    /* time it held that no. of pages */
tivlpai  = c2d(substr(x.i,tivrlct+7Ø,4))    /*    lpa page ins        */
tivhspo  = c2d(substr(x.i,tivrlct+74,4))    /* hiperspace page-out #  */
tivcpus  = c2d(substr(x.i,tivrlct+78,4))    /*    step cpu su         */
tiviocs  = c2d(substr(x.i,tivrlct+82,4))    /*    step i/o su         */
tivmsos  = c2d(substr(x.i,tivrlct+86,4))    /*  step main storage su  */
tivsrbs  = c2d(substr(x.i,tivrlct+9Ø,4))    /*    step srb su         */
tivtsn   =     substr(x.i,tivrlct+94,8)     /* terminal symbolic name */
cpu34    = tivcputm + tivsrbt               /*    total cpu time      */
dspdlytm = tivtrant - cpu34

/* Delay duration means that task was resident but was not
   dispatched. This is the time the task was resident in memory
   but was not executing instructions. It includes delay for CPU
   dispatch, delay for I/O, and/or delay due to page faults. One
   should check the DLY report on RMFIII. The primary delay reasons
   are broken down into processor, device, storage, subsystem,
   operator, and enqueue.                                         */

/* ------------------------------------------------------------- */
/*   Session summary (part two)                                  */
/* ------------------------------------------------------------- */
ses.1 = left(date('n',tivrcdte,'j'),11) ,   /* session end date      */
        left(tivrcdts,12)               ,   /* session end time      */
        left(tivuif,8)                  ,   /* user identification   */
        left(tivtsn,8)                  ,   /* terminal symbolic name*/
        right(tivstat,5)                ,   /* completion code       */
```

```
        right(clock,12)                 ,   /* session clock time    */
        right(totex,7)                  ,   /* excps count           */
        right(termio,7)                 ,   /* session terminal i/o  */
        right(tivcputm,6)               ,   /* tot. tcb time         */
        right(tivsrbt,6)                ,   /* tot. srb time         */
        right(cpu34,6)                  ,   /*tot. cpu (tcb+srb) time */
        right(tivact,12)                ,   /* active time           */
        right(tivtrant,12)              ,   /* residency time        */
        right(dspdlytm,8)               ,   /* delay duration        */
        right(tivsst,8)                 ,   /* step service su       */
        right(tivcpus,8)                ,   /* step cpu su           */
        right(tivsrbs,8)                ,   /* step srb su           */
        right(tiviocs,8)                ,   /* step i/o su           */
        right(tivmsos,8)                    /* step main storage su  */
"EXECIO * DISKW SESSP2 (STEM ses.)"
/* ------------------------------------------------------------ */
/*   Session summary (part 3: storage use, paging & swapping)   */
/* ------------------------------------------------------------ */
sew.1 = left(date('n',tivrcdte,'j'),11) ,   /* session end date      */
        left(tivrcdts,12)               ,   /* session end time      */
        left(tivuif,8)                  ,   /* user identification   */
        left(tivtsn,8)                  ,   /* terminal symbolic name*/
        right(clock,12)                 ,   /* session clock time    */
        right(tivsyst,5)                ,   /* sys area used, top pri*/
        right(tivmcre,5)                ,  /*core actually used in  1k*/
        right(tivefrgn,5)               ,   /*effective rgn size in 1k*/
        right(tivpgin,4)                ,   /* page-in               */
        right(tivpgout,4)               ,   /* page-out              */
        right(tivrgns,4)                ,   /* swaps                 */
        right(tivsin,4)                 ,   /* swap page-in          */
        right(tivsout,4)                ,   /* swap page-out         */
        right(tivvpi,4)                 ,   /* vio page ins          */
        right(tivvpo,4)                 ,   /* vio page outs         */
        right(tivrclam,4)               ,   /* vio reclaims          */
        right(tivcpgin,4)               ,   /* common page-ins       */
        right(tivlpai,4)                ,   /* lpa page ins          */
        right(tivhspi,4)                ,   /* hiperspace page-in    */
        right(tivhspo,4)                ,   /* hiperspace page-out   */
        right(tivpgstl,4)               ,   /* pages stolen          */
        right(tivcpm,4)                 ,   /* cread page miss count */
        right(tivpgsec,10)                  /* count of page seconds */
"EXECIO * DISKW SESSP3 (STEM sew.)"
return
CROSS: procedure
/* ------------------------------------------------------------ */
/*        Cover the midnight crossover                          */
/* ------------------------------------------------------------ */
arg endtime,startime
  select
```

