221

MVS update

February 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.

Useful REXX edit macros

This article contains two useful edit macros – MKCOMMA and MKJOB.

The MKCOMMA macro can be used against any type of file to suffix each line with a comma. This is particularly useful for large SYSIN type streams that need to be modified. Obviously, with a little modification, the macro could suffix any character to the end of a line.

MKCOMMA EDIT MACRO

```
/* REXX */

/*       ************************************************************/
/*       *                                                          */
/*       * MACRO NAME:  MKCOMMA                                     */
/*       *                                                          */
/*       * PURPOSE:     ADD A COMMA AT THE END OF ALL LINES OF      */
/*       *              SYSIN STREAM.                               */
/*       *                                                          */
/*       * PROGRAMMER:  Elizabeth  Bradley.                         */
/*       *                                                          */
/*       * DATE:        26/10/2004.                                 */
/*       *                                                          */
/*       ************************************************************/

ADDRESS "ISPEXEC"            /* WORK UNDER ISPF.                    */
ISREDIT MACRO                /* ISPF EDIT MACRO.                    */
ADDRESS "ISREDIT"            /* EDIT COMMANDS FOLLOW.               */

 "(FIRST) = LINENUM .ZFIRST" /* SET UP POINTER TO FIRST LINE.       */
 "(LAST) = LINENUM .ZLAST"   /* POINT TO LAST LINE.                 */

 DO WHILE FIRST <= LAST      /* LOOP UNTIL ALL LINES PROCESSED.     */
   "(JCLVAR) = LINE" FIRST   /* pick up first line.                 */
   JCLVAR = STRIP(JCLVAR,T) /* strip trailing blanks.               */

 IF SUBSTR(JCLVAR,1,1) = "/" THEN /* if not jcl then process.      */
    JCLVAR = JCLVAR||","     /* concat a , on end of the line.      */
    "LINE" FIRST "= (JCLVAR)" /* write updated line to output.      */
    FIRST = FIRST + 1        /* PROCESS NEXT LINE.                  */
 END                         /* END OF LOOP.                        */
```

```
EXIT Ø                        /* LEAVE THE MACRO WITH RC= Ø.         */
```

The MKJOB macro copies a jobcard and a comment box into a newly-created PDS member. This is particularly useful to ensure that standards are adhered to. Before running, you do have to create members in the dataset you are editing named JOBCARD and COMMENT. Examples of the contents of these members are also shown below.

MKJOB EDIT MACRO

```
/* REXX */

/*        ************************************************************/
/*       *                                                            */
/*       * MACRO NAME:  MKJOB                                         */
/*       *                                                            */
/*       * PURPOSE:     TO SET UP INITIAL JCL IN EDIT LIBRAY.         */
/*       *              WILL COPY STANDARD JOBCARD AND COMMENT        */
/*       *              MEMBER WHEN NEW MEMBER IS CREATED.            */
/*       *                                                            */
/*       * PROGRAMMER:  ELIZABETH BRADLEY.                           */
/*       *                                                            */
/*       * DATE:        18/1Ø/2ØØ4.                                   */
/*       *                                                            */
/*        ************************************************************/

ADDRESS "ISPEXEC"             /* WORK UNDER ISPF.                    */
ISREDIT MACRO                 /* ISPF EDIT MACRO.                    */
ADDRESS "ISREDIT"             /* EDIT COMMANDS FOLLOW.               */

 "COPY JOBCARD AFTER Ø"       /* COPY JOBCARD.                       */
 "COPY COMMENT AFTER .ZLAST" /* COPY COMMENT AFTER LAST CARD.        */

EXIT Ø                        /* LEAVE THE MACRO WITH RC= Ø.         */
```

JOBCARD MEMBER EXAMPLE

```
//EXB7884  JOB  (GEN),'CSM,E.BRADLEY',CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1)
/*JOBPARM SYSAFF=*
```

COMMENT MEMBER EXAMPLE

```
//*
//*       ************************************************************
//*      *                                                            *
```

```
//*      *                                                              *
//*      *                                                              *
//*      *                                                              *
//*      *                                                              *
//*      *                                                              *
//*       ************************************************************
//*
```

Elizabeth Bradley
Systems Programmer
Meerkat Computer Services Ltd (UK)

Customized VSAM access from REXX

We have seen many articles talking about reading VSAM files from a REXX program. In most cases, a flexible REXX interface is written in Assembler and the user is asked to call the module to access one record at a time. However, this practice requires multiple calls to the module to access several records. If the requirement is to read the VSAM file for a given key, it is not necessary to call the module multiple times. It is more efficient to code a customized Assembler function to read the records and return them to the calling program. If the user wants to search VSAM for a search argument having 1,000 records, it is unnecessary to call the module 1,000 times to retrieve all the matching records. The Assembler code will need just a key and key length to search the KSDS. If it finds no matching records, it writes no records to a flat file. For added functionality, we could add a 'limit' parameter to the Assembler code. If an argument retrieves a tremendous number of matching records, it may not be acceptable for some applications/people. The limit parameter is used in the Assembler program to limit the number of matches to the 'limit' value. If the retrieved records exceed the limit value, the Assembler module sets the return value to 'MX'.

(Note: this Assembler has a restriction of 57 characters on the maximum key value; however, it can be changed in the Assembler program.)

Even though REXX offers plenty of functions and other features to work with flat files, when it comes to querying a bigger database from an on-line ISPF application, it doesn't offer any help. Searching a flat file that occupies a few thousand cylinders is not an option either. We just need to create a KSDS, populate/update it with the key and necessary fields, perhaps every day/week, and use this Assembler module to query from an on-line ISPF REXX application.

RXREAD

```
/*rexx*/
/*****************************************/
/* Interface to RXVSMQRY function        */
/* This REXX needs two arguments         */
/* 1st argument: searchkey               */
/* 2nd argument: searchlimit             */
/* (if searchlimit is not passed, default*/
/*  is 10                                */
/*---------------------------------------*/
/* RXVSMQRY parameter details            */
/* Needs two parameters (sep. by commas) */
/* 1st parm : key                        */
/* 2nd parm : length of key              */
/* 3nd parm : query limit                */
/*****************************************/
arg arg1
if arg1='' then
do
   say 'Please pass searchkey and search limit to search VSAM KSDS'
   return
end
parse var arg1 search_key search_limit .
if datatype(search_limit)¬='NUM' then search_limit=10
flat_outds=sysvar('sysuid')||'.RXREAD.OUTPUT'
search_key=strip(search_key)

vsam_dsn='VSAM-KSDS-FILENAME'
"alloc fi(vsamksds) da('"vsam_dsn"') shr reu"
if rc>0 then
do
  say ' VSAM KSDS allocation failed ' rc
```

```
  exit(8)
end
call create_flatfile
hkey=strip(search_key)
hkeyln=length(hkey)
thex=d2x(hkeyln,8)
str="hexkeyln='"thex"'x"
interpret str
thex=d2x(search_limit,8)
str="hexlimit='"thex"'x"
interpret str
bld_parm1=hkey||','
bld_parm2=hexkeyln||','
final_parm=bld_parm1||bld_parm2||hexlimit
ret1=rxvsmqry(''final_parm'')
if ret1='MX' then
do
say ' ------------------------------------------------------------- '
say ' Warning: number of resultant records are curtailed to LIMIT value'
say '          Try with a higher LIMIT value to search for all matches '
say ' ------------------------------------------------------------- '
end
if ret1='OK' | ret1='MX' then
do
   "execio * diskr flatout (stem recs. finis"
   say copies('-',70)
   say ' Search argument ==> 'search_key
   say ' Search resulted in 'recs.0' matches. Here is the result of',
         'search'
   say copies('-',70)
   do i=1 to recs.0
      udate=date('u',substr(recs.i,47,5),'J')
      say left(recs.i,44) udate substr(recs.i,52,6)
   end
   say copies('-',70)
   say 'output written to 'flat_outds
end
if ret1¬='OK' & ret1¬='MX' then
do
   say 'Error while reading VSAM KSDS'
   say left(ret1,20)
   say 'Diagnostic information'
   say '----------------------'
   say 'Feedback code:'
   say c2x(substr(ret1,21,1)) c2x(substr(ret1,22,1)) ,
       c2x(substr(ret1,23,1)) c2x(substr(ret1,24,1))
   say 'GPRegister R15:'
   say c2x(substr(ret1,25,1)) c2x(substr(ret1,26,1)) ,
       c2x(substr(ret1,27,1)) c2x(substr(ret1,28,1))
end
```

```
"free fi(flatout)"
"free fi(vsamksds)"
return
create_flatfile:
a=msg(off)
"delete '"flat_outds"'"
a=msg(on)
"allocate fi(flatout) space(1 1) tracks lrecl(57) new ",
"recfm(f b) dsorg(ps) da('"flat_outds"'"
if rc>Ø then
do
  say ' WORK file(flatout) allocation failed ' rc
  "free fi(vsamksds)"
  exit(8)
end
return
```

RXVSMQRY

```
*****************************************************************
*  IT IS A REXX FUNCTION. THE FOLLOWING PARAMETERS MUST BE PASSED
*
*  RXVSMQRY(KEY,KEYLENGTH,LIMIT)
*
*   KEY - CHARACTERS
*   KEYLENGTH - FULLWORD (4 BYTES)
*   LIMIT     - FULLWORD (4 BYTES)
*
*    I/P  - VSAMKSDS DD
*    O/P  - FLATOUT  DD
*****************************************************************
RXVSMQRY CSECT ,
RXVSMQRY AMODE 24
RXVSMQRY RMODE 24
         SAVE  (14,12)
         BALR  R12,Ø
         USING *,R12
         LA    R2,SAVEAREA
         ST    R2,8(,R13)
         ST    R13,SAVEAREA+4
         LR    R13,R2
*        ------------------------------------------------
*        SET BASE FOR EVAL BLOCK
*        ----------------------
         USING EVALBLOCK,R2
         L     R9,2Ø(R1)            R2 = ADDRESS REXX EVAL BLOCK
         L     R2,Ø(,R9)            LOAD R2 WITH EVALBLOCK ADDR
*        -----PARMIADR & PARMILEN WILL CONTAIN ADR AND LENGTH
         L     R9,16(R1)           R9  = ADDR OF PARM LIST
```

```
         L     R11,Ø(,R9)           R11 = ADDR OF INPUT DSNAME
         ST    R11,PARMIADR         STORE IN-PARM ADDRESS
         L     R9,4(,R9)            R9  = LENGTH OF INPUT DSNAME
         ST    R9,PARMILEN          STORE IN-PARM LENGTH
         LTR   R9,R9
         BZ    PARMERR1
*
         BAL   R6,PARMPROC
*
         OPEN  (FLATDCB,OUTPUT)
         OPEN  (VSAMACB)
         LTR   R15,R15
         BNZ   OPENERR1
         SR    R5,R5                INITIALIZE LIMITCNT
         MODCB RPL=VSAMRPL,KEYLEN=WKEYLEN
         POINT RPL=VSAMRPL
         MODCB RPL=VSAMRPL,OPTCD=(KEY,SEQ,FWD)
         MVC   INKEY(57),WKEY              STORE INPUT KEY IN INKEY
         LA    R1Ø,Ø
         L     R9,WKEYLEN              LOAD KEYLEN
         BCTR  R9,Ø                       REDUCE BY 1 FOR EX MVC
         ST    R9,WKEYLENM                STORE IT
READNEXT DS    ØH
         GET   RPL=VSAMRPL
         ST    R15,REG15
         LTR   R15,R15
         BNZ   GETERR1
         L     R9,WKEYLENM                LOAD KEYLENGTH-1
         EX    R9,CLCINST1                COMPARE WITH RETRIEVE JOBN.
         BNE   DONEALL                    IF NO MATCH, DONEALL
*        L     R5,LIMITCNT                LOAD LIMITCNT
         A     R5,=F'1'                   LIMITCNT=LIMITCNT+1
*        ST    R5,LIMITCNT
         C     R5,WLIMIT                  COMPARE WITH LIMIT PARM
         BH    MAXED
         PUT   FLATDCB,VSAMREC            WRITE OUTFILE
         B     READNEXT
DONEALL  DS    ØH
         MVC   EVALBLOCK_EVLEN,=F'2'              SET RETURN LEN 2
         MVC   EVALBLOCK_EVDATA(2),=CL2'OK'       RETURN OK
*        ------------------------------------------------
CLOSALL  DS    ØH
         CLOSE (VSAMACB,,FLATDCB)
EXIT     L     R13,SAVEAREA+4
         RETURN (14,12),RC=Ø
NOMATCH  DS    ØH
         MVC   EVALBLOCK_EVLEN,=F'2'              SET RETURN LEN 2
         MVC   EVALBLOCK_EVDATA(2),=CL2'NM'       SET NOMATCH FLAG
         B     CLOSALL
MAXED    DS    ØH
```

```
         MVC   EVALBLOCK_EVLEN,=F'2'              SET RETURN LEN 2
         MVC   EVALBLOCK_EVDATA(2),=CL2'MX'       SET MAXED  FLAG
         B     CLOSALL
OPENERR1 DS    ØH
         MVC   EVALBLOCK_EVLEN,=F'2Ø'
         MVC   EVALBLOCK_EVDATA(2Ø),=CL2Ø'OPEN ERROR'
         B     EXIT
GETERR1  DS    ØH
         SHOWCB RPL=VSAMRPL,FIELDS=FDBK,AREA=FDBKCDE,LENGTH=4
         L     R15,REG15
         C     R15,=F'8'
         BE    EODRC
YESGERR  DS    ØH
         MVC   EVALBLOCK_EVDATA(2Ø),=CL2Ø'GET ERROR'
         MVC   EVALBLOCK_EVDATA+2Ø(4),FDBKCDE
         MVC   EVALBLOCK_EVDATA+24(4),REG15
         MVC   EVALBLOCK_EVLEN,=F'3Ø'
         B     CLOSALL
EODRC    DS    ØH
         L     R15,FDBKCDE
         C     R15,=F'4'                     * CHECK FDBK CODE
         BE    DONEALL                       * RC=8 & FDBK=4 -> EOF
         B     YESGERR                       * OTHERWISE GET ERROR
PARMERR1 DS    ØH
         MVC   EVALBLOCK_EVLEN,=F'3Ø'
         MVC   EVALBLOCK_EVDATA(3Ø),=CL3Ø'NO PARMS PASSED'
         B     EXIT
PARMPROC DS    ØH
         L     R1Ø,PARMILEN                R1Ø HAS LENGTH
         LA    R9,Ø                        R9  - PARM LENGTH
         L     R7,PARMIADR                 R7  - START OF PARM
CHKAGAIN LA    R11,Ø(R7,R9)                R7+R9=R11 POINT TO LOC
         CLI   Ø(R11),C','                 CHECK FOR COMMA
         BE    COMFOUND                    IF YES, MOVE PARM
         LA    R9,1(R9)                    INCREMENT LENGTH BY 1
EOPCHK   BCT   R1Ø,CHKAGAIN                LOOP IF R1Ø NOT ZERO
COMFOUND DS    ØH                          MOVE LAST PARM IF ZERO
         L     R4,PARMNADR                 LOAD OFFSET TO R4
         L     R4,PARMPTRS(R4)           LOAD ACTUAL ADDRESS OF 1 PARM
         BCTR  R9,Ø                          REDUCE LENGTH  BY 1
         EX    R9,MVCINST1                   MOVE LENGTH TO MVC LEN
         LTR   R1Ø,R1Ø
         BZR   R6                            LAST PARM MOVED, GO BACK
         LA    R7,Ø(R7,R9)                   CHANGE START LOC
         LA    R7,2(R7)                      TO SKIP COMMA+1
         LA    R9,Ø                          RESET LENGTH
         L     R4,PARMNADR                   LOAD OFFSET AND INCREMENT
         LA    R4,4(R4)                      POINT TO NEXT PARM ADR
         ST    R4,PARMNADR
         B     EOPCHK
```

```
*
*    MVC AND CLC INSTRUCTIONS FOR EX
*
MVCINST1 MVC   Ø(Ø,R4),Ø(R7)                MOVE ONE PARM
CLCINST1 CLC   INKEY(Ø),KEYNAME
*
*  END OF PROGRAM
*
*  DATA AREA STARTS
*
SAVEAREA DS  18F
PARMNADR DC  F'Ø'
*
* THIS PROG CAN HANDLE ONLY THREE PARMS...TO INCREASE
* INCREMENT THE FOLLOWING COUNT AND IDENTIFY WHERE YOUR PARM IS
PARMPTRS DS  3F
         ORG PARMPTRS
         DC  A(WKEY)
         DC  A(WKEYLEN)
         DC  A(WLIMIT)
*
PARMIADR DS  F
PARMILEN DS  F
WKEYLENM DS  F
VSAMREC  DS  ØCL57
KEYNAME  DS  CL57
WKEY     DS  CL57
WKEYLEN  DS  F
WLIMIT   DS  F
REG15    DS  F
*LIMITCNT DS  F
OUTLINE  DS  CL8Ø
VSAMARG  DS  CL8
WMSGAREA DS  CL128
INKEY    DS  CL57
FDBKCDE  DS  F
WKFLD    DS  CL4
MSGFDBK  DS  CL8Ø
         ORG MSGFDBK
         DC  CL24'VSAM FEEDBACK CODE IS:'
MSGCDE   DS  CL3
         ORG
TRTBL    DC  C'Ø123456789ABCDEF'
FLATDCB  DCB   RECFM=FB,MACRF=PM,DDNAME=FLATOUT,DSORG=PS,              *
               LRECL=57,BLKSIZE=Ø
VSAMACB  ACB MACRF=(KEY,DIR,IN),DDNAME=VSAMKSDS
VSAMRPL  RPL ACB=VSAMACB,OPTCD=(KEY,DIR,KGE),                         *
               AM=VSAM,MSGAREA=WMSGAREA,MSGLEN=128,ARG=WKEY,          *
               KEYLEN=57,AREA=VSAMREC,AREALEN=57
         YREGS
```

```
IRXEVALB
END   RXVSMQRY
```


Project inventory

INTRODUCTION

What is an application project; what does it consist of? From a project leader's point of view, it is an assembly of files, programs, procedures, transactions, tables, JCL, etc, created to support certain business processes.

