228

MVS update

September 2005

In this issue

A xephon PUBLICATION

MVS Update

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.

DFHSM problem determination aid facility

As with many IBM products, the problem determination aid facility can gather diagnostic information about DFSMShsm's processing as it executes. This information is stored in a circular log file within storage, and at specific intervals it is transferred from storage to circular DASD files. If you look in the HSM start-up deck, there are two such DASD files that are coded on the ARCPDOX and ARCPDOY JCL DD statements. An example HSM started procedure is shown below:

```
//HSM        PROC CMD=ØØ,
//           EMERG=NO,
//           LOGSW=NO,
//           STARTUP=YES,
//           UID=HSM7881,
//           SIZE=ØM,
//           DDD=2Ø,
//           HOST=,
//           PRIM=,
//           PDA=YES,
//           CDSR=NO,
//           CDSQ=YES,
//           SYSID=CMØ1
//HSM         EXEC PGM=ARCCTL,DYNAMNBR=&DDD,REGION=&SIZE,TIME=144Ø,
//            PARM=('EMERG=&EMERG','LOGSW=&LOGSW','CMD=&CMD','UID=&UID',
//            'STARTUP=&STARTUP','HOST=Ø','PDA=&PDA','CDSR=&CDSR',
//           'CDSQ=&CDSQ')
//HSMPARM      DD    DSN=HSM.&SYSID..PARMLIB,DISP=SHR
//MSYSOUT      DD    SYSOUT=*
//MSYSIN       DD    DUMMY
//SYSPRINT     DD    SYSOUT=*,FREE=CLOSE
//SYSUDUMP     DD    SYSOUT=*
//MIGCAT       DD    DSN=HSM.PROD.MCDS,DISP=SHR
//JOURNAL      DD    DSN=HSM.PROD.JRNL,DISP=SHR
//ARCLOGX      DD    DSN=HSM.PROD.HSMLOGX&HOST.,DISP=OLD
//ARCLOGY      DD    DSN=HSM.PROD.HSMLOGY&HOST.,DISP=OLD
//BAKCAT       DD    DSN=HSM.PROD.BCDS,DISP=SHR
//OFFCAT       DD    DSN=HSM.PROD.OCDS,DISP=SHR
//ARCPDOX      DD    DSN=HSM.PROD.HSMPDOX&HOST.,DISP=OLD
//ARCPDOY      DD    DSN=HSM.PROD.HSMPDOY&HOST.,DISP=OLD
```

The characteristics of the ARCPDOX and ARCPDOY files are shown below:

- Organization – PS.
- Record format – VB.
- Record length – 1024.
- Block size – 27652.

When everything works, these files are of little use – but there will be occasions when you will want to collect and save the data written to these files.

For example you can use the information to show a trace of the HSM region's operating history – so if a problem occurs you can see what was happening at a specific time. Points of contention between subtasks can be identified. Traces can be used to confirm that a problem exists, or to clarify that it does not.

To review these traces there are two specific tools you can utilize. The first is a standard ISPF dataset browse. This allows you to view raw data and can be helpful, but also cumbersome. A more useful way to analyse data is by using the ARCPRPDO utility. This utility is a formatting program and it has a number of control commands that allow you to not only copy data but also to manipulate it.

The main uses of the formatter are to:

- Edit raw trace output.
- Translate trace data into a more readable format
- Select records based on selection criteria.

You can actually manipulate the whole tracing option by issuing commands to turn it on and off. As you can see from the code below, we code PDA=YES to turn it on at start-up, but it can be switched off by issuing a SETSYS PDA(OFF) command and turned back on by issuing a SETSYS PDA(ON) command.

The ARCPRPDO has a number of commands that allow you to format output. These are shown below:

- COPY – copies trace data from the input dataset to the dataset you specify on the //ARCOUT DD statement.
- COMPACT – formats the trace entries into single-line output where possible. The program uses this option by default, so it does not need to be coded.
- FORMAT – if coded, each trace keyword will be output on a separate line.
- NOPRINT – used in conjunction with the COPY option, it stops the trace data being output to the //ARCPRINT DD statement. The latter does still need to be coded, however, to allow the program to output processing messages.

In addition to the primary formatting commands there are a number of sub-selection options that can be coded, allowing you to filter certain records even further.