```
   when endtime > startime then nop
   otherwise  endtime = endtime + 8640000
  end
diftm = smf(endtime - startime)
return diftm

SMF: procedure
/* REXX - convert a SMF time to hh:mm:ss:hd format   */
arg time
    time1 = time % 100
    hh    = time1 % 3600
    hh    = RIGHT("0"||hh,2)
    mm    = (time1 % 60) - (hh * 60)
    mm    = RIGHT("0"||mm,2)
    ss    = time1 - (hh * 3600) - (mm * 60)
    ss    = RIGHT("0"||ss,2)
    fr    = time //  1000
    fr    = RIGHT("0"||fr,2)
    rtime = hh||":"||mm||":"||ss||":"||fr
    return rtime
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

A closer look at the internals of a load module

Very often, in the company where I work, I have had to examine the contents of an executable module.

In practically every case the use of the IBM utility AMBLIST has proven itself to be irreplaceable.

For example, by analysing the CSECTs that make up a module, I can discover:

- Whether a COBOL module was written for LE (CSECT CEE), for COBOL2 NORES (CSECT IGZ), or COBOL1 NORES (CSECT ILBO).
- Whether the module makes any CALLs to other programs and if these are static or dynamic (CSECT $UNRESOLVED).

- The linkage editor version and date of assembly.
- All of the module's attributes (reusability, addressing, etc).

To this end I wrote a code snippet in REXX that can be used under ISPF, in any member list, by simply typing 'xamb' to the left of the member name.

Having the commands AMBLIST/LISTLOAD and AMBLIST/LISTIDR execute in foreground mode guarantees a complete and easy-to-read output.

```
/* REXX ---------------------------------------------- REXX */
/* REXX XAMB - FOREGROUND MODE COMMAND.                 REXX */
/* REXX  - Structure analysis of an executable module -  REXX */
/* REXX ---------------------------------------------- REXX */
 TRACE OFF
 PARSE ARG DSN
 MSG_STATUS = MSG("OFF")
 ADDRESS TSO
         "PROF NOPREF "
/* -------------------------- */
 X = LISTDSI(DSN)
 LIB_INP = SYSDSNAME
 DA = POS('(',DSN) + 1
 DB = POS(')',DSN)
 LM = (DB - DA)
 MEM_INP = SUBSTR(DSN,DA,LM)
 UTSO    = SYSVAR(SYSUID)
 AMB_OUT = UTSO".AMBLIST.OUTPUT"
/* -------------------------- */
         "FREE F(SYSIN SYSUDUMP SYSPRINT)"
         "ALLOC F(SYSIN) UNIT(VIO) RECFM(F B) LRECL(8Ø) NEW DELETE"
 RECSYS = "     LISTLOAD MEMBER="MEM_INP
  QUEUE RECSYS
 RECSYS = "     LISTIDR  MEMBER="MEM_INP
  QUEUE RECSYS
  QUEUE ''
  "EXECIO * DISKW SYSIN (FINIS"
/* -------------------------- */
 RC_LIST = SYSDSN(AMB_OUT)

              IF RC_LIST = 'OK'
               THEN " DEL '"AMB_OUT"' "

         "ALLOC F(SYSPRINT) UNIT(CKPT) TRACKS
          RECFM(F B A) LRECL(121)  SPACE(7 5)
```

```
          NEW CATALOG DSNAME("AMB_OUT")"

         "FREE F(SYSLIB)"
         "ALLOC F(SYSLIB) DA("LIB_INP")  SHR"
         "ALLOC DUMMY DD(SYSUDUMP)"
ADDRESS LINKMVS
         "AMBLIST"
/* ---------------------------- */
ADDRESS ISPEXEC
         "EDIT DATASET("AMB_OUT")"
/* ---------------------------- */
ADDRESS TSO
         "FREE F(SYSIN SYSUDUMP)"
/* ---------------------------- */
EXIT
```

Massimo Ambrosini
Systems Programmer (Italy)

DFSORT enhancements introduced with PTF UQ90053 and incorporated into DFSORT V1.5

DFSORT is IBM's high performance sort, merge, copy, analysis, and reporting product. DFSORT is an optional program product and many sites may have other products such as Syncsort installed. Similar functionality does exist in those products as well.