It stands to reason that all these elements should have something in common, and this is simply achieved by using a naming convention for them – the project name becomes an identification word, embedded into the name of any resource that belongs to the project.

If this convention is followed, it is possible to create a procedure that will extract any file from a system, using the project name as a search criterion.

PROBLEM DESCRIPTION

A project often increases in size over time. At some point, a project leader might ask themself which of the existing things on the system belong to the project.

I posed myself this question some time ago, and it was with some difficulty that I got everything I wanted. Thus I decided to make a procedure that could be executed at any time and that would create a report with an inventory of my project.

I was interested in getting a list of files, VSAM and non-VSAM,

DB2 tables and their indexes, programs, and all the information I would find useful about each of them.

For VSAM files it was file type, expiration date and date of last back-up, keylength, record length, number of records, and allocation parameters. For non-VSAM files and DB2 tables, I wanted much the same information.

I also collected some other useful information from the DB2 environment about indexes, tablespaces, etc, and gathered a list of members from various libraries that interested me, as well as overall statistics.

SOLUTION

In order to create a project inventory, a TSO REXX batch procedure named projinv was created.

If a file belonging to a project has a qualified name, the first qualifier should begin with, or be identical to, the project name. This is mandatory.

The same naming convention can be applied to libraries, while their first qualifiers are generally-accepted names for production libraries. The name of some other qualifier (not the first one) would then be the same as the project name or would have the project name embedded in it.

In this way, libraries containing programs, maps, JCL, CLISTs, REXX, or other procedures, which by naming convention should not be bound to any project, could also be searched.

DB2 table names, program names, map names (or any other procedure) as library members should have a project's name within them.

According to all of the above, projinv must have a project name, project files, and production library first qualifiers as arguments, and also the names of the libraries that are explicitly to be searched.

All the information about the VSAM files is obtained by using

the LISTCAT TSO command and reading its output.

LISTCAT with the level option runs the general search on the first qualifier and finds all the files and libraries; another LISTCAT can be issued on each of them to give detailed information.

Since the LISTCAT to non-VSAM files does not provide as much information as for VSAM, I submitted a LISTDS TSO command on non-VSAM files, then SORT and IEHLIST-LISTVTOC procedures in order to get all the information I wanted.

If a non-VSAM file is not on disk but on tape or some other inaccessible device type, I bypass all of this because it would require handling tapes, slowing down the procedure. The information that the file has been migrated is sufficient.

If a project has many large non-VSAM files, it might significantly slow down projinv execution. In these cases, the call to the sort procedure could be omitted, and record and byte number will be missed for these non-VSAM files and their members.

If a non-VSAM file is a multi-volume file, such information will be presented and the SORT and IEHLIST-LISTVTOC procedures omitted.

Information from the DB2 environment is provided by sequentially reading SYSIBM tables and getting the desired fields.

JCL to execute projinv looks like this:

```
//PROJINV  JOB 1,MSGLEVEL=(1,1),MSGCLASS=X,REGION=ØM,NOTIFY=&SYSUID
//STEPØØ1  EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K
//SYSEXEC  DD DSN=current_lib,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSOUT   DD DSN=sysout_file,DISP=SHR
//SYSPRINT DD DSN=sysprint_file,DISP=SHR
//FILEVSAM DD SYSOUT=*
//FILENONV DD SYSOUT=*
//FILEDB2  DD SYSOUT=*
//FILEMEMB DD SYSOUT=*
//SYSINF   DD DSN=current_lib(sysinf),DISP=SHR
//SYSINV   DD DSN=current_lib(sysinv),DISP=SHR
```

```
//SORTOUT  DD DUMMY
//SYSTSIN  DD   *
%PROJINV  project_name  pffq  plfq  lib1  lib2 ...
/*
```

The projinv procedure is a member of a library whose dsn=current_lib. Sysinf and sysinv are also in current_lib and they serve as sysin files to the sort procedure for fixed and variable record length non-VSAM files:

```
sysinf=" RECORD TYPE=F"
       " SORT FIELDS=(1,1,CH,A),WORK=3"
sysinv=" RECORD TYPE=V"
       " SORT FIELDS=(5,1,CH,A),WORK=3".
```

The sysprint_file and sysout_file are sequential files with a recommended record format of FBA and record length of 121, and enough space allocated.

Filevsam, filenonv, filedb2, and filememb are report files, and the total statistics counter is put in the systsprt file.