The ARCPRPDO selection options are shown below:

- START(date,time) – selects records from a specific date and time.
- END(date,time) – selects records up to a specific date and time.
- TCB(address) – selects records with a TCB address that matches the given address. More than one address can be specified and these must be separated by a comma.
- MODULE(module) – module selects all trace records associated with a module you have coded. Ten modules can be specified separated by commas. Module names must be coded without the prefix of ARC. So to select the ARCSELTV module you would code MODULE(SELTV).
- LOGIC(type) – selects records containing specified logic type. Ten such logic types separated by commas can be coded. Logic types are seen in the LOGIC column of the trace output.
- SCAN(data) – scans for particular types of data.

- RECYCLE – selects records that pertain to RECYCLE processing only.

Below is the JCL to run the formatter to COPY records to an external dataset:

```
//JXB7884    JOB   (7884),'J.BRADLEY',CLASS=A
//*      ************************************************************
//*      *                                                          *
//*      * PDA DUMP FORMATTER FOR HSM.                             *
//*      *                                                          *
//*      ************************************************************
//STEP1       EXEC  PGM=ARCPRPDO
//SYSPRINT    DD    SYSOUT=*
//ARCMSG      DD    SYSOUT=*
//ARCPDO      DD    DSN=HSM.PROD.HSMPDOX1,DISP=SHR
//ARCOUT      DD    DSN=JXB7884.HSM.COPYOUT,DISP=SHR
//ARCPRINT    DD    SYSOUT=*
//SYSIN       DD    *
  COPY
  NOPRINT
  START(94330,00001)
  END(94330,235959)
/*
```

The code below shows how you can use various options to extract information from this dataset:

```
//JXB7884    JOB   (7884),'J.BRADLEY',CLASS=A
//*      ************************************************************
//*      *                                                          *
//*      * DISPLAY PDA TRACE ENTRIES FOR MESSAGES.                  *
//*      *                                                          *
//*      ************************************************************
//STEP1       EXEC  PGM=ARCPRPDO
//SYSPRINT    DD    SYSOUT=*
//ARCMSG      DD    SYSOUT=*
//ARCOUT      DD    DUMMY
//ARCPDO      DD    DSN=JXB7884.HSM.COPYOUT,DISP=SHR
//ARCPRINT    DD    SYSOUT=*
//SYSIN       DD    *
  COMPACT
  LOGIC(MESG)
/*
```

The formatter is a very simple program, but it can be invaluable in diagnosing issues. IBM does recommend the following for usage:

1 Use Compact Module(aaaaa) for ARC0200I and ARC0208I issues, abends, etc.

2 Use Compact Scan(dsn) with a dataset name for errors related to a specific dataset.

3 Use Compact Logic(MESG) to obtain messages issued at the time of an error.

4 Use Format TCB(aaaaaa) to look at task-specific errors.

5 Use Format Start(date,time) and Format End(date,time) to restrict the search to specific times of day or to times when problems were occurring.

Full details of the ARCPRPDO program can be found in the DFSMS manual *DFSMS/MVS DFSMShsm Diagnosis Guide* LY27-9607.

John Bradley
Systems Programmer
Meerkat Computer Services (UK)

How to improve the LOGON proc

In many z/OS installations, maintenance of the logon/TSO procedures (LOGON proc) is often bothersome and lacking in good standards. Generally one should foresee certain categories of user (operators, sysprogs, programmers, etc) and likewise have just as many procedures. These procedures 'grow' with subsequent changes and often they differ substantially from one z/OS partition to the next.

This scene is further complicated when, while installing a new OS, new libraries either are substituted or get introduced.

Moreover, out of (bad) habit, I have always used procedures which have statically allocated files (card JCL: //DDNAME …) excluding more modular techniques.

I have come across the following limitations in particular:

- A test of the entire LOGON proc is required with each update to the JCL.
- In order to execute a 'move, compress, expand' of a dataset, it is necessary for all users connected to TSO to log off.
- The number of datasets allocated during a session is much greater than actually needed; the waste of resources is evident (large w-set) in the same way as performance gets hit (high connect time).
- The JCL coding of the LOGON proc is more 'delicate' – the possibility of error is directly proportional to the number of datasets referenced in the JCL.

For these reasons I decided to completely modify the design of my TSO/ISPF environment.