This article discusses new features for DFSORT for join and match operations, sampling, repeating, and distributing records, arithmetic operations using numeric fields and decimal constants, longer fields for sub-string searches, easier migration from other sort products, and more.

This article is intended to highlight the new features so that you can exploit them at your site. The features also affect the ICETOOL product, which is another program that can be used as part of the DFSort suite of programs.

A brief summary of all the enhancements is shown at the end of the article.

USING with SELECT

The USING(xxxx) operand can now be used with ICETOOL's SELECT operator to process DFSORT control statements. This is similar to INCLUDE, OMIT, and OUTFIL for a SELECT operation. USING(xxxx) is optional for SELECT. The TO(outdd) operand and the DISCARD(savedd) operand must be specified even if the USING(xxxx) operand is specified.

The DFSORT control statements in the associated xxxxCNTL file are used if USING(xxxx) is specified with SELECT. You can use control statements and options in the xxxxCNTL dataset such as INCLUDE, OMIT, OPTION, and OUTFIL to eliminate records, reformat records, and generally to manipulate records. When ICETOOL makes the call to the DFSORT program, it will pass control statements and options that are appropriate for the SELECT operation being performed. You do, however, need to be careful because, if coding is done incorrectly, abends may occur. IBM does specify that you should not supply your own DFSORT INREC, MODS, OUTREC, or SORT statements. Also, if you specify TO(outdd) without DISCARD(savedd), you can further process the outdd records after SELECT processing using one, and only one, OUTFIL statement. If your installation defaults for dynamic allocation are inappropriate for a SELECT operator, you can specify USING(xxxx) and take one of the following actions:

- Override the DYNALLOC option using an OPTION control statement such as:

  ```
  OPTION DYNALLOC=(,8) in the xxxxCNTL data set.
  ```

- Use xxxxWKdd DD statements to override the use of dynamic allocation.

Description of the USING operand

The USING operand specifies the first four characters of the DDname for the control statement dataset to be used by

DFSORT for the operation. xxxx must be four characters, which are valid in a DDname of the form xxxxCNTL. xxxx must not be SYSx. If USING is specified, an xxxxCNTL DD statement must be present and the control statements must be contained in it. These should:

- Conform to the rules for DFSORT's SORTCNTL dataset.
- Generally be used only for an INCLUDE or OMIT statement, comment statements, or appropriate OUTFIL statements.

LISTSDB and LISTNOSDB

Two new operands that have been introduced are LISTSDB and LISTNOSDB. These are used with ICETOOL's DEFAULTS, DISPLAY, and OCCUR operators, and allow you to control the use of system-determined optimum blocksize for LIST datasets.

ICETOOL's DEFAULTS, DISPLAY, and OCCUR operators each produce output in a list dataset associated with a LIST(listdd) operand. If the BLKSIZE for the list dataset is not available and the system-determined optimum blocksize can be used, the BLKSIZE will be set as directed by the new LISTSDB or LISTNOSDB operand if they are coded. If not coded, the SDBMSG installation option from ICEAM2 or ICEAM4 controls this functionality.

LISTSDB has no effect for SYSOUT list datasets. A system-determined optimum block size is not used for spool or dummy datasets.

Description of the LISTSDB/LISTNOSDB operands

The LISTSDB/LISTNOSDB operands can be used to override the SDBMSG value for this LIST dataset. LISTSDB directs ICETOOL to select the system-determined optimum block size for the LIST dataset in the same way as for installation option SDBMSG=YES. LISTNOSDB directs ICETOOL to select the block size for the LIST dataset in the same way as for installation option SDBMSG=NO.

SPLICE operator

SPLICE is another new ICETOOL operator that can be utilized to perform various file 'join' and 'match' operations. SPLICE allows you to create output records in a number of different ways by splicing together fields from records that have the same key, but contain different information.