CODE

```
/*** rexx project inventory ***/
ARG arguments
project_name=STRIP(WORD(arguments,1))
pffq=STRIP(WORD(arguments,2))  /* project files first qualifier */
plfq=STRIP(WORD(arguments,3))  /* production libraries first qualifier*/
ol=Ø
DO i=4 TO WORDS(arguments)
  ol=ol+1
   other_lib.ol=plfq||'.'||STRIP(WORD(arguments,i))
                                    /* various production libraries */
END

vsam_counter=1
/* filevsam header */
vsam.vsam_counter=LEFT('File name',3Ø)||'Ftype    Created   Expire' ,
  ||'    Lastbackup     Keylen  Avglrecl  Maxlrecl  Recnumber',
  ||'  Allspacetype  Primary  Secondary  HI-A-RBA     HI-U-RBA   Extnum'
nonvsam_counter=1
/* filenonv header */
nonvsam.nonvsam_counter=LEFT('File name',3Ø)||'Created   Expire   ' ,
 ||' Lastbackup     Recfm  Lrecl  Dsorg  Memnumber  Recnumber  ' ,
 ||'Total bytes  Last ref  Extnum  Initial alloc  2nd alloc  Comment'
memb_counter=1
```

```
/* filememb header */
members.memb_counter=LEFT('Member name',3Ø)||'Recnumber  Bytes '
/* total statistics counters */
gdg_counter=Ø
library_counter=Ø
other_counter=Ø
other_libs=Ø

/* Search for a project files that produces 2 lines for each file. */
/* First one contains file_type ---- file_name,                    */
/* second line contains catalog name.                            */
t=OUTTRAP('files.')
"LISTCAT LVL ("pffq")"
t=OUTTRAP('OFF')
DO i=1 TO files.Ø BY 2
  PARSE UPPER VAR files.i  file_type  second  file_name
  file_name = STRIP(file_name)
  SELECT
    WHEN file_type='CLUSTER' | file_type='AIX' | file_type='DATA' | ,
         file_type='INDEX' | file_type='PATH' THEN
    DO  /* vsam files */
      vsam_counter=vsam_counter+1
      vsam.vsam_counter = LEFT(file_name,3Ø)||LEFT(STRIP(file_type),9)
      CALL get_vsam_file_info
    END
    WHEN file_type='NONVSAM' THEN
    DO
      nonvsam_counter = nonvsam_counter + 1
      nonvsam.nonvsam_counter = LEFT(file_name,3Ø)
      CALL get_nonvsam_file_info
    END
    WHEN file_type='GDG' THEN gdg_counter=gdg_counter+1
    OTHERWISE other_counter=other_counter+1
  END
END

/* Production libraries (programs, maps, REXX procedures etc) that do */
/* not belong to any specific project, but whose members do, and are  */
/* of interest to a project inventory. Could be as many as the number */
/* of typed arguments.                    */
t=OUTTRAP('libs.')
"LISTCAT LVL ("plfq")"
t=OUTTRAP('OFF')
DO i=1 TO libs.Ø BY 2
  PARSE UPPER VAR libs.i  file_type  second  file_name
  file_name = STRIP(file_name)
  IF file_type='NONVSAM' THEN
  DO
    found=Ø
    DO j=1 to ol
```

```
      IF file_name=other_lib.j THEN found=1
    END
    IF INDEX(file_name, project_name)>Ø | found=1 THEN
    DO
      nonvsam_counter = nonvsam_counter + 1
      nonvsam.nonvsam_counter = LEFT(file_name,3Ø)
      CALL get_nonvsam_file_info
    END
  END
END
"EXECIO * DISKW filevsam (STEM vsam. FINIS "
"EXECIO * DISKW filenonv (STEM nonvsam. FINIS"
"EXECIO * DISKW filememb (STEM members. FINIS"

/** access to DB2 environment **/
'SUBCOM DSNREXX'
ADDRESS DSNREXX
IF rc THEN s_rc = RXSUBCOM('ADD','DSNREXX','DSNREXX')
'CONNECT' 'DSN'

db2_counter=1
/* filedb2 - tables header */
db2.db2_counter='Table name        Creator   Type  Tsname   ' ,
  'Colcount  Recnumber Reclength  Created              Altered'
sqlstmt1=,
  "SELECT NAME, CREATOR, TYPE, TSNAME, COLCOUNT, CARD, RECLENGTH, " ,
  "CREATEDTS, ALTEREDTS " ,
  "FROM SYSIBM.SYSTABLES WHERE NAME" ,
  " LIKE '%"||project_name||"%' ORDER BY NAME"
command='dcl cursor c1'
"EXECSQL DECLARE c1 CURSOR FOR s1"
IF sqlcode¬=Ø THEN CALL error
"EXECSQL PREPARE s1 FROM :sqlstmt1"
command='prepare s1'
IF sqlcode¬=Ø THEN CALL error
"EXECSQL OPEN c1"
command='open cursor c1'
IF sqlcode¬=Ø THEN CALL error
db2_tables=Ø
"EXECSQL FETCH c1 INTO :name, :creator, :type, :tsname, :colcount, ",
                      ":card, :reclength, :createdts, :alteredts"
DO WHILE (SQLCODE = Ø )
  db2_counter=db2_counter+1
  db2_tables=db2_tables+1
  db2.db2_counter=LEFT(name,18)||LEFT(creator,1Ø)||LEFT(type,6)|| ,
     LEFT(tsname,1Ø)||LEFT(colcount,1Ø)||LEFT(card,1Ø)|| ,
     LEFT(reclength,11)||LEFT(SUBSTR(createdts,1,19),21)|| ,
     LEFT(SUBSTR(alteredts,1,19),21)
  "EXECSQL FETCH c1 INTO :name, :creator, :type, :tsname, :colcount, ",
                        ":card, :reclength, :createdts, :alteredts"
```

```
END
"EXECSQL CLOSE c1"
command='close cursor c1'
IF sqlcode¬=Ø THEN CALL error

db2_counter=db2_counter+1
db2.db2_counter=' '
db2_counter=db2_counter+1
/* filedb2 - tablespaces header */
db2.db2_counter='Ts name   Status  Numtables  Space(Kb) '
sqlstmt2=,
  "SELECT NAME, STATUS, NTABLES, SPACE " ,
  "FROM SYSIBM.SYSTABLESPACE WHERE NAME" ,
  " LIKE '%"||project_name||"%' ORDER BY NAME"
command='dcl cursor c2'
"EXECSQL DECLARE c2 CURSOR FOR s2"
IF sqlcode¬=Ø THEN CALL error
"EXECSQL PREPARE s2 FROM :sqlstmt2"
command='prepare s2'
IF sqlcode¬=Ø THEN CALL error
"EXECSQL OPEN c2"
command='open cursor c2'
IF sqlcode¬=Ø THEN CALL error
db2_tspaces=Ø
"EXECSQL FETCH c2 INTO :name, :status, :ntables, :space "
DO WHILE (SQLCODE = Ø )
  db2_counter=db2_counter+1
  db2_tspaces=db2_tspaces+1
   db2.db2_counter=LEFT(name,1Ø)||LEFT(status,8)||LEFT(ntables,11)||space
  "EXECSQL FETCH c2 INTO :name, :status, :ntables, :space "
END
"EXECSQL CLOSE c2"
command='close cursor c2'
IF sqlcode¬=Ø THEN CALL error

db2_counter=db2_counter+1
db2.db2_counter=' '
db2_counter=db2_counter+1
/* filedb2 - indexes header */
db2.db2_counter=LEFT('Index name',18)||LEFT('Table name',18)|| ,
    'Uniquerule  Recnumber  Space(Kb)  Colname'
sqlstmt3=,
  "SELECT name, tbname, uniquerule, fullkeycard, space, colname " ,
  "FROM SYSIBM.SYSINDEXES, SYSIBM.SYSKEYS " ,
  "WHERE TBNAME LIKE '%"||project_name||"%' AND NAME=IXNAME " ,
  "ORDER BY TBNAME, NAME"
command='dcl cursor c3'
"EXECSQL DECLARE c3 CURSOR FOR s3"
IF sqlcode¬=Ø THEN CALL error
"EXECSQL PREPARE s3 FROM :sqlstmt3"
```

```
command='prepare s3'
IF sqlcode¬=Ø THEN CALL error
"EXECSQL OPEN c3"
command='open cursor c3'
IF sqlcode¬=Ø THEN CALL error
prev_name=' '
"EXECSQL FETCH c3 INTO :name, :tbname, :uniquerule, :fullkeycard, " ,
   ":space, :colname"
db2_indexes=Ø
DO WHILE (SQLCODE = Ø )
  IF name¬=prev_name THEN
  DO
    db2_counter=db2_counter+1
    db2_indexes=db2_indexes+1
    db2.db2_counter=LEFT(name,18)||LEFT(tbname,18)|| ,
     LEFT(uniquerule,12)||LEFT(fullkeycard,11)||LEFT(space,11)||colname
  END
  ELSE db2.db2_counter=prev_line||'  '||colname
  prev_line=db2.db2_counter
  prev_name=name
  "EXECSQL FETCH c3 INTO :name, :tbname, :uniquerule, :fullkeycard, " ,
    ":space, :colname"
END
"EXECSQL CLOSE c3"
command='close cursor c3'
IF sqlcode¬=Ø THEN CALL error

db2_counter=db2_counter+1
db2.db2_counter=' '
 db2_counter=db2_counter+1
/* filedb2 - foreignkeys header */
db2.db2_counter=LEFT('Rel name',1Ø)||LEFT('Table name',18)|| ,
   LEFT('Parent table',18)||'Column'
sqlstmt4=,
  "SELECT a.relname, a.tbname, reftbname, colname " ,
  "FROM SYSIBM.SYSRELS a, SYSIBM.SYSFOREIGNKEYS b" ,
  "WHERE a.tbname LIKE '%"||project_name||"%' AND " ,
        "a.relname=b.relname " ,
  "ORDER BY a.tbname, a.relname"
command='dcl cursor c4'
"EXECSQL DECLARE c4 CURSOR FOR s4"
IF sqlcode¬=Ø THEN CALL error
"EXECSQL PREPARE s4 FROM :sqlstmt4"
command='prepare s4'
IF sqlcode¬=Ø THEN CALL error
"EXECSQL OPEN c4"
command='open cursor c4'
IF sqlcode¬=Ø THEN CALL error
prev_name=' '
db2_rels=Ø
```

```
"EXECSQL FETCH c4 INTO :relname, :tbname, :reftbname, :colname"
DO WHILE (SQLCODE = Ø )
  IF relname¬=prev_name THEN
  DO
    db2_counter=db2_counter+1
    db2.db2_counter=LEFT(relname,1Ø)||LEFT(tbname,18)|| ,
         LEFT(reftbname,18)||colname
    db2_rels=db2_rels+1
  END
  ELSE db2.db2_counter=prev_line||'  '||colname
  prev_line=db2.db2_counter
  prev_name=relname
  "EXECSQL FETCH c4 INTO :relname, :tbname, :reftbname, :colname"
END
"EXECSQL CLOSE c4"
command='close cursor c4'
IF sqlcode¬=Ø THEN CALL error

db2_counter=db2_counter+1
db2.db2_counter=' '
db2_counter=db2_counter+1
/* filedb2 - planes header */
db2.db2_counter='Plan name  Creator   Binddate  Valid  Operative  ' ,
    ||'Bindtime  Boundby   Dbrm      Number'
sqlstmt5=,
  "SELECT a.name, creator, binddate, valid, operative, bindtime, " ,
         "boundby, b.name" ,
  "FROM SYSIBM.SYSPLAN a, SYSIBM.SYSDBRM b" ,
  "WHERE a.name LIKE '%"||project_name||"%' AND a.name=plname " ,
  "ORDER BY a.name, b.name"
command='dcl cursor c5'
"EXECSQL DECLARE c5 CURSOR FOR s5"
IF sqlcode¬=Ø THEN CALL error
"EXECSQL PREPARE s5 FROM :sqlstmt5"
command='prepare s5'
IF sqlcode¬=Ø THEN CALL error
"EXECSQL OPEN c5"
command='open cursor c5'
IF sqlcode¬=Ø THEN CALL error
prev_name=' '
dbrm_counter=Ø
db2_plan=Ø
"EXECSQL FETCH c5 INTO :plname, :creator, :binddate, :valid, " ,
                      ":operative, :bindtime, :boundby, :dbrm"
DO WHILE (SQLCODE = Ø )
  IF plname¬=prev_name THEN
  DO
    IF dbrm_counter>Ø THEN db2.db2_counter=prev_line||dbrm_counter
    dbrm_counter=1
    db2_counter=db2_counter+1
```

```
    db2_plan=db2_plan+1
    db2.db2_counter=LEFT(plname,11)||LEFT(creator,1Ø)|| ,
         LEFT(binddate,1Ø)||LEFT(valid,7)||LEFT(operative,11)|| ,
         LEFT(bindtime,1Ø)||LEFT(boundby,1Ø)||LEFT(dbrm,1Ø)
    prev_line=db2.db2_counter
  END
  ELSE dbrm_counter=dbrm_counter+1
  prev_name=plname
  "EXECSQL FETCH c5 INTO :plname, :creator, :binddate, :valid, " ,
                        ":operative, :bindtime, :boundby, :dbrm"
END
IF dbrm_counter>Ø THEN db2.db2_counter=prev_line||dbrm_counter
"EXECSQL CLOSE c5"
command='close cursor c5'
IF sqlcode¬=Ø THEN CALL error
s_rc = RXSUBCOM('DELETE','DSNREXX','DSNREXX')

/* return to tso */
ADDRESS TSO
"EXECIO * DISKW filedb2 (STEM db2. FINIS "

/** gathered overall statistics **/
SAY 'Project '||project_name||' has '||vsam_counter-1|| ,
    ' vsam (cluster, data, index, path, aix) files'
SAY 'Project '||project_name||' has '||gdg_counter||' gdg files'
SAY 'Project '||project_name||' has '||other_counter|| ,
    ' other (usercatalog, space, alias) files'
SAY 'Project '||project_name||' has '|| ,
    nonvsam_counter-1-other_libs||' nonvsam files, amongst which ',
    ||library_counter-other_libs||' libraries'
SAY 'Project '||project_name||' has '||db2_tables||' db2 tables'
SAY 'Project '||project_name||' has '||db2_tspaces||' db2 tablespaces'
SAY 'Project '||project_name||' has '||db2_indexes||' indexes on ',
    'db2 tables'
SAY 'Project '||project_name||' has '||db2_rels||' referential ' ,
    'constraints on db2 tables'
SAY 'Project '||project_name||' has '||db2_plan||' planes '

EXIT

get_vsam_file_info: /* reading the listcat output for vsam file */
t=OUTTRAP('listcatline.')
   "LISTCAT ENT('"file_name"') ALL"
   t=OUTTRAP('OFF')
   PARSE UPPER VAR listcatline.4  dsowner  crdate
   PARSE UPPER VAR listcatline.5  release  exdate
   crdate=SUBSTR(STRIP(crdate),17)
   exdate=SUBSTR(STRIP(exdate),17)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(crdate,1Ø) ,
                     ||LEFT(exdate,1Ø)
```

```
IF file_type='PATH' | file_type='AIX' THEN RETURN
IF file_type='DATA' | file_type='INDEX' ,
THEN vsam.vsam_counter = vsam.vsam_counter||LEFT(' ',15)
DO k=6 TO listcatline.Ø
  IF file_type='CLUSTER' & WORD(listcatline.k,3)='LBACKUP' THEN
  DO
    lbackup=SUBSTR(STRIP(WORD(listcatline.k,4)),4)
    vsam.vsam_counter = vsam.vsam_counter||LEFT(lbackup,15)
   RETURN
  END
  IF SUBSTR(WORD(listcatline.k,1),1,6)='KEYLEN' THEN
  DO
    keylen=STRIP(WORD(listcatline.k,1))
    DO l=1 TO LENGTH(keylen)
      IF VERIFY(SUBSTR(keylen,l,1),'Ø123456789')=Ø THEN LEAVE
    END
    keylen=SUBSTR(keylen,l)
    IF file_type='DATA' ,
    THEN vsam.vsam_counter = vsam.vsam_counter||LEFT(keylen,8)
    ELSE vsam.vsam_counter = vsam.vsam_counter||LEFT(' ',8)
  END
  IF SUBSTR(WORD(listcatline.k,2),1,8)='AVGLRECL' THEN
  DO
    avglrecl=STRIP(WORD(listcatline.k,2))
    DO l=1 TO LENGTH(avglrecl)
      IF VERIFY(SUBSTR(avglrecl,l,1),'Ø123456789')=Ø THEN LEAVE
    END
    avglrecl=SUBSTR(avglrecl,l)
    vsam.vsam_counter = vsam.vsam_counter||LEFT(avglrecl,1Ø)
  END
  IF SUBSTR(WORD(listcatline.k,2),1,8)='MAXLRECL' THEN
  DO
    maxlrecl=STRIP(WORD(listcatline.k,2))
    DO l=1 TO LENGTH(maxlrecl)
      IF VERIFY(SUBSTR(maxlrecl,l,1),'Ø123456789')=Ø THEN LEAVE
    END
    maxlrecl=SUBSTR(maxlrecl,l)
    vsam.vsam_counter = vsam.vsam_counter||LEFT(maxlrecl,1Ø)
  END
  IF SUBSTR(WORD(listcatline.k,1),1,9)='REC-TOTAL' THEN
  DO
    recnum=STRIP(WORD(listcatline.k,1))
    DO l=1 TO LENGTH(recnum)
      IF VERIFY(SUBSTR(recnum,l,1),'Ø123456789')=Ø THEN LEAVE
    END
    recnum=SUBSTR(recnum,l)
    vsam.vsam_counter = vsam.vsam_counter||LEFT(recnum,11)
  END
  IF SUBSTR(WORD(listcatline.k,1),1,1Ø)='SPACE-TYPE' THEN
  DO
```

```
   allspa=SUBSTR(STRIP(WORD(listcatline.k,1)),11)
   DO l=1 TO LENGTH(allspa)
     IF VERIFY(SUBSTR(allspa,l,1),'-')=1 THEN LEAVE
   END
   allspa=SUBSTR(allspa,l)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(allspa,14)
END
IF SUBSTR(WORD(listcatline.k,1),1,9)='SPACE-PRI' THEN
DO
   spapri=STRIP(WORD(listcatline.k,1))
   DO l=1 TO LENGTH(spapri)
     IF VERIFY(SUBSTR(spapri,l,1),'Ø123456789')=Ø THEN LEAVE
   END
   spapri=SUBSTR(spapri,l)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(spapri,9)
END
IF SUBSTR(WORD(listcatline.k,1),1,9)='SPACE-SEC' THEN
DO
   spasec=STRIP(WORD(listcatline.k,1))
   DO l=1 TO LENGTH(spasec)
     IF VERIFY(SUBSTR(spasec,l,1),'Ø123456789')=Ø THEN LEAVE
   END
   spasec=SUBSTR(spasec,l)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(spasec,11)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(hiarba,13)
   vsam.vsam_counter = vsam.vsam_counter||LEFT(hiurba,11)
END
IF SUBSTR(WORD(listcatline.k,2),1,8)='HI-A-RBA' THEN
DO
   hiarba=STRIP(WORD(listcatline.k,2))
   DO l=1 TO LENGTH(hiarba)
     IF VERIFY(SUBSTR(hiarba,l,1),'Ø123456789')=Ø THEN LEAVE
   END
   hiarba=SUBSTR(hiarba,l)
END
IF SUBSTR(WORD(listcatline.k,2),1,8)='HI-U-RBA' THEN
DO
   hiurba=STRIP(WORD(listcatline.k,2))
   DO l=1 TO LENGTH(hiurba)
     IF VERIFY(SUBSTR(hiurba,l,1),'Ø123456789')=Ø THEN LEAVE
   END
   hiurba=SUBSTR(hiurba,l)
END
IF SUBSTR(WORD(listcatline.k,4),1,13)='EXTENT-NUMBER' THEN
DO
   extn=STRIP(WORD(listcatline.k,4))
   DO l=1 TO LENGTH(extn)
     IF VERIFY(SUBSTR(extn,l,1),'Ø123456789')=Ø THEN LEAVE
   END
   extn=SUBSTR(extn,l)
```

```
        vsam.vsam_counter = vsam.vsam_counter||LEFT(extn,6)
      END
    END

    RETURN

get_nonvsam_file_info:
    t=OUTTRAP('listcatline.')
    "LISTCAT ENT('"file_name"') ALL"
    t=OUTTRAP('OFF')
    PARSE UPPER VAR listcatline.4  dsowner  crdate
    PARSE UPPER VAR listcatline.5  release  exdate
    crdate=SUBSTR(STRIP(crdate),17)
    exdate=SUBSTR(STRIP(exdate),17)
    nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||LEFT(crdate,1Ø) ,
       ||LEFT(exdate,1Ø)
    lbackup=" "
    volser=" "
    multi_volume=Ø
    devtype=" "
    DO k=6 TO listcatline.Ø
      IF WORD(listcatline.k,3)='LBACKUP' ,
      THEN lbackup=SUBSTR(STRIP(WORD(listcatline.k,4)),4)
      IF SUBSTR(WORD(listcatline.k,1),1,6)='VOLSER' THEN
      DO
        IF volser¬=" " THEN multi_volume=1
        volser=STRIP(SUBSTR(WORD(listcatline.k,1),7))
        DO l=1 TO LENGTH(volser)
          IF SUBSTR(volser,l,1)¬="-" THEN LEAVE
        END
        volser=SUBSTR(volser,l)
      END
      IF SUBSTR(WORD(listcatline.k,2),1,7)='DEVTYPE' & devtype=" " THEN
      DO
        devtype=STRIP(WORD(listcatline.k,2))
        DO l=1 TO LENGTH(devtype)
          IF VERIFY(SUBSTR(devtype,l,1),'Ø123456789')=Ø THEN LEAVE
        END
        devtype=SUBSTR(devtype,l)
        devtype=SUBSTR(devtype,1,LENGTH(devtype)-1)
        accessible=3Ø1Ø2ØØF
                  /* this is hexadecimal number of disk on IBM system */
                  /* any other is inaccessible to further commands    */
      END
      IF SUBSTR(WORD(listcatline.k,1),1,7)='DSNTYPE' THEN
      DO
        dsntype=STRIP(SUBSTR(WORD(listcatline.k,1),8))
        DO l=1 TO LENGTH(dsntype)
          IF SUBSTR(dsntype,l,1)¬="-" THEN LEAVE
        END
```

```
        dsntype=SUBSTR(dsntype,1)
        IF dsntype='LIBRARY' THEN
        DO
          library_counter=library_counter+1
          DO j=1 to ol
            IF file_name=other_lib.j THEN other_libs=other_libs+1
          END
        END
      END
    END
    nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||LEFT(lbackup,15)
    IF devtype¬=accessible THEN
    DO
      nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||'file is ' ,
          ||'inaccessible. volser = '||volser
      RETURN
    END
    /* listds gives the record format, record length, block size and data
set organization */
    t=OUTTRAP('listdsline.')
    "LISTDS '"file_name"' MEMBERS"
    t=OUTTRAP('OFF')
    PARSE UPPER VAR listdsline.3  recfm  lrecl  blksize  dsorg
    nonvsam.nonvsam_counter = nonvsam.nonvsam_counter|| ,
       LEFT(STRIP(recfm),7)||LEFT(STRIP(lrecl),7)||LEFT(STRIP(dsorg),7)
    IF multi_volume=1 THEN
    DO
      nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||'file is ' ,
       ||'multi volume'
      RETURN
    END
    IF dsorg='PO' THEN
    DO
/* lists members of library (dsorg='PO'), all for project libraries, */
/* only ones with project name imbedded for general libraries        */
      mem_counter=0
      lib_bytes=0
      lib_recnum=0
      DO k=7 TO listdsline.0
        found=0
        DO j=1 to ol
          IF file_name=other_lib.j & INDEX(listdsline.k,project_name)>0
          THEN found=1
        END
        IF INDEX(file_name, project_name)>0 | found=1 THEN
        DO
          mem_counter=mem_counter+1
          sort_file=file_name||'('||STRIP(listdsline.k)||')'
          CALL calculate
          memb_counter=memb_counter+1
```

```
          members.memb_counter=LEFT(sort_file,30)|| ,
            LEFT(number_of_records,11)||number_of_bytes
          lib_bytes=lib_bytes+number_of_bytes
          lib_recnum=lib_recnum+number_of_records
        END
      END
      nonvsam.nonvsam_counter = nonvsam.nonvsam_counter|| ,
        LEFT(mem_counter,11)||LEFT(lib_recnum,11)||LEFT(lib_bytes,13)
   END
   ELSE DO
     sort_file=file_name
     CALL calculate
     nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||LEFT(' ',11) ,
       ||LEFT(number_of_records,11)||LEFT(number_of_bytes,13)
   END
/* Invoking iehlist which reads the sysprint file produced by listvtoc*/
/* command, applied on the volume on which specific file resides      */
   ih=1
   ihstem.ih=' LISTVTOC FORMAT,VOL=SYSDA='||volser
                                              /* volser from listcat */
   "ALLOC F(SYSIN) DA('current_lib(SYSINI)') SHR REU"
   IF rc>0 THEN
   DO
     SAY "error alloc sysin - iehlist "rc
     EXIT rc
   END
   "ALLOC F(DD1) DA('"file_name"') UNIT(SYSDA) VOL("volser") SHR"
   IF rc>0 THEN
   DO
     SAY "error alloc dd1 "rc
     EXIT rc
   END
   "EXECIO * DISKW sysin (STEM ihstem. FINIS "
   "TSOEXEC CALL 'SYS1.LINKLIB(IEHLIST)'"
   "FREE FI(DD1)"
   IF rc>0 THEN
   DO
     SAY "error free dd1 "rc
     EXIT rc
   END
   "FREE FI(SYSIN)"
   IF rc>0 THEN
   DO
     SAY "error free sysin "rc
     EXIT rc
   END
   found=0
   comment=' '
   DO WHILE rc¬=2
     "EXECIO 1 DISKR sysprint"
```

```
      IF rc=Ø THEN
      DO
        IF found>Ø THEN found=found+1
        PARSE PULL rec
        IF WORD(rec,1)=file_name THEN
        DO
          found=1
          dateref=STRIP(WORD(rec,6))
          extnum=LEFT(STRIP(WORD(rec,7)),8)
        END
        IF found=3 THEN alloc=SUBSTR(rec,27,24)
        IF found>3 & SUBSTR(rec,6Ø,9)='THERE ARE'
        THEN comment=SUBSTR(rec,69)
        IF found>3 & SUBSTR(rec,1,5)='----' THEN LEAVE
      END
    END
    "EXECIO Ø DISKR sysprint (FINIS"

    nonvsam.nonvsam_counter = nonvsam.nonvsam_counter||   ,
      LEFT(dateref,1Ø)||LEFT(extnum,8)||alloc||'   '||STRIP(comment)

    RETURN

calculate: /* calculates number of records and bytes in sort_file */
    number_of_bytes = Ø
    number_of_records = Ø
    number_of_bytes_sorted = Ø
    "ALLOC FI(sortin) DA('"sort_file"') SHR REU"
    IF rc>Ø THEN
    DO
      SAY "error alloc fi(sortin) "sort_file
      EXIT rc
    END

    IF SUBSTR(recfm,1,1)='F'
    THEN "ALLOC FI(sysin) DA('current_lib(SYSINF)') SHR REU"
    ELSE "ALLOC FI(sysin) DA('current_lib(SYSINV)') SHR REU"
    IF rc>Ø THEN
    DO
      SAY "error alloc fi(sysin) RC="rc
      EXIT rc
    END

    "CALL 'sys1.sortlpa(sort)'"
    IF rc>Ø THEN
    DO
      SAY "error call sort "sort_file "RC="rc
      EXIT rc
    END
```

```
   "FREE FI(sortin)"
   IF rc>Ø THEN
   DO
     SAY "error free fi(sortin) "sort_file
     EXIT rc
   END

   "FREE FI(sysin)"
   IF rc>Ø THEN
   DO
     SAY "error free fi(sysin) "
     EXIT rc
   END

   DO WHILE rc¬=2
     "EXECIO 1 DISKR sysout"
     IF rc=Ø THEN
     DO
       PARSE PULL rec
       IF (WORD(rec,1)="ICEØ54I") ,
       THEN number_of_records=WORD(rec,8)
       IF (WORD(rec,1)="ICE134I") THEN
       DO
         number_of_bytes_sorted=WORD(rec,7)
         /* variable file sort adds 4 bytes to each record */
         IF recfm="VB"
         THEN number_of_bytes = ,
             (number_of_bytes_sorted - (number_of_records * 4))
         ELSE number_of_bytes = number_of_bytes_sorted
       END
     END
   END
   "EXECIO Ø DISKR sysout (FINIS"

   RETURN

error:
   SAY 'SQL error '||command|| sqlcode
   rc=1Ø
   EXIT rc
```

EXAMPLE

I ran projinv with following arguments:

```
//SYSTSIN  DD   *
%PROJINV  MYPR  MYPRP  PRODL  BATCH.PLI  CICS.PLI
/*
```

MYPR is my project, MYPRP is the project files' first qualifier, PRODL is the production libraries' first qualifier, (PRODL.)BATCH.PLI is the batch programs library, and (PRODL.)CICS.PLI is the CICS programs library.