The first thing I did was to eliminate from the LOGON proc all those ISPF application files that don't belong to the z/OS Server Pack.

Next I created a short REXX (driver) code for each single application in which I execute all the necessary actions using the commands ISPEXEC LIBDEF DDname, ISPEXEC SELECT CMD/PGM, and ALTLIB.

Each specific driver/REXX will be executed only when the user activates that function in its particular ISPF panel.

Moreover, following the suggestions contained in *ISPF/Service Guide z/OS*, *Description of ISPF Service*, *User link libraries*, I have reduced the dimensions of the STEPLIB of the LOGON proc.

It is often possible to eliminate a library from a STEPLIB/ LOGON proc by defining some user_loadlib with the statement:

```
ISPEXEC LIBDEF ISPLLIB ID(user_loadlib) STACK
```

and by utilizing the following instruction for modules and commands:

```
ISPEXEC SELECT CMD(myprog)  NEWAPPL(anyname)  PASSLIB
```

It should be noted that many OEMs do not foresee this 'dynamic' coding in an ISPF environment.

In fact, their CLIST/REXX for the start-up of their products' primary functions usually do not contain the keywords PASSLIB and NEWAPPL for each ISPEXEC SELECT. The use of these is principally to guarantee the correct communication between the various ISPF panels – it's a simple addition that doesn't cause any kind of problem.

With this work I have 'only' decreased the number of JCL/DD cards in the LOGON proc by about 40%. But what can I do for the rest of the procedure?

My LOGON procs are always numerous and too diverse in relation to the type of users. To this end I have written a short CLIST code to have a single 'flexible' procedure for all users.

Here is TSOPROC, the LOGON/TSO procedure:

```
//$TSLPGEN PROC
//** ---------------------------- **
//** SAMPLE  LOGON PROCEDURE      **
//** ---------------------------- **
//$TSLPGEN EXEC PGM=IKJEFT01,DYNAMNBR=256,
//   PARM='%TSOCLST Y'                                    <- UPDATE
//SYSPROC  DD DISP=SHR,DSN=your.user.pds.clist            <- UPDATE
//SYSHELP  DD DISP=SHR,DSN=SYS1.HELP
//         DD DISP=SHR,DSN=ISF.SISFHELP
//         DD DISP=SHR,DSN=SYS1.HELPENP
//         DD DISP=SHR,DSN=ISP.SISPHELP
//SYSLBC   DD DISP=SHR,DSN=SYS1.BRODCAST
//SYSPRINT DD TERM=TS,SYSOUT=*
//SYSTERM  DD TERM=TS,SYSOUT=*
//SYSIN    DD TERM=TS
//** ---------------------------- **
//FWORKINP DD DISP=SHR,DSN=your.user.pds.clist(TSOXALC)  <- UPDATE
//FWORKOUT DD DISP=(,PASS),UNIT=VIO,SPACE=(TRK,(1,1,1)),//
DSN=&TEMP(ALCFLY),DCB=(LRECL=80,RECFM=FB,DSORG=PO)
```

where in the JCL card *PARM=%TSOCLST Y*, the *Y* character identifies a specific category of user.

Notice that once you have coded the table TSOXALC, the only thing that will differentiate one LOGON proc from another is the alphanumeric byte showing the user's category.