Description of the SPLICE operator

The SPLICE operator splices together fields from records with matching numeric or character field values. This makes it possible to join fields from different types of input record to create an output record with information from two or more records. An example of why you would potentially want to do this would be if you wanted to reformat the records from two or more datasets to temporary datasets, and concatenate those temporary datasets together as input to the SPLICE operator. The first duplicate is treated as a 'base record'. The last duplicate is treated as an 'overlay record'. Specified fields from the overlay record are overlaid onto the base record. Thus, the output record consists of fields from the base record intermixed with specified fields from the overlay record. You can use the KEEPNODUPS operand to keep the non-duplicate records as well as the spliced records. The non-duplicate records will be unchanged.

DFSORT is called to sort the indd dataset. ICETOOL uses its E35 exit to determine which records to splice and include in the outdd dataset. ICETOOL passes the EQUALS operand to DFSORT to ensure that duplicates are kept in their original input order.

The DFSORT control statements in xxxxCNTL are used if USING(xxxx) is specified. You can use DFSORT control statements and options in the xxxxCNTL dataset such as INCLUDE, OMIT, OPTION, and OUTFIL to eliminate records, reformat records, create reports, and so on.

When ICETOOL calls DFSORT, it passes control statements and options appropriate for the SPLICE operation being

performed. To avoid unintended results or abends, you should not use USING(xxxx) and xxxxCNTL to override the DFSORT control statements or options passed by ICETOOL.

OUTFIL ENHANCEMENTS

There are several new enhancements to DFSORT's powerful OUTFIL multiple output and reporting control statement.

The SAMPLE=n and SAMPLE=(n,m) options of OUTFIL allow you to sample records in a variety of ways. SAMPLE=n and SAMPLE=(n,m) are optional, and mutually exclusive, for OUTFIL. OUTFIL's STARTREC option can be used to start processing for an OUTFIL group at a specific OUTFIL input record. OUTFIL's ENDREC option can be used to end processing for an OUTFIL group at a specific OUTFIL input record. OUTFIL's new SAMPLE option can be used to select a sample of OUTFIL input records for an OUTFIL group using a specific interval and number of records in that interval. Separately or together, STARTREC, ENDREC, and SAMPLE can be used to select a range of records to which subsequent OUTFIL processing will apply.

OUTFIL's new REPEAT option can be used to repeat each OUTFIL output record a specified number of times. OUTFIL OUTREC's SEQNUM option can be used to assign a different sequence number to each repeated record. REPEAT specifies the number of times each OUTFIL output record is to be repeated for this OUTFIL group. Each OUTFIL output record is written *n* times. If SEQNUM is used in the OUTREC parameter for this OUTFIL group, the sequence number will be incremented for each repeated record.

The SPLITBY=n option of OUTFIL allows you to write groups of records in rotation among multiple output datasets. SPLIT and SPLITBY=n are optional and mutually exclusive for OUTFIL. OUTFIL's SPLIT option can be used to distribute one record at a time among the OUTFIL datasets. OUTFIL's new SPLITBY option can be used to distribute multiple records at

a time among the OUTFIL datasets. With SPLIT, the first output record is written to the first OUTFIL dataset in the group, the second output record is written to the second dataset, and so on. When each OUTFIL dataset has one record, the rotation starts again with the first OUTFIL dataset – a sort of round-robin approach.

SPLITBY can be used to rotate by a specified number of records rather than by one record, for example, records 1–10 to the first OUTFIL dataset, records 11–20 to the second OUTFIL dataset, and so on.

DFSORT ENHANCEMENTS TO REFORMATTING

Mathematical expressions and decimal constants

INREC, OUTREC, and OUTFIL OUTREC can be used to insert decimal constants and arithmetic expressions into records as numeric or edited character values. Optionally, you can choose to include the following in your reformatted INREC, OUTREC, and OUTFIL OUTREC records:

- Decimal constants converted to BI, FI, PD, ZD, or FS numeric values, or to CH values edited to contain signs, thousands separators, decimal points, leading zeros, or no leading zeros.
- The results of arithmetic expressions combining fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD, and SUB) and parentheses, converted to BI, FI, PD, ZD, or FS numeric values, or to CH values edited to contain signs, thousands separators, decimal points, leading zeros, or no leading zeros.

Constant DATE4

The DATE4 option for INREC, OUTREC, and OUTFIL OUTREC will allow a timestamp to be inserted for the DFSORT run in your records in the form yyyy-mm-dd-hh.mm.ss. Optionally, you can choose to include a CH timestamp representing the

date of the run as a separation field in your INREC, OUTREC, and OUTFIL OUTREC records.