The systsprt file looks like this:

```
READY
%PROJINV  MYPR  MYPRP  PRODL  BATCH.PLI  CICS.PLI
Project MYPR has 82 vsam (cluster, data, index, path, aix) files
Project MYPR has 0 gdg files
Project MYPR has 0 other (usercatalog, space, alias) files
Project MYPR has 115 nonvsam files, amongst which 5 libraries
Project MYPR has 46 db2 tables
Project MYPR has 5 db2 tablespaces
Project MYPR has 64 indexes on  db2 tables
Project MYPR has 15 referential  constraints on db2 tables
Project MYPR has 5 planes
READY
END
```

(Part of) filevsam looks like this:

```
File name                 Ftype    Created   Expire    Lastbackup
Keylen  Avglrecl  Maxlrecl
MYPRP.ISTPROM.CI29        CLUSTER  2004.278  0000.000  2004.278.1827
MYPRP.ISTPROM.CI29.DATA   DATA     2004.278  0000.000                  16
380       380
MYPRP.ISTPROM.CI29.INDEX  INDEX    2004.278  0000.000
0         505

Recnumber  Allspacetype  Primary  Secondary  HI-A-RBA    HI-U-RBA
Extnum

8215665    CYLINDER      3296     300        5265162240  3813089280  1
1     0     0     0
4913       TRACK         70       1          2531328     2515456     34
0     0     0     0
```

File MYPRP.ISTPROM.CI29, shown in the example, is a large multi-volume file, which could be concluded from the number of different extents (per volume). *Lastbackup* information refers to the cluster file, and all the others to data and index.

(Part of) filenonv looks like this:

```
File name              Created   Expire    Lastbackup     Recfm  Lrecl
Dsorg  Memnumber
MYPRP.DATKINCL.TEMP    2002.349  0000.000  2003.017.1720  FB     214
```

```
PS
PRODL.MYPR.FTP          2003.328  0000.000  2004.133.2133  FB      80
PO     5
MYPRP.PENSAMO.TEMP      2002.028  0000.000  XXXX.XXX.XXXX  file is
inaccessible. volser =

Recnumber  Total bytes  Last ref  Extnum  Initial alloc  2nd alloc
Comment
373        79822        2004.301  1         RECS             100    0
EMPTY TRACK(S).
263        21040        2004.301  1         CYLS               1    10
EMPTY TRACK(s).
MIGRAT
```

There are two sequential datasets in the example above, one of them migrated, and a library with five members.

(Part of) filedb2 looks like this:

```
Table name         Creator   Type  Tsname    Colcount  Recnumber
Reclength  Created              Altered
TBMYPR08           MYPRP     T*    TSMYPR01  29        605729    128
2002-10-21-10.51.18  2002-10-21-10.51.37

(* A-alias, T-table, V-view).

Ts name   Status  Numtables  Space(Kb)
TSMYPR01  A*      25         156240

(* A-available, P,S-check pending state, T,C-definition is incomplete).

Index name         Table name         Uniquerule  Recnumber  Space(Kb)
Colname
XMYPR011           TBMYPR01           P*          42593      1008
ID_NUMBER

(* whether the index is unique: P-yes primary, D-no(duplicates allowed),
  U-yes, C-yes (UNIQUE constraint), N-yes (UNIQUE WHERE NOT NULL)).

Rel name  Table name        Parent table      Column
R0206     TBMYPR02          TBMYPR06          ACCOUNT_NUMBER

Plan name  Creator   Binddate  Valid  Operative  Bindtime  Boundby
Dbrm      Number
MYPR0001   USERID    041025    Y      Y*         13314532  USERID
MYPRAZ35  90

(* Y-yes, N-no, bind or rebind required!).
```

(Part of) filememb looks like this:

```
Member name                    Recnumber  Bytes
PRODL.BATCH.PLI(MYPRAPOS)      71         5680
PRODL.CICS.PLI(MYPR0001)       235        18800
PRODL.MYPR.CNTL(BIND0001)      37         2960
PRODL.MYPR.FTP(FTPCHECK)       151        12080
```

Vladimir Antonijevic
IT Analyst
Postal Savings Bank (Serbia and Montenegro)

Monitoring dataspaces

Some time ago I ran into a situation where an ISV's product address space wouldn't come up because it couldn't create a SCOPE=COMMON dataspace. The vendor's advice was to increase system's MAXCAD parameter (and to re-IPL the system). This annoying experience has led me to a question: is there a way for me to see how many CADs are on the system as well how many SCOPE=COMMON dataspace are already initialized?

Before proceeding any further let us see what the dataspaces are. In February 1988, IBM announced ESA/370, which provided new capabilities for the IBM ES/3090 processors, including, amongst other things, the access registers. Access registers made it easier to separate application code from data, resulting in improvements in data volume handling capabilities and data isolation. ESA/370 provided a new virtual storage space for data only, called a dataspace. Unlike a cross memory address space, no application code can execute in a dataspace. Dataspaces are data-only spaces that can hold up to 2GB of data and the pages of data in a dataspace are part of the inventory of the owning address space. (A note: from a performance point of view, remember that the system swaps in and out a dataspace as it swaps in and out the address space that dispatched the owning TCB. Thus, dataspaces shared by programs that run in other

address spaces must be owned by TCBs in non-swappable address spaces. When the owning address space terminates, its dataspaces are destroyed.)

Dataspaces provide integrity and isolation for the data they contain in much the same way as address spaces provide integrity and isolation for the code and data they contain. They are a flexible solution to problems related to accessing large amounts of data. Data in a dataspace is byte addressable and can be manipulated directly by a program in an address space.

There are two basic ways to place data in a dataspace. One way is through using buffers in the program's address space. A second way avoids the use of address space virtual storage as an intermediate buffer area. Instead, through data-in-virtual services, a program can move data into a dataspace directly. How to use, as well as analyse, data-in-virtual may be found in 'Analysing data-in-virtual statistics' (*MVS Update*, November 2003, issue 206).

There are three types of dataspace – SCOPE=SINGLE, SCOPE=ALL, and SCOPE=COMMON.

A SCOPE=COMMON dataspace can be used by all programs in the system. It provides a commonly addressable area similar to the Common Storage Area (CSA). SCOPE=COMMON dataspaces are used by MVS and can also be used by subsystems and applications that need common storage. Each SCOPE=COMMON dataspace uses one entry on all primary address space access lists (PASN-AL) in the system. Because the maximum PASN-AL is 250, each SCOPE=COMMON dataspace your program creates and adds to the PASN-AL reduces the number of SCOPE=SINGLE and SCOPE=ALL dataspaces that a program can address through its PASN-AL. Therefore, it is recommended that installations allow each subsystem or application only one SCOPE=COMMON dataspace. Performance hint: bear in mind that all dataspaces are paged to local page datasets, regardless of SCOPE.

The MAXCAD parameter specified in parmlib member

IEASYS*xx* is used to reserve the number of entries available for SCOPE=COMMON dataspaces on all PASN-ALs in the system. In fact, MAXCAD controls the number of Common Area Dataspaces that are preallocated during system initialization and cause space to be taken up in the PASN access list for every address space in the system. The number required depends on the workload/exploiters on your system. When selecting a value for MAXCAD, allow for the SCOPE=COMMON dataspaces that MVS uses, then allow for the SCOPE=COMMON dataspaces that subsystems or applications use. It is interesting to note that the maximum CADs value under z/Architecture mode includes two reserved entries used internally by z/OS that are not accounted for in the user specified MAXCAD value in IEASYS*xx*.

Now we come to the initial question: how does one 'see' a z/OS dataspace in SDSF and in monitors like RMF? From the RMF III option STORF it is possible to view the total FRAME occupied by the ASID (Address Space plus Dataspaces) and from the console with the command **D A,jobname** you can see the number of dataspaces created. Some subsystems provide more meaningful information on their use of dataspaces. For an example try to issue command **D NET,STORUSE,DSPNAME=***, which shows information about all dataspaces accessed by VTAM.

However, what was needed was a procedure that would take a snapshot of the system's real storage and list all dataspaces preallocated during system initialization. A quick and simple way to get the dataspaces that are in use is available from MXI (MVS eXtended Information). MXI is free and available from www.rocketsoftware.com/portfolio/mxi/index2.htm. It does not use any method of CPU serial number protection or encryption. No passwords or activation zaps are required. The neat REXX interface that MXI provides was utilized to get the information needed. The following REXX EXEC (MXIDSP) displays the dataspaces that are in use by using the DSP command.

MXIDSP EXEC

```
/* REXX - EXEC to invoke MXI Dataspaces Report using LIBDEFs     */
/*                                                               */
ADDRESS TSO
 userid=SYSVAR(SYSUID)
 dspr = userid!!'.dsp.rep'
 x = MSG('ON')
 IF SYSDSN(dspr) = 'OK'
 THEN "DELETE "dspr" PURGE"
 "ALLOC FILE(DATASP) DA("dspr")",
    " UNIT(SYSALLDA) NEW TRACKS SPACE(3,1) CATALOG",
    " REUSE RELEASE LRECL(67) RECFM(F B)"
/*---------------------------------------------------------------*/
/* Header for Dataspace initialized report                       */
/*---------------------------------------------------------------*/
  dshd.1 = left('Dataspace Report for System' MVSVAR(SYSNAME),
           ||" running" MVSVAR(SYSOPSYS),12Ø)
  dshd.2 = left(' ',1)
  dshd.3 = left('Report produced on:',19),
           ||left(' ',1,' ')||left(date(),1Ø),
           ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
  dshd.4 = left(' ',1)
  dshd.5 = left('Part 1: Display the dataspaces that are in use',46)
  dshd.6 = left(' ',1)
 "EXECIO * DISKW DATASP (STEM dshd.)"
/*---------------------------------------------------------------*/
/* Header for Dataspace real storage report                      */
/*---------------------------------------------------------------*/
  rshd.1 = left(' ',1)
  rshd.2 = left('Part 2:',8)
  rshd.3 = left('Display the current real storage usage by',42)
  rshd.4 = left('address space using Dataspaces',3Ø)
  rshd.5 = left(' ',1)
  rshd.6 = left('Jobname  ASID      Real',25),
           ||left('WorkSet   Fixed  Data',25)
  rshd.7 = left('-',47,'-')
arg hlq
if hlq = "" then HLQ = 'SYS2.MXI'
address ISPEXEC "LIBDEF ISPTLIB DATASET ID('"hlq".TABLES') STACK"
address ISPEXEC "LIBDEF ISPLLIB DATASET ID('"hlq".LOAD') STACK"
  X = MXIREXX('LINE.','DSP')
  dspt= substr(line.Ø,6,3) - 4             /* No. of dataspaces */
  I =3
   ccad = substr(LINE.i,6,2)               /*   Get current CAD */
   dcad = substr(LINE.i,9,2) - 2           /*   Get defined CAD */
   shd.1 = left('Dataspaces:',11)   left(dspt,6)
   shd.2 = left('Common Area Dataspaces (CAD):',3Ø)
   shd.3 = left('Total   CAD:',12)  left(dcad+2,3)
   shd.4 = left('Defined CAD:',12)  left(dcad,3),
           ||left('  /MAXCAD parm value/',25)
```

```
   shd.5 = left('Current CAD:',12)  left(ccad,3),
           ||left('  /SCOPE=COMMON dataspaces/',29)
   shd.6 = left(' ',1)
   shd.7 = left('DSname   Jobname  ASID ASCB   ',33),
           ||left('ASTE    SToken          Scope',33)
   shd.8 = left('-',64,'-')
 "EXECIO * DISKW DATASP (STEM shd.)"
  DO I = 5 TO LINE.Ø
  j =i-4
      x.1 = STRIP(LINE.I)
 "EXECIO * DISKW DATASP (STEM x.)"
    END
 "EXECIO * DISKW DATASP (STEM rshd.)"
  X = MXIREXX('LINE.','RS')
   Tds4k = Ø
  DO I = 6 TO LINE.Ø
   jobn = word(LINE.i,1)             /*   The address space name*/
   asid = word(LINE.i,2)             /*   The hexadecimal ASID  */
   rsto = word(LINE.i,3)             /*   No. of RS frames used */
   wkst = word(LINE.i,5)             /*The working set size (K) */
   fsto = word(LINE.i,6)               No. of fixed frames used */
   ds4k = word(LINE.i,8) /* No.of user key dataspace blocks (4K)*/
   Tds4k = tds4k + ds4k  /*Sum of user key dataspace blocks (4K)*/
   Select
     when ds4k > Ø then do
     y.1 = left(jobn,8)   left(asid,7) ,
           right(rsto,6)  right(wkst,7) ,
           right(fsto,7)  right(ds4k,6)
 "EXECIO * DISKW DATASP (STEM y.)"
   end
   otherwise nop
  end
 end
  rst.1  = left(' ',4Ø,' ') left('-',5,'-')
  rst.2  = left('Total 4k ds pages:',39) right(tds4k,6)
 "EXECIO * DISKW DATASP (STEM rst.)"
  ADDRESS ISPEXEC "LIBDEF ISPLLIB";
  ADDRESS ISPEXEC "LIBDEF ISPTLIB";

/*  Close & free report file    */

  "EXECIO Ø DISKW DATASP  (FINIS "
  "free FILE(DATASP)"
  Address ISPEXEC
 "ISPEXEC BROWSE DATASET('"dspr"')"
 exit
```