Here is the source for the CLIST TSOCLST:

```
PROC 1 TSOGRP                                  /* TSOCLST         */
   CONTROL NOFLUSH NOMSG MAIN                  /* 2ØØ5            */
PROFILE    MODE WTPMSG MSGID                   /*                 */
/* ------------------------------------------ /* -------------- */
SET RANGEOK = &STR(YNWKLEQXJ)                  /*                 */
IF &SYSINDEX(&TSOGRP,&RANGEOK) = Ø             /*                 */+
 THEN DO                                       /*                 */
  CONTROL FLUSH MSG LIST SYMLIST CONLIST       /*                 */
   WRITE * MSG > ERR: TSO GROUP IS NOT CORRECT /*                 */
    EXIT CODE(2Ø)                              /* ERROR/EXIT      */
     END                                       /*                 */
/* ------------------------------------------ /* -------------- */
OPENFILE FWORKINP INPUT                        /* OPEN WORK FILE  */
OPENFILE FWORKOUT OUTPUT                       /* OPEN WORK FILE  */
SET I = Ø                                      /* COUNTER/INPUT   */
SET O = Ø                                      /* COUNTER/OUTPUT  */
                                               /*                 */
LOOPØØ:                                        /*                 */+
DO UNTIL &EOF NE Ø                             /*                 */
 ERROR DO                                      /* EOF ROUTINE     */
  SET EOF = &LASTCC                            /*                 */
   GOTO LOOP99                                 /*                 */
    END                                        /*                 */
     GETFILE  FWORKINP                         /* READ TSOXALC    */
     SET EOF = &LASTCC                         /*                 */
     SET I = &I + 1                            /*                 */
     SET SCAN  = &SUBSTR(59:68,&FWORKINP)      /*                 */
     SET GRPOK = &SYSINDEX(&TSOGRP,&SCAN)      /*                 */
 IF &SYSINDEX(&SCAN,&STR(ALL)) NE Ø            /* SEARCH 'ALL'    */+
  THEN SET GRPOK = 1                           /* ALWAYS VALID    */
 IF &GRPOK EQ Ø                                /* SEARCH TSOGRP   */+
  THEN GOTO LOOP99                             /*                 */
   SET FWORKOUT = &FWORKINP                    /* IF VALID GROUP  */
    PUTFILE FWORKOUT                           /* WRITE RECORD    */
     SET O = &O + 1                            /*                 */
                                               /*                 */
LOOP99: END                                    /*                 */
/* ------------------------------------------ /* -------------- */
ERROR OFF                                      /*                 */
CLOSFILE FWORKINP                              /* CLOSE WORK FILE */
CLOSFILE FWORKOUT                              /* CLOSE WORK FILE */
IF &I EQ Ø                                     /*                 */+
 THEN DO                                       /*                 */
```

```
  CONTROL FLUSH MSG LIST SYMLIST CONLIST         /*                */
   WRITE * MSG > ERR: FILE FWORKINP EMPTY        /*                */
    EXIT CODE(21)                                /* ERROR/EXIT     */
     END                                         /*                */
IF &O EQ Ø                                       /*                */+
 THEN DO                                         /*                */
  CONTROL FLUSH MSG LIST SYMLIST CONLIST         /*                */
   WRITE * MSG > ERR: FILE FWORKOUT EMPTY        /*                */
    EXIT CODE(22)                                /* ERROR/EXIT     */
     END                                         /*                */
/* ------------------------------------------- /* -------------- */
ALTLIB ACT APPLICATION(CLIST) DDNAME(FWORKOUT)   /* EXEC CMD       */
      %ALCFLY                                    /* DYNAMIC        */
ALTLIB DEA ALL                                   /* ALLOCATION     */
EXIT                                             /* OK/EXIT        */
/* ------------------------------------------- /* -------------- */
```

This code selects the cards from source TSOXALC. In the end, the single allocations will be executed through the command:

```
    %ALCFLY
```