INCLUDE and OMIT enhancements

There are a number of new filtering control statements for the INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT control statements.

The maximum length for an SS (substring) field used with INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT has been raised from 256 bytes to 32,752 bytes. The substring comparison test available with the INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT statements will allow you to include or omit records that have a specified character or hexadecimal constant anywhere within a field (which can be up to 32,752 bytes long). Thus, you can check for a constant anywhere in a large field, or even anywhere in an entire record, more easily.

A new DATE4 option of INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT allows you to compare a field with a timestamp for your DFSORT run in the form yyyy-mm-dd-hh.mm.ss, or with a portion of that timestamp truncated on the right.

The field-to-constant comparison test available with the INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT statements now allows you to use DATE4 as a constant representing the date and time of a run. DATE4 can be compared with a BI, CH, AC, AQ, or D2 field. The DATE4 keyword is not allowed as a symbol.

A PD0 field can now be compared with a hexadecimal constant or with another PD0 field for INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT.

The field-to-field and field-to-constant comparison tests available with the INCLUDE, OMIT, OUTFIL INCLUDE, and OUTFIL OMIT statements now allow you to use PD0 fields. A

2–8 byte PD0 field can be compared with another 2–8 byte PD0 field, or with a hexadecimal constant.

Enhancements to DFSORT's control statement continuation rules allow you to continue a line that breaks at column 71 anywhere in columns 2 to 16 of the next line. When a continuation line exists it will be treated as a logical extension of the preceding line. Either an operand or a remark field can begin on one line and continue on the next line.

When PARMDDN=DDname is specified at the installation of DFSORT, the //DFSPARM DD dataset will be used if a //DDname DD dataset is not present. When PARMDDN=DFSPARM is specified or defaulted at installation time, DFSORT will continue to use a //$ORTPARM DD dataset if a //DFSPARM DD dataset is not present. The PARMDDN=DDname installation option indicates the name of the DDname for the DFSPARM dataset. DFSORT can now use a //DFSPARM DD statement when PARMDDN=DDname specifies a DDname other than DFSPARM, but a //DDname DD statement is not present.

ENHANCEMENTS SUMMARY

ICETOOL enhancements

A new SPLICE operator helps you to perform various file 'join' and 'match' operations. SPLICE can be used to create output records in a variety of ways by splicing together fields from records that have the same key, but contain different information.

Non-duplicate records can be deleted or kept.

The USING(xxxx) option can now be used with ICETOOL's SELECT operator to process DFSORT control statements such as INCLUDE, OMIT, and OUTFIL for a SELECT operation.

There are new LISTSDB and LISTNOSDB options for ICETOOL's DEFAULTS.

The DISPLAY and OCCUR operators allow control of system-determined optimum blocksize for LIST datasets.

OUTFIL enhancements

New SAMPLE=n and SAMPLE=(n,m) options of OUTFIL allow you to sample records in a variety of different ways.

A new REPEAT=n option of OUTFIL allows you to write each output record more than once.

A new SPLITBY=n option of OUTFIL allows you to write groups of records in rotation among multiple output datasets.

OUTFIL OUTREC now allows you to insert decimal constants (+n and -n) in your records as BI, FI, PD, ZD, FS, or edited CH values.

OUTFIL OUTREC now allows you to combine fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD, SUB), and parentheses to form arithmetic expressions, and place the results in your records as BI, FI, PD, ZD, FS, or edited CH values.

There is a new DATE4 option of OUTFIL OUTREC, which will allow you to insert a timestamp for your DFSORT run in the form yyyy-mm-dd-hh.mm.ss into your records.

The maximum length for an SS field used with OUTFIL INCLUDE and OUTFIL OMIT has been increased and is now 32,752 bytes.

A new DATE4 option of OUTFIL INCLUDE and OUTFIL OMIT allows you to compare fields with a timestamp for your DFSORT run in the form yyyy-mm-dd-hh.mm.ss or with a portion of that timestamp truncated on the right.

A PD0 field can now be compared with a hexadecimal constant or with another PD0 field for OUTFIL INCLUDE and OMIT.

INREC and OUTREC enhancements

INREC and OUTREC now allow you to insert decimal constants

(+n and -n) in your records as BI, FI, PD, ZD, FS, or edited CH values.