The EXEC gives the following output (without the headings and with some lines cut out that are similar to those shown):

```
Part 1: Display the dataspaces that are in use
Dataspaces: 112
Common Area Dataspaces (CAD):
Total   CAD: 27
Defined CAD: 25   /MAXCAD parm value/
Current CAD: 15   /SCOPE=COMMON dataspaces/

DSname   Jobname  ASID ASCB      ASTE    SToken           Scope
-------------------------------------------------------------
CCOFGSDO DLF      0019 00FA9280 7CC14F00 8000310000000057
CCSVLLA  VLF      0018 00FA9400 7B428100 8000340000000063
CSFDS001 CSFSTART 0022 00F3F280 7FA41000 8000080200000013
CSM31002 *MASTER* 0001 00FC7900 7F009800 80002C000000007A COMMON
CSM64001 *MASTER* 0001 00FC7900 7F009680 80002B0000000079 COMMON
CSVRTLS  *MASTER* 0001 00FC7900 7F009380 8000180000000018 COMMON
DCSVLLA  VLF      0018 00FA9400 7FA41080 8000060200000012
DFHDT001 CICSPROD 003A 00FB4280 7CC14C80 80004400000019E7
EPWFDSPC EPWFFST  0021 00F3F400 7F009600 8000120100000013 COMMON
EPWPDSPC EPWFFST  0021 00F3F400 7F009580 800005020000000F COMMON
EPWS76QU EPWFFST  0021 00F3F400 7F009700 8000130100000014 COMMON
ERBENQDS RMF      001A 00FA9100 6F925080 80004A0000000112
ERBOPDDS RMFGAT   0076 00FB4D00 7CC14E80 80004D0000000113
ERBWSTOR RMFGAT   0076 00FB4D00 7FA41100 800009020000030A
HFSDSPS1 OMVS     000E 00F40100 7CC14800 8000200000000038
HFSDSP01 OMVS     000E 00F40100 7CC14A00 8000220000000039
IEECMCAD *MASTER* 0001 00FC7900 7F009400 800001020000000001 COMMON
IEEMCNG  CONSOLE  000A 00F89100 7FC8F080 8000020100000003
IEEMCS01 CONSOLE  000A 00F89100 7CC14600 8000190000000019
IEEMCS02 CONSOLE  000A 00F89100 7CC14D00 80003C00000000A1
IEEMCS03 CONSOLE  000A 00F89100 7CC14F80 80004800000000B1
IRR00001 *MASTER* 0001 00FC7900 7FC8F100 8000030100000004
IRR00002 *MASTER* 0001 00FC7900 7FC8F180 8000040100000005
.............................................................
IST394F0 VTAM     001E 00F3F880 7F009900 8000370000000088 COMMON
IXCAP1DS XCFAS    0006 00F89400 7CC14480 80000B000000000B
IXCDSTKF XCFAS    0006 00F89400 7EFBE380 8000150000000015
IXLCTCAD XCFAS    0006 00F89400 7F009200 80000E000000000E COMMON
JES00001 JESXCF   0010 00F8A180 7EFBE600 8000270000000003E
JES00002 JESXCF   0010 00F8A180 7EFBE780 8000320000000060
JES2CKVR JES2AUX  0029 00FA3680 7EFBE800 8000330000000061
.............................................................
SDUMPRDS DUMPSRV  0005 00F89580 7CC14100 8000040000000004
SDUMPRSD *MASTER* 0001 00FC7900 7EFBE000 8000010000000001
SDUMPSDS DUMPSRV  0005 00F89580 7CC14080 8000030000000003
SYSANT00 ANTMAIN  000C 00F3DA80 7F009500 8000020200000009 COMMON
SYSANT01 ANTAS000 000D 00F40280 7F009780 8000070200000011 COMMON
SYSBMFDS SYSBMAS  0009 00FAE780 7F009300 8000170000000017 COMMON
SYSDS000 RASP     0003 00F89880 7FF14040 800000007FF14040
SYSIOS00 IOSAS    0012 00FB3D00 7C4DA080 800004020000000E
SYSIXG01 IXGLOGR  0013 00FB3580 7EFBED80 80004000000000A5
```

```
SYSLOGRØ IXGLOGR  ØØ13 ØØFB358Ø 7EFBE48Ø 8ØØØ1FØØØØØØØØ32
SYSZBPXU OMVS     ØØØE ØØF4Ø1ØØ 7EFBE58Ø 8ØØØ1DØØØØØØØ37
SYSZBPX1 OMVS     ØØØE ØØF4Ø1ØØ 7FØØ948Ø 8ØØØ1BØØØØØØØ1B COMMON
SYSZBPX2 OMVS     ØØØE ØØF4Ø1ØØ 7EFBE5ØØ 8ØØØ23ØØØØØØØ35
SYSZBPX3 OMVS     ØØØE ØØF4Ø1ØØ 7CC149ØØ 8ØØØ21ØØØØØØØ36
SYSZILM1 *MASTER* ØØØ1 ØØFC79ØØ 7FØØ928Ø 8ØØØ16ØØØØØØØ16 COMMON
TCPIPDS1 TCPIP    ØØ1D ØØF3FAØØ 7FØØ988Ø 8ØØØ29ØØØØØØØ7B COMMON
ØØØØØEDC TCPIP    ØØ1D ØØF3FAØØ 7CC14EØØ 8ØØØ28ØØØØØØØ7C
ØØØØØIEF JES2     ØØ23 ØØF8C4ØØ 7CC14D8Ø 8ØØØ2AØØØØØØØ74
Part 2:
Display the current real storage usage by
address space using Dataspaces
Jobname  ASID      Real  WorkSet   Fixed  Data
-----------------------------------------------
CICSPROD ØØ3A     27ØØ6  1Ø8Ø24     9Ø5   4Ø96
RMFGAT   ØØ76      9485   3794Ø      88  16384
CICSTEST ØØ6F      7944   31776     172   4Ø96
CICSTRAN ØØ7F      6368   25472     141   4Ø96
                                      -----
Total 4k ds pages:                    28672
```

Output fields legend for part 1 of the report:

- DSname – name of the dataspace.
- Jobname – the jobname of the address space that owns the dataspace.
- ASID – hex ASID of the owning address space.
- ASCB – the address of the ASCB of the owning address space.
- ASTE – the ASTE address for the dataspace.
- SToken – hex token of the dataspace.
- Scope – whether the dataspace is SCOPE=COMMON or not.

Output fields legend for part 2 of the report:

- Jobname – the address space name.
- ASID – the hexadecimal ASID.
- Real – the number of real storage frames used.
- WorkSet – the working set size (in K) represents the

storage the user has when the job is actually running. Shared page counts are not included in the WorkSet.

- Fixed – the number of fixed frames used. (A fixed frame is a frame that cannot be paged out of central storage.)
- Data – the number of user key dataspace blocks (4K).

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Which characters are present in my datasets – character check and comparison routine

INTRODUCTION

We've been having problems recently with a product from a third-party supplier. The components we have comprise a performance reporting system on the mainframe, which delivers reports held on a server, that are displayed with the help of a browser.

The installation of the product provided a chance for innovation, because the 'works on all platforms' solution for the mainframe assumed that IBM's WebSphere was implemented. We have only the first non-free version in our shop and this was too old for the software. We were then faced with the choice of:

- Purchasing WebSphere (very expensive for one application!).
- Acquiring a new Windows Server (also not cheap).
- Installing the product on the Webserver (USS) with the help of Tomcat.

As you can probably guess we took the gamble and tried to install Tomcat on the mainframe along with the product version for Windows Tomcat.

Surprisingly, after lots of trial and error we (a colleague with extensive Unix and MVS skills and myself) succeeded in this.

The main problem we are now encountering is that while several reports are OK and can be displayed perfectly, others just fail.

We believe this problem exists because the software was developed and tested in America and is now running on a system in Germany. Basically, our codepages are different and it could be that they are simply not correctly converted. The product on the mainframe generates reports in XML format in EBCDIC, which are then converted to ASCII to be used on the server.

To help us track down the problem and to be able to report our findings, I have developed the REXX routine CHARCHK. This routine goes through two datasets and registers in hexadecimal which characters have been used and which characters appear in one dataset and not in the other.

The routine can be called from JCL (changed by hand), from TSO direct, or from JCL generated with the use of the second REXX (ISPF) routine CHKCHAR.

INTERNAL WORKINGS

After what seemed to be a huge amount of thought as to how I could compare two datasets and locate characters that exist in one and not in the other I stumbled upon a relatively simple idea.

I create two arrays (in stem variables) that are accessed with an index between 0 and 255. These are the decimal values of the hexadecimal values X'00' to X'FF', ie the complete codepage character set.

What I did then was simply initialize these arrays to zero. Each character in a dataset is read and its hexadecimal value is converted to decimal; then the array cell indexed by this value is incremented by 1.

At the end of the dataset we have an array with positive values or zero. The array cells with a value of zero have not been found in the corresponding dataset.

When both datasets have been read we just need to compare the values contained in the two arrays. For example, array1.217 with a value of 72 shows that the character ‘R’ has been found 72 times in dataset 1 (217 decimal = ‘D9’ hexadecimal = character ‘R’ (EBCDIC)).

The routine CHKCHAR calls ISPF panels, which allow easier selection of the datasets to be compared. This includes listing panels when high-level qualifiers (and wildcards) are used.

Then the routine builds the JCL that will be used to call routine CHARCHK.

EXAMPLE

```
##########################################
# Characters found in both datasets:   #
##########################################
#     Ø       #           #     ØØ     #
#     64      #           #     4Ø     #
#     75      #     .     #     4B     #
#     76      #     <     #     4C     #
#     77      #     (     #     4D     #
#     8Ø      #     &     #     5Ø     #
#     9Ø      #     £     #     5A     #
#     91      #     $     #     5B     #
#     93      #     )     #     5D     #
#     94      #     ;     #     5E     #
#     96      #     -     #     6Ø     #
#     97      #     /     #     61     #
#     1Ø7     #     ,     #     6B     #
#     1Ø9     #     _     #     6D     #
#     11Ø     #     >     #     6E     #
#     111     #     ?     #     6F     #
#     122     #     :     #     7A     #
#     126     #     =     #     7E     #
#     127     #     „     #     7F     #
#     129     #     a     #     81     #
#     13Ø     #     b     #     82     #
#     131     #     c     #     83     #
#     132     #     d     #     84     #
#     133     #     e     #     85     #
```

```
#     134     #     f     #     86     #
#     135     #     g     #     87     #
#     136     #     h     #     88     #
#     137     #     i     #     89     #
#     145     #     j     #     91     #
#     146     #     k     #     92     #
#     147     #     l     #     93     #
#     148     #     m     #     94     #
#     149     #     n     #     95     #
#     150     #     o     #     96     #
#     151     #     p     #     97     #
```

SUMMARY

So far we have been able to determine that when we request codepage IBM-273 conversion from the software, the character represented by X'B5' is being falsely converted to X'7C'. Naturally, we have reported this to the supplier of the software.

Other characters have been found in failing reports that are normal in Germany but seem to be causing problems when converted to ASCII and used on the server.

The investigation continues!

TRICK USE

Create a dataset containing all the ‘site accepted’ characters, and use this as one of the datasets to identify unacceptable characters found in the other dataset.

SUPPLIED CODE

The following code is included:

- JCL – CHKCHAR, which is example JCL for those who don’t need ISPF input/selection.
- CHKCHAR REXX – the ISPF controller and JCL generator routine.
- CHARCHK – the character recognition and comparison routine.

- Panels – CHKCHAR (main panel) and CHKCDSP1 (dataset list panel).
- HELP panels – CHKC1H (main help panel), CHKC1H1 (help panel), and CHKC1H2 (help panel).
- JCL skeleton – CHKCJCL (JCL skeleton).

CHKCHAR JCL

```
//AL13745# JOB (SRØØ,SRØ16882,SRØ16882),'CHARCHCK',
//             CLASS=Ø,MSGCLASS=H,REGION=ØM,
//             NOTIFY=AL13745,MSGLEVEL=(1,1)
/*JOBPARM L=Ø1ØØ,G=99999
//*  -------------------------------------------------------------
//*
//*  Batch job to run the character checking routine CHARCHK
//*  This routine compares the characters found in two datasets
//*  and reports on where the characters have been found.
//*  The second parameter is a switch for the trace function of REXX.
//*  Default is 'o' for OFF.
//*
//*  -------------------------------------------------------------
//REXXØ1  EXEC PGM=IKJEFT1A
//SYSEXEC  DD  DISP=SHR,DSN=AL13745.ISPF.EXEC
//CHKCRES  DD  DISP=MOD,DSN=AL13745.CHKCHAR.RESULTS
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD *
 %CHARCHK O +
 'ALEDS.STROBE.DCCP2Ø1.XØØ1DØØ1' +
 'ALEDS.STROBE.DCCP2Ø2.XØØ1DØØ1'
/*
```

CHKCHAR REXX EXEC

```
/* REXX */
/* ****************************************************************** */
/* ### CHKCHAR  ### */
/* ------------------------------------------------------------------ */
/* Description:                                                       */
/* Panel control routine for JCL generator for routine CHARCHK.       */
/* Routine to check which characters exist in one dataset and compare */
/* them with the characters found in another dataset.                 */
/*                                                                    */
/* ------------------------------------------------------------------ */
/* Created ....: 1Ø.Ø9.2ØØ4      Rolf Parker                          */
/* ------------------------------------------------------------------ */
```

```
/* Updates ....: ØØ.ØØ.ØØØØ xx None                                    */
/* ******************************************************************* */
trace o
/* ******************************************************************* */
/* * Initialization * BEGIN ************************************** */
/* ******************************************************************* */
/* Definition variables for CHKCHAR.RESULTS store                      */
/*      "STORCLAS(BAØØ) MGMTCLAS(MALU0###)",                           */
/*      "VOLUME(ALUØØ4) UNIT(339Ø)",                                   */
/*      "SPACE(2,5) TRACKS"                                            */
  stor1 = 'BAØØ'
  mgmt1 = 'MALU0###'
  vol1  = 'ALUØØ4'
  unit1 = '339Ø'
  spc1  = '2,5'
  dfltdsn = "'" || SYSVAR(SYSPREF) || ".CHKCHAR.RESULTS" || "'"
/* ******************************************************************* */
/* Definition variables for JCL DSN                                    */
/*      "STORCLAS(BAØØ) MGMTCLAS(MALU0###)",                           */
/*      "VOLUME(ALUØØ4) UNIT(339Ø)",                                   */
/*      "SPACE(1,1) TRACKS"                                            */
  stor2 = 'BAØØ'
  mgmt2 = 'MALU0###'
  vol2  = 'ALUØØ4'
  unit2 = '339Ø'
  spc2  = '1,1'
  jcldsn = "'" || SYSVAR(SYSPREF) || ".chkchar.TEMP" || "'"
/* ******************************************************************* */
  chkresddn = 'CHKCRES'
  jclddn = 'JCLDDN'
  srcdsn1 =''
  srcdsn2 =''
  srcdsn.1 =''
  srcdsn.2 =''
  pfkey  =''
/* * Initialization * END **************************************** */
/* */
/* ******************************************************************* */
/* * Main loop for chkchar                                         * */
/* * Display and check input PANEL 'chkchar' until:                * */
/* * Abandoned with PFKEY PF4                                      * */
/* * Exited for further processing with PFKEY PF3                 * */
/* *   and checked input is OK                                    * */
/* ******************************************************************* */
do forever
  panelrc = ''
  'ISPEXEC DISPLAY PANEL(chkchar)'
  panelrc = rc
  if panelrc = 8  & pfkey = 'PFØ4' then leave
  contents = ''
```

```
  call check_input
  if panelrc = 8  & pfkey = 'PFØ3' & contents = 'OK' then leave
end
  if pfkey = 'PF04' then exit
  call build_jcl
  rc = MSG('OFF')
  "FREE DSNAME("jcldsn")"
  rc = MSG('ON')
return
/* ********************************************************************* */
/* * check_input and subroutines ********************************** */
/* *** check_input  ************************************************** */
check_input:
  srcdsn.1 = srcdsn1
  srcdsn.2 = srcdsn2
  do index = 1 to 2
    call check_srcdsn
  end
  srcdsn1=srcdsn.1
  srcdsn2=srcdsn.2
  /*                                                                   */
  /* check_rc -- at end of step if greater than Ø then validation      */
  /* incomplete                                                        */
  check_rc = Ø
  /* if no override dataset is specified use the default               */
  if outdsn = '' then outdsn = dfltdsn
  call check_outdsn
  rc = MSG('OFF')
  /* dots used to avoid high-level Alias being used as qualified       */
  /* dataset                                                           */
  dots = POS('.',srcdsn1)
  sysdsn = SYSDSN(srcdsn1)
  if ( SYSDSN = 'OK' & dots > Ø ) | srcdsn1 = '' then
    do
      nop
    end
    else
    do
      check_rc = check_rc + 1
    end
  dots = POS('.',srcdsn2)
  sysdsn = SYSDSN(srcdsn2)
  if ( SYSDSN = 'OK' & dots > Ø ) | srcdsn2 = '' then
    do
      nop
    end
    else
    do
      check_rc = check_rc + 1
    end
```

```
  rc = MSG('ON')
  /* if check_rc has remained Ø then input has been validated as OK   */
  if check_rc = Ø then contents = 'OK'
return
/* ******************************************************************* */
/* *** check_srcdsn ************************************************** */
/* ******************************************************************* */
check_srcdsn:
  i = index
  srcdsn = srcdsn.i
  q = i + 1
  q = SUBSTR(q,1,1)
  DSNLDS = ''
  Tablename         = '$'q''Time('S') /* Unique Named Temp Table */
  TableVarNames     = 'Display1'
  srcdsn = STRIP(srcdsn,'B')
  if SUBSTR(srcdsn,1,1) = "'" then
    nop
  else
    do
      srcdsn = "'" || SYSVAR(SYSPREF) || "." || srcdsn
      if SUBSTR(srcdsn,2,1) = "." then
        srcdsn = "'" || SUBSTR(srcdsn,'3')
        srcdsn = STRIP(srcdsn,'T','.')
    end
  srcdsn = STRIP(srcdsn,'B',"'")
  srcdsn_a = "'"srcdsn"'"
  dots = POS('.',srcdsn)
  rc = MSG('OFF')
  sysdsn = SYSDSN(srcdsn_a)
  rc = MSG('ON')
  if SYSDSN = 'OK' & dots > Ø then
    do
      srcdsn = srcdsn_a
      return
    end
  if srcdsn = '' then srcdsn = "'"userid()
  srchmsg1 = 'Listing for '
  srchmsg2 = srcdsn
  dspcnt = Ø
  disp.  = ''
  'ISPEXEC LMDINIT LISTID(list2)
                  LEVEL('srcdsn')'
  lmdinit_rc = rc
  title  = 'SOURCE SELECTION LIST'
  title2 = 'SELECT A DATASET FROM THE LIST'
  lmdlist_rc = Ø
  do while lmdlist_rc = Ø
    'ISPEXEC LMDLIST LISTID('list2')
                     OPTION(LIST)
```

```
                    DATASET(DSNLDS)
                    STATS(NO)'
  lmdlist_rc = rc
  if lmdlist_rc = 0 then
    do
      dspcnt = dspcnt + 1
      disp.dspcnt = left(DSNLDS,44,' ')
    end
  if lmdlist_rc = 4 then
    do
      srchmsg1 = 'No Datasets found for '
      srchmsg2 = srcdsn
      title2 = ''
    end
end
'ISPEXEC LMDLIST LISTID('list2') OPTION(FREE)'
    lmdlist_rc = rc
'ISPEXEC LMDFREE LISTID('list2')'
    lmdfree_rc = rc
'ISPEXEC TBCREATE 'Tablename' NAMES('TableVarNames') NOWRITE SHARE'
disp.0 = dspcnt
do l = 1 to disp.0
   parse var disp.l display1
   'ISPEXEC TBADD' Tablename
end
row_ind = ''
'ISPEXEC TBTOP' Tablename
'ISPEXEC TBDISPL 'Tablename' PANEL(CHKCDSP1)'
index_used = ''
do while ZTDSELS > 0
  do
    'ISPEXEC TBGET' Tablename
    srcdsn = STRIP(display1,'B')
    srcdsn = "'" || srcdsn || "'"
    if index_used = 0 | index_used = 1 then
      do
        if spare > 0 then
        do
          spindex = 7 - spare
          srcdsn.spindex = srcdsn
          spare = spare -1
        end
        index_used = 1
      end
    if index_used = '' then
      do
        index_used = 0
        srcdsn.index = srcdsn
      end
  end
```

```
    if index_used & spare < 1 then leave
    if ZTDSELS > 1 then
      'ISPEXEC TBDISPL 'Tablename
    else ZTDSELS = ZTDSELS - 1
  end
  'ISPEXEC TBEND' Tablename
return
/* ******************************************************************* */
/* *** check_outdsn ************************************************** */
check_outdsn:
  dots = POS('.',outdsn)
  rc = MSG('OFF')
  sysdsn = SYSDSN(outdsn)
  select
    when ( SYSDSN = 'OK' & dots > Ø ) then
      do
        ADDRESS TSO
        "FREE DSNAME("outdsn")"
        "ALLOC FI("chkresddn") DA("outdsn") SHR REUSE "
      end
    when ( SYSDSN = 'DATASET NOT FOUND' & dots > Ø ) then
      do
        ADDRESS TSO
        "FREE DSNAME("outdsn")"
        "ALLOC DD("chkresddn") NEW CATALOG REUSE ",
        "DSN("outdsn") ",
        "LRECL(8Ø) BLKSIZE(3272Ø) RECFM(F,B) ",
        "DSORG(PS) ",
        "STORCLAS("stor1") MGMTCLAS("mgmt1")",
        "VOLUME("vol1") UNIT("unit1")",
        "SPACE("spc1") TRACKS"
      end
    otherwise
      do
        outdsn = dfltdsn
        call check_outdsn
      end
  end
  "FREE DSNAME("outdsn")"
  rc = MSG('ON')
return
/* * check_input and subroutines ************************************** */
/* ******************************************************************* */
/* */
/* ******************************************************************* */
/* * build_jcl ********************************************************* */
build_jcl:
  outdsn = strip(outdsn,'B',"'")
  rc = MSG('OFF')
  ADDRESS TSO
```

```
  "FREE DSNAME("jcldsn")"
  "DELETE "jcldsn
  "ALLOC DD(ISPFILE) NEW CATALOG REUSE ",
  "DSN("jcldsn") ",
  "LRECL(8Ø) BLKSIZE(32720) RECFM(F,B) ",
  "DSORG(PS) ",
  "STORCLAS("stor2") MGMTCLAS("mgmt2")",
  "VOLUME("vol2") UNIT("unit2")",
  "SPACE("spc2") TRACKS"
/* ******************************************************************* */
  "ISPEXEC FTOPEN "
  "ISPEXEC FTINCL CHKCJCL"
  "ISPEXEC FTCLOSE "
  "ISPEXEC EDIT DATASET("jcldsn") "
/* ******************************************************************* */
  rc = MSG('ON')
return
/* * build_jcl ******************************************************** */
/* ******************************************************************* */
```