The TSOXALC table looks like:

```
PROC Ø                                           /*ZOSTSO:ALL      */
/* ------------------------------------------- /*ZOSTSO:ALL      */
/* SAMPLE TABLE TSOXALC                        /*ZOSTSO:ALL      */
/* ------------------------------------------- /*ZOSTSO:ALL      */
   CONTROL NOFLUSH NOMSG MAIN                    /*ZOSTSO:ALL      */
PROFILE    MODE WTPMSG MSGID                     /*ZOSTSO:ALL      */
                                                 /*ZOSTSO:ALL      */
FREE FI(ISPLLIB,ISPPLIB,ISPMLIB,ISPTLIB,         /*ZOSTSO:ALL      */ +
        ISPSLIB,ISPTABL,SMPTABL,ISPMALT,         /*ZOSTSO:ALL      */ +
        IPCSPARM,ISPPALT,ISPMALT,ISPILIB,        /*ZOSTSO:Y        */ +
        HCDPROF)                                 /*ZOSTSO:ALL      */
/* ------------------------------------------- /*ZOSTSO:ALL      */
SET &DSNAME = &SYSUID..PROFILE                   /*ZOSTSO:ALL      */
.............................                    /*ZOSTSO:ALL      */
CARD FOR ALLOCATION USER PROFILE                 /*ZOSTSO:ALL      */
 "   "         "    "    "                      /*ZOSTSO:ALL      */
 "   "         "    "    "                      /*ZOSTSO:ALL      */
 .............................                   /*ZOSTSO:ALL      */
/* ------------------------------------------- /*ZOSTSO:ALL      */
FREE FI(SYSPROC)                                 /*ZOSTSO:ALL      */
ALLOC FI(SYSPROC) SHR DA(                        /*ZOSTSO:ALL      */ +
      ,CPAC.CMDPROC'                             /*ZOSTSO:WKY      */ +
      ,INST.CPAC.SCPPCENU'                       /*ZOSTSO:KY       */ +
      ,USER.PDS.SYSPROC.TYPE1'                   /*ZOSTSO:WKQY     */ +
      ,USER.PDS.SYSPROC.TYPE2'                   /*ZOSTSO:WKQL     */ +
```

```
      ,USER.PDS.SYSPROC.TYPE3'                    /*ZOSTSO:JY        */ +
      ,............'                              /*ZOSTSO:.....     */ +
                  )                               /*ZOSTSO:ALL       */
%EXAMPLE                                          /*ZOSTSO:.....     */
ANY_USER_CMD ..                                   /*ZOSTSO:.....     */
...............                                   /*ZOSTSO:.....     */
...............                                   /*ZOSTSO:.....     */
                                                  /*ZOSTSO:ALL       */
FREE FI(........)                                 /*ZOSTSO:.....     */
ALLOC FI(.......) SHR DA(                         /*ZOSTSO:.....     */ +
ANY_ALLOCATION ....'                              /*ZOSTSO:.....     */ +
      '............'                              /*ZOSTSO:.....     */ +
                  )                               /*ZOSTSO:ALL       */
/* ----------------------------------------- /*ZOSTSO:ALL       */
ERROR RETURN                                      /*ZOSTSO:ALL       */
PDF PANEL(ISPPRIM) LOGO(........)                 /*ZOSTSO:ALL       */
```

This is where I put all the allocation information necessary for all the TSO users.

In the comments zone on the right (from column 59 to 68), I specify:

- ALL – to execute the allocation for all the users.
- C – an alphanumeric byte to select a user's category.

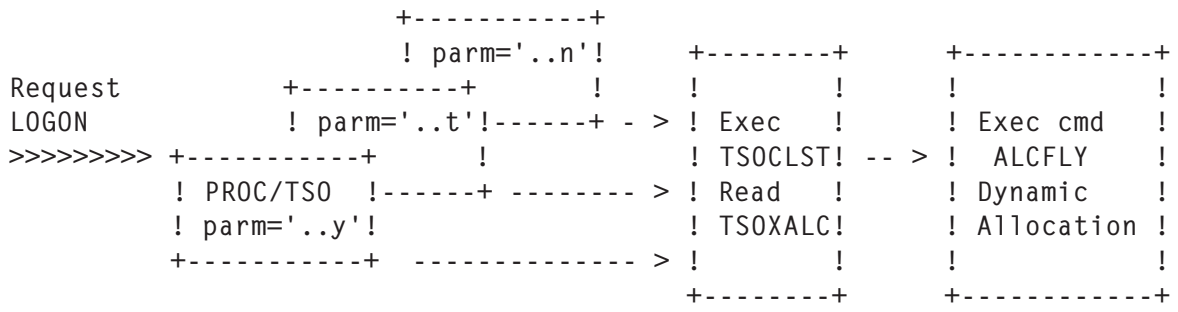
All the users defined are in the variable RANGEOK in the CLIST TSOCLST.

Finally, with the following JCL, TSOVERF, I can, at any time, verify the syntax of the CLIST/card executed for each user's category during LOGON processing:

```
//TSOVERF  JOB ..............................
//** -------------------------------- **
//** SIMULATE ALLOCATION FOR LOGON PROC **
//** -------------------------------- **
//TSOVERF1 EXEC PGM=IKJEFTØ1,
//    PARM='%TSOCLST Y'                                    <- UPDATE
//SYSPROC  DD   DISP=SHR,DSN=your.user.pds.clist           <- UPDATE
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//FWORKINP DD   DISP=SHR,DSN=your.user.pds.clist(TSOXALC)  <- UPDATE
//FWORKOUT DD   DISP=(,PASS),UNIT=VIO,SPACE=(TRK,(1,1,1)),
// DSN=&TEMP(ALCFLY),DCB=SYS1.PARMLIB
//** -------------------------------- **
//TSOVERF2 EXEC PGM=IEBGENER,COND=(Ø,NE)
//SYSUT1   DD   DISP=(OLD,PASS),DSN=&TEMP(ALCFLY)
```

```
//SYSUT2   DD   SYSOUT=*
//SYSIN    DD   DUMMY
//SYSPRINT DD   DUMMY
```