INREC and OUTREC now allow you to combine fields (p,m,f), decimal constants (+n and -n), operators (MIN, MAX, MUL, DIV, MOD, ADD, SUB) and parentheses to form arithmetic expressions, and place the results in your records as BI, FI, PD, ZD, FS, or edited CH values.

A new DATE4 option of INREC and OUTREC allows you to insert a timestamp for your DFSORT run in the form yyyy-mm-dd-hh.mm.ss into your records.

INCLUDE and OMIT enhancements

The maximum length for an SS field used with INCLUDE and OMIT has been increased to a maximum of 32,752 bytes.

A new DATE4 option of INCLUDE and OMIT allows you to compare fields with a timestamp for your DFSORT run in the form yyyy-mm-dd-hh.mm.ss or with a portion of that timestamp truncated on the right.

A PD0 field can now be compared with a hexadecimal constant or with another PD0 field for INCLUDE and OMIT.

FORMAT=f can now be used with mixed p,m and p,m,f fields in the COND operand for INCLUDE and OMIT.

SORT, MERGE, and SUM enhancement

FORMAT=f can now be used with mixed p,m and p,m,f fields in the FIELDS operand for SORT, MERGE, and SUM.

The **f** from FORMAT=f will be used for p,m fields but not for p,m,f fields.

Other enhancements

Enhancements to DFSORT's control statement continuation rules allow you to continue a line that breaks at column 71 anywhere in columns 2 to 16 of the next line.

When PARMDDN=DDname is specified at installation time, DFSORT will now use a //DFSPARM DD dataset if a //DDname DD dataset is not present.

When PARMDDN=DFSPARM is specified or defaulted at installation time, DFSORT will continue to use a //$ORTPARM DD dataset if a //DFSPARM DD dataset is not present.

Messages

Be aware that a number of messages relating to DFSORT have also changed. I will not discuss or present them here. Further details on the messages that have been changed can be obtained either from PTF documentation or in the new DFSORT manuals.

John Bradley
Systems Programmer
Meerkat Computer Services

Serena Software has announced Version 6.2 of Serena TeamTrack for ChangeMan, its application development management software.

The new version has stronger bi-directional communication between TeamTrack, its request management solution, and Serena ChangeMan ZMF, its mainframe change management product. Whenever changes occur to a software change package on the mainframe, the mainframe will proactively communicate with TeamTrack and let it know of those changes.

The new version also extends the reach of the mainframe to non-mainframe users. Users can initiate, manage, and approve changes to mainframe applications directly from the Web.

For further information contact:
URL: www.serena.com/Products/teamtrack/Home.asp.

* * *

Mobius Management Systems and Network Appliance have announced the integration of Mobius ViewDirect Total Content Management (TCM) software with NetApp's NearStore disk-based nearline storage systems. The result is a hybrid solution that consolidates the archiving, recall, and management of content from any source, including Unix, Windows, Linux, and z/OS.

ViewDirect TCM integrates enterprise content in a consolidated repository or through access to multiple disparate repositories. The system includes a suite of content-centric applications enabling regulatory compliance and automated business processes. NearStore combines the Data ONTAP operating system with ATA disk drives for near-primary storage performance. The integrated solution helps to manage diverse content while addressing the need for rapid retrieval and high availability.

For further information contact:
URL: www.mobius.com.
URL: www.netapp.com.

* * *

MacKinney Systems has announced SimpList, its new productivity aid for TSO/ISPF.

The product allows users to improve their productivity while working in an ISPF environment. Instead of moving from one panel or vendor product to another, SimpList provides a centralized workbench. Multiple object types (eg datasets, DB2 tables, VSAM files) can be stored in object lists and selected for a variety of functions (eg browse, edit, print) by simple point-and-shoot selection. Based on the object type and function, SimpList automatically invokes the appropriate tool, vendor product, or built-in facility to handle the request.

For further information contact:
URL: www.mackinney.com/products/SIM/simplist.htm.

* * *

Mainstar Software has announced SYSchange, which is designed to control changes and ensure the integrity of critical system libraries.

SYSchange provides a way to control changes centrally and to automate processes. SYSchange is intended to protect system software with a secure and accountable process for implementing changes and making it simpler to restore an environment.

For further information contact:
URL: www.mainstar.com/products/dr/sys/index.asp.