CHARCHK REXX

```
/* REXX                                                                */
/* ******************************************************************* */
/* ### CHARCHK  ###                                                    */
/* ------------------------------------------------------------------- */
/* Description:                                                        */
/* Routine to check which characters exist in one dataset and compare */
/* them with the characters found in another dataset.                  */
/*                                                                     */
/* ------------------------------------------------------------------- */
/* Created ....: Ø7.Ø9.2ØØ4      Rolf Parker                           */
/* ------------------------------------------------------------------- */
/* Updates ....: ØØ.ØØ.ØØØØ xx None                                    */
/* ******************************************************************* */
/* ******************************************************************* */
/* trace switch                                                        */
/* to enable the control of the trace mode by use of a parameter       */
/* All, Commands, Error, Failure, Intermediates, Labels, Normal        */
/* Off, Results and Scan                                               */
/* ******************************************************************* */
trace_ok = Ø
parse upper arg traceswitch dsn1 dsn2 dsn3
traceswitch = substr(traceswitch,1,1)
if traceswitch = 'A' | ,
   traceswitch = 'C' | ,
   traceswitch = 'E' | ,
   traceswitch = 'F' | ,
   traceswitch = 'I' | ,
```

```
   traceswitch = 'L' | ,
   traceswitch = 'N' | ,
   traceswitch = 'O' | ,
   traceswitch = 'R' | ,
   traceswitch = 'S' then trace_ok = 1
if trace_ok then
  nop
else
  traceswitch = 'o'
/* ******************************************************************* */
interpret 'trace' traceswitch
/* ******************************************************************* */

parse SOURCE src1 src2 src3 src4 src5 src6 src7 src8 src9
/* ******************************************************************* */
  retcode = 0
  q = 0
  dd1 = 'CHKFILE1'
  dd2 = 'CHKFILE2'
  dd3 = 'CHKCRES'
/* ******************************************************************* */
/* if the routine is called directly from TSO and not through JCL     */
/* then check that the third dataset name is valid - if not then      */
/* create a temporary dataset for the results of the comparison.      */
/* ******************************************************************* */
  if src8 = 'ISPF' then
    do
      if dsn3 <> '' & SYSDSN(dsn3) = 'OK' then
        do
          "FREE DSNAME("dsn3")"
          "ALLOC DD("dd3") DSN("dsn3") SHR MOD "
        end
      else
        do
          stor3 = 'BA00'
          mgmt3 = 'MALU0###'
          vol3  = 'ALU004'
          unit3 = '3390'
          spc3  = '1,1'
          dsn3  ="'"userid()".D"||date('J')||".TEMP'"
          "FREE DSNAME("dsn3")"
          "DELETE "dsn3
          "ALLOC DD("dd3") NEW CATALOG REUSE ",
          "DSN("dsn3") ",
          "LRECL(80) BLKSIZE(32720) RECFM(F,B) ",
          "DSORG(PS) ",
          "STORCLAS("stor3") MGMTCLAS("mgmt3")",
          "VOLUME("vol3") UNIT("unit3")",
          "SPACE("spc3") TRACKS"
        end
```

```
    end
/* ******************************************************************* */
/* when available shows the origin of this called REXX routine         */
/* if unavailable the various variables will be shown as '?'           */
/* src3 = member name                                                  */
/* src5 = dataset name                                                 */
/* src4 = DD name                                                      */
/* ******************************************************************* */
say  '--------------------------------------------------------------'
say 'running: 'src3' from: 'src5' DD: 'src4
say  '--------------------------------------------------------------'
q = q + 1
ot.q  ='-------------------------------------------------------------
-'
q = q + 1
ot.q ='running: 'src3' from: 'src5' DD: 'src4
q = q + 1
ot.q  ='-------------------------------------------------------------
-'
/* ******************************************************************* */
/* direct naming of the input datasets simply change the names and     */
/* uncomment - useful for testing purposes                             */
/* ******************************************************************* */
/*dsn1 = "'ALEDS.STROBE.PALLF011.X001D001'"*/
/*dsn2 = "'ALEDS.STROBE.PALLF005.X001D001'"*/

/* ******************************************************************* */
/* check and proceed if input datasets are OK                          */
/* ******************************************************************* */
  ADDRESS 'TSO'

  if SYSDSN(dsn1) = 'OK' & SYSDSN(dsn2) = 'OK' then
    do
      list1 = ''
      list2 = ''
      "FREE DSNAME("dsn1")"
      "ALLOC DD("dd1") DSN("dsn1") SHR MOD "
      x = LISTDSI(dsn1)
      lrecl1 = SYSLRECL
      lreclx = lrecl1
      seldsn = dsn1
      seldd  = dd1
      listx = ''
      call listchar
      list1 = listx
say  '--------------------------------------------------------------'
      say 'list1 =' list1
say  '--------------------------------------------------------------'
      do p = 0 to 255
        charray1.p = charray.p
```

```
      end
      "FREE DSNAME("dsn2")"
      "ALLOC DD("dd2") DSN("dsn2") SHR MOD "
      x = LISTDSI(dsn2)
      lrecl2 = SYSLRECL
      lreclx = lrecl2
      seldsn = dsn2
      seldd  = dd2
      listx = ''
      call listchar
      list2 = listx
say  '------------------------------------------------------------------'
      say 'list2 =' list2
say  '------------------------------------------------------------------'
      do p = Ø to 255
        charray2.p = charray.p
      end
      call compchar
    end
'EXECIO 'q' DISKW  'dd3' (STEM ot. FINIS'
if src8 = 'ISPF' then
  do
    address TSO
    "ISPEXEC EDIT DATASET("dsn3") "
  end
exit retcode
/* *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*- */
/* -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-* */
/* *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*- */

/* ******************************************************************** */
/* listchar:                                                            */
/* read complete dataset, if non-empty then proceed with character      */
/* comparison.                                                          */
/* ******************************************************************** */
listchar:
  'EXECIO * DISKR  'seldd' (STEM infile. FINIS'
  if rc = Ø then
    do
      if infile.Ø > Ø then
        do
          say '----------------------------------------------------------'
          say 'processing ' infile.Ø 'records from' seldsn
          say '----------------------------------------------------------'
          q = q + 1
          ot.q ='------------------------------------------------------------'
          q = q + 1
          ot.q ='processing ' infile.Ø 'records from' seldsn
          q = q + 1
          ot.q ='------------------------------------------------------------'
```

```
          call examrcrd
        end
      else
        do
          say '-----------------------------------------------------'
          say 'dataset ' seldsn ' is EMPTY'
          say '-----------------------------------------------------'
        end
    end
  else
    do
       retcode = rc
       say '-----------------------------------------------------'
       say 'error reading dataset ' seldsn
       say '-----------------------------------------------------'
    end
  "FREE DSNAME("seldsn")"
return

/* ******************************************************************* */
/* examrcrd:                                                           */
/* examine each record                                                 */
/* ******************************************************************* */
examrcrd:
  do k = 0 to 255
    charray.k = 0
  end
                    /* prepare position check and timer        */
  load = 0
  loadtot = 0
  pro100 = infile.0
  pro10  = trunc(pro100/10)
  stage = 0
  seconds = trunc(time('R'))
                    /* for each record in dataset              */
  do i = 1 to infile.0
    load = load + 1
                    /* check for stages of 10%                 */
    if load = pro10 then
      do
        seconds = left(trunc(time('E')),3,' ')
        if seconds <> 1 then stext = 'seconds'
        else stext = 'second'
        stage = stage + 1
        loadtot = stage * 10
        loadtot = loadtot || '% in' seconds stext
        q = q + 1
        ot.q = loadtot
        say loadtot
        load = 0
```

```
      end
    call examchar
  end
/* ******************************************************************* */
/* the array has one cell per possible character                       */
/* the characters are represented by x'00' to x'FF'                    */
/* ie 0 to 255                                                         */
/* ******************************************************************* */
  do l = 0 to 255
    if charray.l > 0 then
      do
        char = d2c(l)
        listx = listx || char
      end
  end

return

/* ******************************************************************* */
/* examchar:                                                           */
/* examine the characters, note each unique character                  */
/* ******************************************************************* */
examchar:
  do j = 1 to lreclx
    char = substr(infile.i,j,1)
    indx = c2d(char)
    charray.indx = charray.indx + 1
  end
return

/* ******************************************************************* */
/* compchar:                                                           */
/* compare the characters found in each dataset                        */
/* ******************************************************************* */
compchar:
  do n = 0 to 255
    charfnd1.n = 0
    charfnd2.n = 0
    charfndb.n = 0
    charfndn.n = 0
  end
/* ******************************************************************* */
/* compare the arrays from the two datasets and establish which        */
/* characters are in which dataset                                     */
/* ******************************************************************* */
  do m = 0 to 255
    select
      when charray1.m = 0 & charray2.m = 0 then
        do
          charfndn.m = 1
```

```
        end
      when charray1.m > 0 & charray2.m > 0 then
        do
          charfndb.m = 1
        end
      when charray1.m > 0 & charray2.m = 0 then
        do
          charfnd1.m = 1
        end
      when charray1.m = 0 & charray2.m > 0 then
        do
          charfnd2.m = 1
        end
      otherwise
        nop
    end
  end
/* ********************************************************************* */
/* report the findings                                                   */
/* ********************************************************************* */
  say '#########################################'
  say '# Characters found in both datasets:   #'
  say '#########################################'
  q = q + 1
  ot.q ='#########################################'
  q = q + 1
  ot.q ='# Characters found in both datasets:   #'
  q = q + 1
  ot.q ='#########################################'
  do o = 0 to 255
    if charfndb.o > 0 then
      do
        char = d2c(o)
        hex = c2x(char)
        o = left(o,3,' ')
        say '#     'o'     #     'char'     #     'hex'     #'
        q = q + 1
        ot.q ='#     'o'     #     'char'     #     'hex'     #'
      end
  end
  say '#########################################'
  say '-----------------------------------------'
  say '#########################################'
  say '# Characters found in neither dataset: #'
  say '#########################################'
  q = q + 1
  ot.q ='#########################################'
  q = q + 1
  ot.q ='# Characters found in neither dataset: #'
  q = q + 1
```

```
ot.q ='#########################################'
do o = Ø to 255
  if charfndn.o > Ø then
    do
      char = d2c(o)
      hex = c2x(char)
      o = left(o,3,' ')
      say '#     'o'     #     'char'     #     'hex'     #'
      q = q + 1
      ot.q ='#     'o'     #     'char'     #     'hex'     #'
    end
end
say '#########################################'
say '-----------------------------------'
say '#########################################'
say '# Characters found only in:            #'
say '# 'dsn1
say '#########################################'
q = q + 1
ot.q ='#########################################'
q = q + 1
ot.q ='-----------------------------------'
q = q + 1
ot.q ='#########################################'
q = q + 1
ot.q ='# Characters found only in:            #'
q = q + 1
ot.q ='# 'dsn1
q = q + 1
ot.q ='#########################################'
do o = Ø to 255
  if charfnd1.o > Ø then
    do
      char = d2c(o)
      hex = c2x(char)
      o = left(o,3,' ')
      say '#     'o'     #     'char'     #     'hex'     #'
      q = q + 1
      ot.q ='#     'o'     #     'char'     #     'hex'     #'
    end
end
say '#########################################'
say '-----------------------------------'
say '#########################################'
say '# Characters found only in:            #'
say '# 'dsn2
say '#########################################'
q = q + 1
ot.q ='#########################################'
q = q + 1
```

```
  ot.q ='---------------------------------------'
  q = q + 1
  ot.q ='#######################################'
  q = q + 1
  ot.q ='# Characters found only in:            #'
  q = q + 1
  ot.q ='# 'dsn2
  q = q + 1
  ot.q ='#######################################'
  do o = Ø to 255
    if charfnd2.o > Ø then
      do
        char = d2c(o)
        hex = c2x(char)
        o = left(o,3,' ')
        say '#     'o'     #     'char'     #     'hex'     #'
        q = q + 1
        ot.q ='#     'o'     #     'char'     #     'hex'     #'
      end
  end
  say '#######################################'
  q = q + 1
  ot.q ='#######################################'
return
```

CHKCHAR PANEL

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  ]   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
  }   TYPE(TEXT) INTENS(HIGH) COLOR(RED)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(7Ø,42) OUTLINE(BOX)
%/-/#Compare characters found in two datasets ] /-/
$zuser   +/ /$timestmp        +
+Command ===>_zcmd                                          +
+/ /[info                   / /+
/ /]  CCC  H   H K   K {  CCC H   H   A   RRRR  / /
/ /] C     H   H K  K  { C    H   H  A A  R  R  / /
/ /]C      H   H K K   {C     H   H A   A R  R  / /
/ /]C      HHHHH KK    {C     HHHHH AAAAA RRR   / /
/ /]C      H   H K K   {C     H   H A   A R R   / /
/ /] C     H   H K  K  { C    H   H A   A R  R  / /
/ /]  CCC  H   H K   K {  CCC H   H A   A R   R / /
/ /]                                          / /
+
```