In summary:



Massimo Ambrosini
Systems Programmer (Italy)

Dataset browse utility

PURPOSE

This tool can be used to open a dataset in edit/view/browse mode from JCL.

USAGE

This tool is used typically in JCL when the user wants to open the dataset that's referred to in the JCL DD statements.

Most of us would like to use shortcuts when we work on mainframe systems. When it comes to datasets, certainly it would be really useful if we had a handy tool to open a dataset without doing any copying/pasting of dataset names or switching between panels and typing a long dataset name(s). It becomes a very tedious job if the dataset happens to be a GDG with a relative generation number because we need to go back and start looking for the correct version.

DU is designed to work like a REXX macro on datasets opened in edit/view mode(s).

Set-up procedure:

1 Copy the source code below into a PDS and name it ‘DU’.

2 Make sure that you have allocated this PDS to your profile, otherwise you will get an error message like ‘invalid command’ when you issue ‘DU’.

```
/*rexx*/
/*********************************************************************/
/* This REXX is to view/edit/browse a non-VSAM dataset coded in    **/
/* standard JCL.                                                     */
/*                                                                   */
/* Requirements: The JCL dataset name needs to be fully resolved     */
/*                                                                   */
/* HOW TO USE  :*Type in the command <du> at the command prompt      */
/*               after opening the JCL in view/edit mode.            */
/*                                                                   */
/*              *Place the cursor on a line where in the dataset     */
/*               name appears in the JCL with the keyword DSN=       */
/*              *Cursor can be anywhere (any column) on the line     */
/*               that has DSN= Keyword.                              */
//*             *Press ENTER after performing the above steps.       */
/*              *If the dataset is valid, it will open in view mode */
/*              *If it's invalid, then an appropriate error message */
/*               will be thrown.                                     */
/*              *If the dataset is to opened in edit or browse mode */
/*               type <du e> or <du b> at the command prompt to      */
/*               indicate whether you want to see it in EDIT mode or*/
/*               BROWSE mode.                                        */
/*              *Specify any other parm and <du> will assume         */
/*               VIEW mode, which is default.                        */
/*********************************************************************/
"isredit macro (mode)"
upper mode
if (mode ¬= 'B' & mode ¬='E' & mode ¬= 'V') then
    mode = 'V'
"isredit (dsn) = Line .zcsr"
strt_pos = Index(dsn,'DSN=') + 4
if strt_pos <= 4 then
 do
  errmsg = "'DSN=' keyword expected but not found"
  call setmsg_rtn
 end
end_poss = Index(dsn,' ',strt_pos)
end_posc = Index(dsn,',',strt_pos)
```

```
if end_poss > Ø then
   do
   if end_posc > Ø then
      do
        if end_posc < end_poss then
           end_pos=end_posc
        else
           end_pos=end_poss
      end
   else
      end_pos=end_poss
   end
else
   if end_posc > Ø then
      end_pos=end_posc
   else
      do
        errmsg = 'Delimiter Space or Comma not found. No action taken'
        call setmsg_rtn
      end
len_dsn = end_pos - strt_pos
dsn = strip(substr(dsn,strt_pos,len_dsn))
Call identify_DSN
"ispexec control errors return"
x = outtrap(msg.)
" listcat entries('"dsn"')"
x = outtrap(off)
if word(msg.1,1)='GDG' & word(msg.1,2)='BASE' then
   do
     errmsg = 'DSN is a GDG BASE.'
     call setmsg_rtn
   end
if sysdsn("'"||dsn||"'") = 'DATASET NOT FOUND' then
  do
    errmsg = 'Dataset does not exist'
    call setmsg_rtn
  end