```
{ Datasets to Compare:
{  1. £z                                                          +
{  2. £z                                                          +
+
{ Write output to Dataset:
{ £z                                                            +
+
+
/ /%<ENTER>+Check Input    %PF3+Proceed    %PF4+Cancel/ /
+
+
+
)INIT
.HELP = CHKC1H
.zvars ='(srcdsn1,srcdsn2,outdsn)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
)REINIT
REFRESH(*)
)PROC
IF (.MSG NE '')
  &info = '######### ERROR #########'
IF (&useout  NE 'Y')
  &useout  = 'N'
&pfkey = &zpfkey
)HELP
FIELD(srcdsn1) PANEL(CHKC1H1)
FIELD(srcdsn2) PANEL(CHKC1H1)
FIELD(outdsn)  PANEL(CHKC1H2)
)END
```

CHKCDSP1 PANEL

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  #   TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
  ]   TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ)
  {   TYPE(OUTPUT) INTENS(HIGH) COLOR(WHITE)
)BODY DEFAULT(%+_) EXPAND(//)
+                                                                +
+Command==>_zcmd                                 / /+Scroll=>_AMT +
$zuser    +/ /]title                              +/ /$timestmp        +
+/ /[info                   / /+
+/ /#srchmsg                                    + / /
+/ /{title2                                     + / /
%S+SELECT    %<ENTER> + PROCESS SELECTION  %PFØ3 + PROCEED %PFØ4 +CANCEL
ALL
+? DATASET  +
```

```
)MODEL CLEAR(fun) ROWS(ALL)
£z#display1                                                                +
)INIT
.zvars ='(fun)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
&AMT = 'HALF'
&srchmsg = '&srchmsg1 &srchmsg2'
)REINIT
REFRESH(*)
)PROC
IF (.MSG NE '')
  &info = '######### ERROR #########'
)HELP
)END
```

CHKC1H PANEL

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  ]   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(7Ø,36) OUTLINE(BOX)
% /-/ Help Help Help Help Help Help Help /-/
+ This Panel is an input aid to produce JCL to run the routine
+%CHARCHK+, this routine compares the characters found in two datasets
+ and reports similarities and exceptions. After successful
+ selection of the required comparison source datasets simply
+ submit the generated JCL
+
+
% Enter corresponding number for further information
+
+
%/-/#Compare characters found in two datasets ] /-/
$zuser   +/ /$timestmp        +
+Command ===>_zcmd                                                  +
+/ /[info                   / /+
/ /]  CCC  H   H K   K {  CCC H   H   A   RRRR  / /
/ /] C     H   H K  K  { C    H   H  A A  R  R  / /
/ /]C      H   H K K   {C     H   H A   A R  R  / /
/ /]C      HHHHH KK    {C     HHHHH AAAAA RRR   / /
/ /]C      H   H K K   {C     H   H A   A R R   / /
/ /] C     H   H K  K  { C    H   H A   A R  R  / /
/ /]  CCC  H   H K   K {  CCC H   H A   A R   R / /
/ /]                                          / /
+
```

```
{ Datasets to Compare:
{  1. %(1)                                                          +
{  2.                                                               +
+
{ Write output to Dataset:
{   %(2)                                                         +
+
+
/ /%<ENTER>+Check Input    %PF3+Proceed    %PF4+Cancel/ /
+
)INIT
)PROC
&ZIND=YES
&ZSEL=TRANS(&ZCMD
           1,*CHKC1H1
           2,*CHKC1H2
           *,'?')
)END
```

CHKC1H1 PANEL

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  ]   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(70,22) OUTLINE(BOX)
% /-/ Help Help Help Help Help Help Help /-/
+
+ Enter the fully-qualified dataset name here
+
+ Alternatively enter a partially-qualified dataset name and
+ this will be used to produce a list where the required
+ dataset may be selected.
+ a %'+ at the start signifies that the directly following input
+ starts with the high-level qualifier.
+ If there is no %'+ then the user's prefix is inserted into the
+ selection criteria.
)INIT
)PROC
&ZHTOP=CHKC1H
&ZUP=CHKC1H
&ZCONT=CHKC1H2
)END
```

CHKC1H2 PANEL

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH) COLOR(WHITE)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  ]   TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
  {   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
  #   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
)BODY DEFAULT(%+_) EXPAND(//) WINDOW(70,22) OUTLINE(BOX)
% /-/ Help Help Help Help Help Help Help /-/
+
+ This is the name of the dataset where the results of the
+ comparison are written.
+ Enter a dataset name here when the default value should not
+ be used.
+ When the dataset exists it will be added to, when it does not
+ exist it will be created.
+ If no name is entered here it will be automatically entered with
+ the default value.
)INIT
)PROC
&ZHTOP=CHKC1H
&ZUP=CHKC1H1
&ZCONT=CHKC1H
)END
```

CHKCJCL SKELETON

```
//AL13745# JOB (SR00,SR016882,SR016882),'CHARCHCK',
//             CLASS=0,MSGCLASS=H,REGION=0M,
//             NOTIFY=AL13745,MSGLEVEL=(1,1)
/*JOBPARM L=0100,G=99999
//*  -------------------------------------------------------------------
//*
//*  Batch job to run the character checking routine CHARCHK
//*  This routine compares the characetrs found in two datasets
//*  and reports on where the characters have been found.
//*  The second parameter is a switch for the trace function of REXX.
//*  Default is 'o' for OFF.
//*
//*  -------------------------------------------------------------------
//REXX01   EXEC PGM=IKJEFT1A
//SYSEXEC  DD  DISP=SHR,DSN=AL13745.ISPF.EXEC
//CHKCRES  DD  DISP=MOD,DSN=&OUTDSN
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD *
 %CHARCHK O +
 &SRCDSN1 +
```

```
 &SRCDSN2
/*
```

Rolf Parker
Systems Programmer (Germany)

Punching policy statements directly from the CDS

Sometimes a utility does not need to be complex or tricky, it just needs to be written.

This is so often the case with newer technologies or products. What I am presenting with this article is one such opportunity. Many, if not most, of us are now running parallel sysplexes and have to work with and update the policies that are kept in the CDSs (Couple Data Sets). Notice that I didn't say they are *maintained* in the CDSs. They are kept in the CDSs, but we must maintain them elsewhere, because they cannot be manipulated the way a parmlib member, for instance, can be edited; they can only be replaced, added, or deleted.

The way I believe most shops maintain their policies is by keeping a card image deck of the statements used to create the existing policy in some agreed location. When a change is needed, the existing deck of statements is modified and then used to replace the existing policy, and the new deck is then saved back in the agreed location. If everyone follows these procedures, and nothing ever happens to the dataset where the statements are kept, this will work fine. If, however, someone makes a change and doesn't save the images back where they belong, or if the deck is accidentally modified, or even if the policy is updated from an entirely different set of statements, then you have an error waiting to happen. The error won't occur at the time the rules are broken; it will become apparent when the next person makes a change.

You can see therefore what the basic problem is. When you go to update the CFRM, or some other policy, you need to be absolutely certain that the copy of the statements used originally to create the existing policy is correct, and complete, but you really have no easy way of verifying that. We can print the policy, but it is generally too large to be easily or quickly compared with an existing deck of statements used to create a new policy. Moreover, when the policies are created from an original deck of statements, the policy sizes are rounded up, generally to the next multiple of four, making it nearly impossible to use some type of comparison utility to compare the listing and the existing card image definitions.

I did not try to write a utility to directly manipulate a policy in order to solve this problem, although that would be an interesting piece of code to write. What I did write is a very simple utility to 'punch' a deck of cards that can be used to exactly and completely replace a named policy. For those of you without a physical card punch, any 80-byte logical record length output dataset will do. This deck will be a good solid starting point from which changes to the existing policy can be reliably made. In line with the policy of keeping it as simple as possible, I decided to incorporate a two-step process. The first step is to use the existing, formally supported, administrative data utility to produce a listing of all of the policies of a given type, and have the listing written to a temporary dataset. The temporary dataset is then read back in as input to a second step. This is the code that is presented later in this article. This module is passed the policy name to be punched in the parm field. Then it is a straightforward task to handle the simple reformatting required to produce a deck of 'cards' suitable to replace an existing policy, and of course to stop when the policy being punched ends, either with the end of the input or the beginning of the next policy.

By using the administrative data utility we insulate ourselves from the potential of internal changes to the format of the policies that are kept in the CDS. I had hoped to provide myself with a large degree of commonality between input sources for

different types of policies, but I soon found formatting differences between the CFRM formatted output and the SFM formatted output. In particular, when searching through the CFRM output to locate the policy name we are interested in, we need to look for a literal followed by an opening parenthesis, and then the name left justified within an 8-byte blank padded field before finding a closing parenthesis. The SFM policy, on the other hand, has the same literal followed by a nicely edited opening parenthesis followed immediately by the name, and a closing parenthesis with no embedded blanks. The utility looks for names formatted in either manner, for all policy types. I handled it that way in the expectation that IBM will one day apply the same formatting rules to all of the reports. One other possibility is covered by using the administrative data utility as an initial source of data input: if a new policy type is created, it is likely that it will be administered with the existing utility. If a new policy type is created in the future it is reasonable to expect that when printed, or reported on, it will take the same basic format as the existing reports. That would either make this utility already capable of handling the new policy types, or it would, hopefully, require only minimum changes to make it capable of handling a newer policy type. We want to make our utilities as simple as possible, but we also want to keep an eye to the future and design programs that will need as little maintenance as possible.

As stated earlier, this is a two-step job, so I have included a copy of the JCL to run the job. Please notice that the first step is the standard IBM administrative data utility, without modification; it does, however, require that the input statement be updated to specify the appropriate policy type – CFRM, SFM, and so on. The output from the first step is written to a temporary dataset, to be read as input in the second step. The second step is the utility we have been describing. It does take an input parameter to specify the policy name to be punched as well as the input report of the correct data type produced in the first step.

In our shop we most often change the CFRM (Coupling

Facility Resource Management) policies, but the utility that I present here will also work for SFM (Sysplex Failure Management) and ARM (Automatic Restart Management) policies.

JCL

```
//TØSMØXZ JOB (4Ø3Ø,37),'ANY JOB DESCRIPTION',
//        CLASS=X,MSGCLASS=A,
//        NOTIFY=TØSMØ,REGION=5M
//*
//*   REFER TO -  TØSMØ.TSO.JCL(STCFRMØ1)
//*%PDSDOC ØØ PUNCH POLICY STATEMENTS FROM ACTIVE POLICY
//*
//STEPTEST EXEC  PGM=IXCMIAPU
//SYSPRINT DD    DSN=&&TEMP1,UNIT=339Ø,DISP=(NEW,PASS),
//         SPACE=(CYL,(5,5)),DCB=(LRECL=132,RECFM=FBA)
//SYSIN    DD    *
  DATA TYPE(CFRM) REPORT(YES)
//STEP1      EXEC  PGM=STCFRMØ1,PARM=TECPOL2   <<=== POLNAME IN PARM
//STEPLIB    DD    DSN=ANY.LOAD.LIB.FOR.THE.MODULE,DISP=SHR
//SYSUDUMP   DD    SYSOUT=*
//SYSIN      DD    DSN=&&TEMP1,DISP=SHR
//SYSPUNCH   DD    DSN=SYS1.SYSPLEX.CNTRL(PUNCHED),DISP=SHR
```

STCFRM01

Here is the source for this simple utility. Some would argue that this could have been better done with REXX, but I simply prefer Assembler, even with a task as simple as this.

```
STCFRMØ1 CSECT
** * ------------------------------------------------------------ * **
** * -- STCFRMØ1 -                                               -- * **
** * --  PURPOSE PUNCH A POLICY FROM THE ACTIVE CDS DATA SET     -- * **
** * --                                                          -- * **
** * -- CHECK PARM FOR POLICY NAME TO GET  IE                    -- * **
** * --    PARM=STATPOL1                                         -- * **
** * --                                                          -- * **
** * -- REOPEN THE SYSPRINT DATASET, AND LOOK FOR               -- * **
** * -- DEFINE POLICY NAME(ABCDEFGH)  -- * **                   -- * **
** * -- FORMAT THE REST AS OUTPUT STATEMENTS UNTIL THE NEXT     -- * **
** * -- DEFINE POLICY NAME STATEMENT IS FOUND OR EOF OF COURSE  -- * **
** * -- REFORMAT AS NEEDED AND WRITE TO SYSPUNCH                -- * **
** * -- STRIPPING OUT.................                          -- * **
** * --        LINE # 1-6
```

```
** * --       1ADMINISTRATIVE DATA Ø-2Ø                              -- * **
** * --       +__________ Ø-11                                       -- * **
** * --       BLANKS 1-4Ø                                            -- * **
** * --                                                              -- * **
** * ---------------------------------------------------------------- * **
STCFRMØ1 AMODE 24
STCFRMØ1 RMODE 24
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         USING STCFRMØ1,15              TEMP ADDRESSABILITY
         B     @PROLOG                  AROUND THE EYECATCHER
         DC    C'STCFRMØ1 - PUNCH CFRM DEFINITIONS - &SYSDATE '
         DS    ØH
@PROLOG  BAKR  R14,RØ                   PUSH ENVIRONMENT ON STACK
         LA    R5,Ø(RØ,R1)              GET POINTER TO PARM
         LR    R12,R15
         DROP  R15
         USING STCFRMØ1,R12             R12 IS NOW BASE
         L     R2,DYNSIZE               * LENGTH TO GET
          STORAGE OBTAIN,ADDR=(1),SP=Ø,LENGTH=(2)
         LTR   R15,R15                  TEST RETURN CODE
         BNZ   EARLYOUT                  - IF WE COULDN'T GET ANY- END.
         LR    R13,R1                   GET ADDR OF AREA
         USING DYNAREA,R13              SET ADDRESSABILITY FOR SAVEAREA
         L     R5,Ø(R5)                 POINT TO ACTUAL PARM AREA
         LH    R6,Ø(R5)                 GET PARM LENGTH
         BCTR  R6,RØ                    REDUCE FOR EXECUTE TO COME...
         LTR   R6,R6                    CHECK FOR ZERO
         BZ    NOPARM
         CH    R6,=H'7'                 CHECK FOR GREATER THAN MAX.
         BNH   OKASIS                   CONTINUE IF 8 OR LESS - ELSE..
         LH    R6,=H'7'                 FORCE A MAX OF 8 BYTES
OKASIS   MVI   PARM,X'4Ø'               CLEAR THE RECEIVING FIELD
         MVC   PARM+1(L'PARM-1),PARM
         EX    R6,MOVEPRM               THEN FILL IT WITH THE PARM
         MVC   PARM2,PARM               MOVE A COPY TO PARM2
```

```
         LA    R6,1               INDEX VALUE TO USE
         LA    R7,PARM2+7         END OF SEARCH AREA
         LA    R4,PARM2           START OF SEARCH AREA
LOOKBLNK CLI   Ø(R4),C' '         LOOK FOR A BLANK
         BE    SETPAREN
         BXLE  R4,R6,LOOKBLNK
         B     SETLEN             FALL THROUGH IF A FULL 8 CHARS USED
SETPAREN MVI   Ø(R4),C')'
         LA    R4,1(R4)
SETLEN   LA    R6,PARM2           GET START OF ADDR
         SR    R4,R6              GET
         BCTR  R4,RØ              R4 IS SET FOR AN EX TO CHECK PARM2
PARMSSET EQU   *
** * ------------------------------------------------------------ * **
** * -- OPEN INPUT AND OUTPUT FILES                              -- * **
** * ------------------------------------------------------------ * **
         OPEN  (DDIN,(INPUT))
         LA    R8,DDIN
         USING IHADCB,R8
         TM    DCBOFLGS,DCBOFOPN  CHECK THE OPEN FLAGS
         BO    OPNISOK1
         WTO   'STCFRMØØ1-E-ERROR OPENING DD SYSIN FILE'
         B     RET12Ø                  AND GO HOME WITH A BAD RC.
OPNISOK1 OPEN  (DDOUT,(OUTPUT))
         LA    R8,DDIN
         USING IHADCB,R8
         TM    DCBOFLGS,DCBOFOPN  CHECK THE OPEN FLAGS
         BO    LOOPØ1
         WTO   'STCFRMØØ1-E-ERROR OPENING DD SYSPUNCH'
         B     RET121                 AND GO HOME WITH A BAD RC.
LOOPØ1   GET   DDIN,BUFIN              GET AN INPUT RECORD
* SKIP THROUGH UNTIL "DEFINPOLICY NAME(" AND OUR PARM IS FOUND
         CLI   BUFIN+1,C'D'             QUICK CHECK
         BNE   TSTSFM
         CLC   BUFIN+2(18),=C'EFINE POLICY NAME('
         BE    CHKCFRM                  LOOK FOR FULL LITERAL
         B     LOOPØ1
TSTSFM   CLI   BUFIN+3,C'D'             QUICK CHECK
         BNE   LOOPØ1
         CLC   BUFIN+4(18),=C'EFINE POLICY NAME('
         BNE   LOOPØ1                   LOOK FOR FULL LITERAL
* WE HAVE A DEFINITION POLICY STATEMENT, IS IT OURS?
         EX    R4,CLCPARM2
*        CLC   BUFIN+22(R4),PARM2 TEST AGAINST PARM TYPE 2
         BE    MATCHED
CHKCFRM  CLC   BUFIN+2Ø(8),PARM   TEST AGAINST PARM TYPE 1
         BNE   LOOPØ1
MATCHED  MVC   BUFIN+33(12),=C'REPLACE(YES)'
* WE FOUND A GOOD RECORD, NOW WE CAN START PUNCHING CARDS !!!
         B     PUTREC1
```