else
    if substr(sysdsn("'"||dsn||"'"),1,2Ø) = 'INVALID DATASET NAME' then
      do
        errmsg = 'Invalid dataset Name'
        call setmsg_rtn
      end
dsn = holddsn
  if mode = 'B' then
     do
        "ispexec Browse dataset('"dsn"')"
        Select
        when rc = Ø then
             Signal Close_rtn
```

```
        when rc = 12 then
             errmsg = 'PS file is empty or Member in the PDS is empty'
        when rc = 14 then
             errmsg = ' Member/Dataset is in Use (Contention)'
        when rc = 16 then
             errmsg = 'No members found in the given pattern/Empty PDS'
        when rc = 18 then
             errmsg = 'VSAM datasets not supported'
        otherwise
             errmsg = 'Unknown error'
        end
        call setmsg_rtn
     end
  else
     do
        if mode = 'E' then
         "ispexec Edit dataset('"dsn"')"
        else
         "ispexec View dataset('"dsn"')"
        Select
        when rc = 0 | rc = 4 then
             Signal Close_rtn
        when rc = 14 then
             errmsg = 'Member does not exist'
        when rc = 16 then
             errmsg = 'No Match found/Invalid member name'
        when rc = 18 then
             errmsg = 'VSAM datasets not supported'
        otherwise
             errmsg = 'Unknown error'
        end
        call setmsg_rtn
     end
Close_rtn:
Return
identify_DSN:
holddsn = dsn
 if substr(reverse(dsn),1,1) = ')' then
    do
      op_pos = Index(reverse(dsn),'(')
      str_enc = reverse(substr(reverse(dsn),2,(op_pos - 2)))
      if substr(str_enc,1,1) = '+' then
         do
           if substr(str_enc,2) = 0 then
              do
                vers = 0
                Call Get_version
              end
           else
              do
```

```
                errmsg = 'GDG Name with +version can not be viewed'
                Signal setmsg_rtn
              end
          end
       else
          if substr(str_enc,1,1) = '-' then
             do
               vers = substr(str_enc,2)
               Call Get_version
             end
          else
             if datatype(str_enc) = 'NUM' then
                do
                  if str_enc = Ø then
                     do
                       vers = Ø
                       Call Get_version
                     end
                  else
                     do
                      errmsg = 'Member name is invalid'
                      Signal setmsg_rtn
                     end
                end
             else
                do
                  dsn = reverse(substr(reverse(dsn),op_pos+1))
                  mem = strip(str_enc)
                end
    end
 Return
Get_version:
 if datatype(vers) = 'NUM' then
    do
      gdg = reverse(substr(reverse(dsn),op_pos+1))
      Call get_absgen
      if cc = Ø then
         dsn = gdg
      else
         do
           if cc = 2 | cc = 4 then
              errmsg = '-'||vers||' Version does not'||,
                                                ' exist for this base'
           else
              errmsg = 'Listcat failed. Check the basename'
           Signal setmsg_rtn
         end
    end
Return
get_absgen:
```

```
gdg = strip(gdg,B,"'")
x   = outtrap(msg.)
"listcat entries('"gdg"') "
lc_rc = rc
x   = outtrap(off)
if lc_rc = 0 then
  do
    if msg.0 = 2 then
       do
         cc = 2
         gdg = ''
       end
    else
       if msg.0 > (vers * 2 + 2) then
          do
            cc = 0
            tmp = msg.0 - ((vers * 2) + 1)
            gdg = word(msg.tmp,3)
          end
       else
          do
            cc = 4
            gdg = ''
          end
  end
else
  do
    cc = 8
    gdg = ''
  end
Return

setmsg_rtn:
 zedlmsg = errmsg
 "ispexec setmsg msg(isrz001)"
 Exit 8
```

Execution:

1 Open JCL in edit or view mode.

2 At the command prompt, type in the command name ‘DU’ and place the cursor anywhere on a line that has the keyword *DSN*= (*DSN*= should be in upper-case only).