```
LOOPØ2   GET   DDIN,BUFIN
         CLI   BUFIN+1,C'D'             QUICK CHECK FOR POLICY BREAK
         BNE   TSTOK1
         CLC   BUFIN+2(18),=C'EFINE POLICY NAME('   IF FOUND THEN
         BE    EOFIN1                                  WE ARE DONE!
TSTOK1   CLC   BUFIN(4),=X'4Ø4Ø4Ø4Ø'    STRIP HEADERS ETC.
         BNE   LOOPØ2
         CLI   BUFIN+1,C' '
         BNE   PUTREC1
         CLC   BUFIN+2(8Ø),BUFIN+1      A BLANK LINE ?
         BE    LOOPØ2
PUTREC1  PUT   DDOUT,BUFIN
         B     LOOPØ2
EOFIN1   EQU   *   END OF THE INPUT FILE, OR WE JUST DECIDED TO END.
         CLOSE (DDIN,,DDOUT)
         B     RETURNØØ
MOVEPRM  MVC   PARM(Ø),2(R5)
CLCPARM2 CLC   BUFIN+22(Ø),PARM2  TEST AGAINST PARM TYPE 2
NOPARM   EQU   *                        NO PARM PASSED - DIE NOW.
         SR    RØ,RØ
         WTO   'STCFRMØØ1 -E- NO PARM PRESENT, PARM MUST = CFRM POLICY X
               NAME TO BE PUNCHED.'
         B     RET12Ø
RETURNØØ EQU   *
         BAL   R11,RELSTG               RELEASE ACQUIRED STORAGE
EARLYOUT SR    R15,R15                  SET RC=ØØ
         PR                             AND GO ON HOME.
RET121   EQU   *
         CLOSE (DDIN)
RET12Ø   EQU   *                        NOTHING TO CLOSE ...JUST END
         BAL   R11,RELSTG               RELEASE ACQUIRED STORAGE
         L     R15,=F'16'               SET RC=16
         PR                             AND GO ON HOME.
RELSTG   L     R2,DYNSIZE               * LENGTH TO RELEASE
         LR    R3,R13
         STORAGE RELEASE,ADDR=(3),LENGTH=(2)    RELEASE STORAGE
         B     Ø(R11)
** * ------------------------------------------------------------ * **
** * -- STATIC STORAGE AREA HERE - LTORG - MODELS ETC.          -- * **
** * ------------------------------------------------------------ * **
DYNSIZE  DC    AL4(@DYNSIZE)            DYNAM AREA SIZE
TRHEX    DC    C'Ø123456789ABCDEF'
         LTORG
DDIN     DCB   DSORG=PS,RECFM=FB,MACRF=(GM),DDNAME=SYSIN,              X
               EODAD=EOFIN1
DDOUT    DCB   DSORG=PS,RECFM=FB,MACRF=(PM),DDNAME=SYSPUNCH,           X
               LRECL=8Ø
** * ------------------------------------------------------------ * **
** * -- DYNAMIC STORAGE AREA - REGISTER SAVE, PARM SAVE ETC.    -- * **
```

```
** * ------------------------------------------------------- * **
DYNAREA  DSECT
SAVEAREA DS    18F                        STANDARD SAVE AREA
PARM     DS    CL8                        THE PASSED CFRM NAME
PARM2    DS    CL8                        THE PASSED CFRM NAME
BUFIN    DS    CL133                      BUFFER AREA FOR INPUT RECORDS
BUFOUT   DS    CL8Ø                       MY OUTPUT AREA
@ENDDYN  DS    ØX
@DYNSIZE EQU   ((@ENDDYN-DYNAREA+7)/8)*8  AREA SIZE DBL WORD MULTIPLES
         DCBD
         END
```

That's all there is to it. Nothing fancy, nothing special, just basic straightforward reading, comparing, some minor reformatting, and writing – not very challenging at all, but, as we said earlier, sometimes all you need is a really simple program. If you need to make a quick change to a CFRM policy and you cannot find the source that was used originally to create it, or even worse, if you find it but are pretty sure it has been changed since it was originally used, well then, this is just the thing to get you going again. I use the utility before every change I make to the policy, even if I was the person to make it.

I hope this will be of value to you, and that it will spur you on to write more of your own utilities. Remember – they don't have to be jewels of modern programming, they just have to be useful to you in your environment.

Stephen McColley
Senior Systems Programmer
SunTrust Bank (USA)

Utility to free unused space from datasets

In the shops where I have worked, I have seen guys from the Application Support Group doing space management manually. Though these datasets are SMS-managed, SMS space management runs only at specific times in a day.

During the days when heavy load is expected in the overnight batch, space management is carried out manually to make sure a certain amount of free space is available to avoid job failures. The usual approach is to list the datasets from 3.4 and sort them based on the percent usage. All the datasets with an unused percentage greater than, say, 30% are freed. This exercise was time consuming and boring. I have developed a small job to assist. It is based on the NaviQuest ISMF batch utility, which comes with DFSMS.

Before using the job you need to identify the dataset containing all the NaviQuest CLISTs – usually SYS1.DGTCLIB, if not renamed at you shop. Then modify the job accordingly. Follow the instructions given in the job carefully. More documentation on the parameters to be specified for the dataset filter criteria can be obtained from the NaviQuest user guide.

NaviQuest is available from Option 11 of ISMF. However, not many application people will have noticed this because the option is not available in user mode, which is commonly provided to them. There are many other very good features provided by NaviQuest that can be explored.

A CLIST called MYFREE was provided. This CLIST frees the unused space from the dataset. This is similar to the FREE command from the DSLIST menu.

FREE_JCL

```
//DSLISTØ1 JOB (XYZ),'DSLIST',
//         CLASS=D,
//         NOTIFY=&SYSUID,
//         MSGCLASS=X
//*******************************************************************
//         SET PREFIX=USERID   /* Set your prefix *
//         SET PREFIX1=SYS1    /* Set TLIB, MLIB etc Prefix */
//*******************************************************************
//*   Instructions Before Submitting:                               *
//*   -> Change JOBCARD to suite your installation requirements. (#1)*
//*   -> Change all XXXXXXX to your userid (#55, #56, #1Ø2)         *
//*   -> Change PLIB, MLIB, SLIB, TLIB, and SYSPROC to your         *
//*      installation-specific datasets.                            *
//*   -> Change LSTNAME to a more meaningful name to you.(#57 & #1Ø3)*
```

```
//*   -> Change HLQ.** to dataset mask of your choice. (#60)         *
//*   -> Change 'volser' to the Volume name from which datasets      *
//*      need to be freed. This can be generic name like A*.(#61)    *
//*   -> You can generate the list from CATALOG. In this case        *
//*      Change SOURCENL(1) to SOURCENL(2) and VTOCVSER parameter    *
//*      is not needed. (#58)                                        *
//*********************************************************************
//*********************************************************************
//*  SET the type of execution FREE/LIST. In FREE mode the           *
//*  datasets will be freed and in LIST mode the datasets eligible   *
//*  for FREEing will be reported. This REPORT can be verified       *
//*  and can be input for the STEP FREE in this JCL.                 *
//*                                                                  *
//*  CC = 0  FREE Mode                                               *
//*  CC = 4  LIST Mode                                               *
//*********************************************************************
//EXECTYPE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD *
   SET LASTCC = 4                  /* List Mode */
/*
//*********************************************************************
//* Generate Dataset List.                                           *
//*********************************************************************
//SAVELIST EXEC PGM=IKJEFT01,DYNAMNBR=50
//ISPPLIB  DD DISP=SHR,DSN=&PREFIX1..PLIB
//ISPMLIB  DD DISP=SHR,DSN=&PREFIX1..MLIB
//ISPSLIB  DD DISP=SHR,DSN=&PREFIX1..SLIB
//ISPTLIB  DD DISP=SHR,DSN=&PREFIX..TLIB
//         DD DISP=SHR,DSN=&PREFIX1..TLIB
//SYSPROC  DD DISP=SHR,DSN=&PREFIX1..CLIB
//ISPTABL  DD DISP=SHR,DSN=&PREFIX..TLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//ISPLOG   DD SYSOUT=*,DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//ISPPROF  DD DSN=&&PROF,DISP=(NEW,DELETE,DELETE),
//         DCB=(&PREFIX1..TLIB),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSTSIN  DD *
DELETE 'XXXXXXX.DATASET.REPORT'
PROFILE PREFIX(XXXXXXX)
ISPSTART CMD(ACBQBAI2 SAVE LSTNAME +
SOURCENL(1) +
VTOCDATA(Y) +
DSN('HLQ.**') +
VTOCVSER(volser) +
HSMDATA(N) +
NOTUSED%(GT 40)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
```

```
/*
//*******************************************************************
//*                                                                 *
//*  Allocate ISPFILE, where the generated report is saved.         *
//*  NOTE: The dataset being allocated should not be a temporary    *
//*        dataset.                                                 *
//*                                                                 *
//*******************************************************************
//ALCISPFL EXEC PGM=IEFBR14
//ISPFILE DD  DSN=&PREFIX..DATASET.REPORT,DISP=(NEW,CATLG),
//    BLKSIZE=Ø,SPACE=(CYL,(2Ø,1Ø)),RECFM=FBA,LRECL=133,UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//*******************************************************************
//*                                                                 *
//* Dataset report generation step.                                 *
//*                                                                 *
//*    PARAMETER FOLLOWING ACBQBAR1 - ISMF SAVED LIST NAME (INPUT)  *
//*    ISPFILE - DATA SET REPORT (OUTPUT, FROM ALCISPFL STEP)       *
//*    SYSIN   - KEY WORDS TO SPECIFY THE DATA IN THE REPORT        *
//*                                                                 *
//*******************************************************************
//GENREP   EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø
//ISPPLIB  DD DISP=SHR,DSN=&PREFIX1..PLIB
//ISPMLIB  DD DISP=SHR,DSN=&PREFIX1..MLIB
//ISPSLIB  DD DISP=SHR,DSN=&PREFIX1..SLIB
//ISPTLIB  DD DISP=SHR,DSN=&PREFIX..TLIB
//         DD DISP=SHR,DSN=&PREFIX1..TLIB
//SYSPROC  DD DISP=SHR,DSN=&PREFIX1..CLIB
//ISPTABL  DD DISP=SHR,DSN=&PREFIX..TLIB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//ISPLOG   DD SYSOUT=*,DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//ISPPROF  DD DSN=&&PROF,DISP=(NEW,DELETE,DELETE),
//         DCB=(&PREFIX1..TLIB),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//ISPFILE DD  DSN=&PREFIX..DATASET.REPORT,DISP=OLD
//SYSTSIN  DD *
PROFILE PREFIX(XXXXXXX)
ISPSTART CMD(ACBQBAR1 LSTNAME) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
/*
//SYSIN DD  *
DSN
VOLSER
ALLOCSP
ALLOCUSED
%NOTUSED
CREATE
/*
```

```
//*******************************************************************
//*                                                                 *
//* Generate the INPUT for FREE.                                    *
//*                                                                 *
//*******************************************************************
//SORT   EXEC PGM=SORT
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SORTIN   DD DISP=SHR,DSN=&PREFIX..DATASET.REPORT
//SORTOUT  DD DSN=&&TEMP,DISP=(,PASS),
//            SPACE=(CYL,(1,1)),DCB=*.SORTIN
//SORTWK1  DD UNIT=SYSDA,SPACE=(CYL,(2Ø,5))
//SORTWK2  DD UNIT=SYSDA,SPACE=(CYL,(2Ø,5))
//SORTWK3  DD UNIT=SYSDA,SPACE=(CYL,(2Ø,5))
//SORTWK4  DD UNIT=SYSDA,SPACE=(CYL,(2Ø,5))
//SORTWK5  DD UNIT=SYSDA,SPACE=(CYL,(2Ø,5))
//SYSIN    DD *
 OMIT COND=((1,1,CH,EQ,C'1'),OR,(1,5,CH,EQ,C'     '),OR,
            (3,11,CH,EQ,C'DATASETNAME'),OR,
            (3,11,CH,EQ,C'----------'),OR,
            (3,11,CH,EQ,C'**********'),OR,
            (15,1,CH,EQ,C'-'),OR,
            (16,11,CH,EQ,C'L E G E N D'))
 SORT FIELDS=(3,44,CH,A)
 OUTREC FIELDS=(C' MYFREE ',9:3,44)
/*
//    IF EXECTYPE.RC = Ø THEN
//*******************************************************************
//*  Free the datasets                                              *
//*  Make sure the REXX EXEC MYFREE is available in the dataset     *
//*  specified in SYSPROC                                           *
//*******************************************************************
//FREE   EXEC PGM=IKJEFTØ1
//SYSPROC  DD DISP=SHR,DSN=&PREFIX..CLIST
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  PROFILE NOPREFIX
//         DD DSN=&&TEMP,DISP=(OLD,DELETE)
//*******************************************************************
//    ELSE
//*******************************************************************
//* Copy the dataset report                                         *
//*******************************************************************
//REPGEN EXEC PGM=IEBGENER
//SYSUT1   DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSPRINT DD  SYSOUT=*
```

```
//*******************************************************************
//    ENDIF
//*******************************************************************
```

MYFREE_REXX

```
/**Rexx************************************************************/
/*      EXEC to FREE unused space in a dataset                     */
/*******************************************************************/
ARG dsname .
IF SYSDSN("'"dsname"'") = 'OK'
Then Do
  "ALLOC f(infile) ds("dsname") mod release"
  "EXECIO Ø diskw infile (open finis"
  "Free f(infile"
  End
Else Nop
RETURN
```

SRG
Systems Programmer (UK)

Cole Software has announced Release z1.6 of z/XDC, its Assembler development tool.

Major changes in this release include full support for the high-level Assembler's restructured ADATA. It can use the information contained in ADATA files to produce source-level maps of programs and control blocks located in storage. There's also full support for z/OS 1.6's new ALRF (Address Space Number, ASN, and Linkage Index, LX, Reuse Facility) support. With ALRF, z/OS can permit the reuse of Address Space Numbers (ASNs) in circumstances where previously they could not be reused.

For further information contact:
Cole Software, 736 Fox Hollow Road, Afton, VA 22920, USA.
Tel: (540) 456 8210.
URL: www.colesoft.com/pr041129.html.

* * *

BMC Software has announced the availability of its Topology Discovery, the third component of the company's business-aware discovery and detection solutions in its Change and Configuration Management (CCM) offering.

Topology Discovery provides IT managers with visibility into complex dependencies between applications and the underlying infrastructure to optimize business availability and speed up problem resolution processes.

Topology Discovery is an agent-less solution that uses native communications protocols to discover the elements within enterprise application infrastructures, along with their logical and physical dependencies.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042, USA.
Tel: (713) 918 8800.
URL: www.bmc.com/corporate/nr2004/120704_1.html.

* * *

Computer Associates has announced the general availability of three Unicenter performance management products. They are the Unicenter NetMaster r11 suite of products, Unicenter CA-OPS/MVS Event Management and Automation r11, and Unicenter CA-SYSVIEW Realtime Performance Management r11.

Unicenter NetMaster r11 is a management suite for mainframe-connected TCP/IP and SNA networks, which provides Web browser access to real-time and historical information.

Unicenter CA-OPS/MVS Event Management and Automation r11, is an automated systems operation solution for z/OS environments, which helps streamline business processes, increase productivity, and reduce operating costs by helping to ensure optimum system availability.

Unicenter CA-SYSVIEW Realtime Performance Management r11 enables mainframe system programmers to determine the cause of degraded z/OS system performance by reviewing short-term historical event data.

For further information contact:
Computer Associates International, One Computer Associates Plaza, Islandia, NY 11749, USA.
Tel: (631) 342 6000.
URL: www3.ca.com/press/pressrelease.asp?CID=65139.

* * *