3 Hit the *Enter* key.

4 Based on the dataset name and environment, the tool will display appropriate error messages.

Error message	*Cause*	*Solution*
'DSN=' keyword expected but not found	The line on which the cursor is placed does not have the keyword, 'DSN='	Place the cursor on the line where the keyword 'DSN=' is coded with the dataset name that you are trying to open.
DSN is a GDG BASE	DSN= keyword contains the dataset name which is a GDGBASE name without any relative generation number	Use option 3.4 to view the GDG base and available versions or use relative generation number, if known.
Invalid dataset name	The dataset name violates mainframe DS naming standards	Correct the DS name and try again.
PS file is empty or member in the PDS is empty	In browse mode, an attempt was made to open an empty DS or empty member in a PDS	Browse mode, its not possible to open a empty dataset. Use EDIT/VIEW instead.
Member does not exist	In browse mode, an attempt was made to open a member that's not present in the PDS	Check the member name or use EDIT/VIEW mode instead.
No members found in the given pattern/empty PDS	The PDS is empty or the wildcard specified does not find any matching members in the PDS	Check the wildcard or PDS.
VSAM datasets not supported	DSN= keyword contains a dataset which is a VSAM file	VSAM files are not supported by this tool. Use FileAid or any other tool to open VSAM type datasets.
Unknown error	Unknown error occurred while opening the dataset	Try again.
Member/dataset is in use (contention)	The dataset or the member is being used by the other user/job in EDIT/OLD mode.	Wait until the job completes or user releases the dataset/use browse or view mode/check whether the dataset is opened in a screen in SPLIT mode by yourself.

Figure 1: Errors, causes, and solutions (continues)

Error Message	*Cause*	*Solution*
GDG name with +version cannot be viewed	Any dataset-name(+XX) is not supported with this tool because this tool assumes that the dataset name is GDG BASE. Hence it does not take the +version to find out the absolute generation.	Correct the version number or check the member name in case of PDS and try again.
Member name is invalid	The member name contains only numeric digits.	Check the sign or member name.
-XX version does not exist for this base	-XX version is not an active generation/version for this base	The datasets may have been scratched. Check the limit parameter value of the base or check the version number specified. It can't be greater than 255
Listcat failed. Check the basename	The GDG basename is incorrect	The GDG base does not exist. Check the base name.

Figure 1: Errors, causes, and solutions (continued)

5 By default, VIEW mode is used while using the command 'DU'. If BROWSE or EDIT mode is desired, supply one of the parameters, B or E. The command name and the parameters are case-insensitive. Supplying any other parameter will cause the tool to assume VIEW mode.

Figure 1 shows potential errors, their causes, and solutions.

LIMITATIONS/WARNING

The tool terminates its execution as soon as it finds the first error.

There could be situations like the following:

```
DSN=INVALID.BASE.NAME(+1)
```

When you try DU on this, it may give you an error message ‘GDG Name with +version cannot be viewed’ instead of saying ‘Listcat failed. Check the Basename’ as you might have expected. The order with which the error situations are analysed by this tool is solely at the author’s discretion and hence it is not guaranteed that the tool will always give you the error message that you expect.

Suresh Kumar Murugesan
Systems Analyst
Cognizant Technology Solutions (USA)

Unix System Services interface for z/OS

This interface can be used to write and execute Unix script files in z/OS without a Unix emulation program such as telnet. (Telnet is a program that is used to connect to all Unix operating systems.)

CODEPAGE

There is a default codepage for the z/OS operating system and USS (Unix System Services) environment.

The z/OS Unix shells and utilities are configured as EBCDIC programs, which means that they encode their characters in the EBCDIC codeset.

The default codepage for the USS (Unix System Services) is IBM-1047. These defaults are in the TCP/IP start-up configuration file at the DDname PROFILE.

Browse the file associated with the profile DDname and look for:

```
codepage
```

The result can be:

```
CodePage ISO8859-1 IBM-1Ø47
```

The second codepage (IBM-1047) is the USS environment default.

To connect to TSO we use an emulator and the characters typed or shown during the session follow the codepage set up in the emulator.

For example, if the codepage of our emulator is IBM-280, when we type the characters #!/bin/sh they will be shown as £é/bin/sh in the USS environment.

For this reason I have written an interface that converts the codepage automatically.

WHERE TO FIND THE EMULATOR CODEPAGE

From the emulator bar command choose **Communications** – see Figure 1. Select **configure** and **session parameter** and the emulation codepage shown in Figure 2 will appear.

Our emulation codepage is IBM-1047.

PMCOPY

This program manages option 2 to copy the script from z/OS to the USS environment with the correct code page (IBM-1047).

```
/* rexx */
/* This program manages the option #2
   Copy the script from ZOS to USS environment
```

Figure 1: Emulator bar

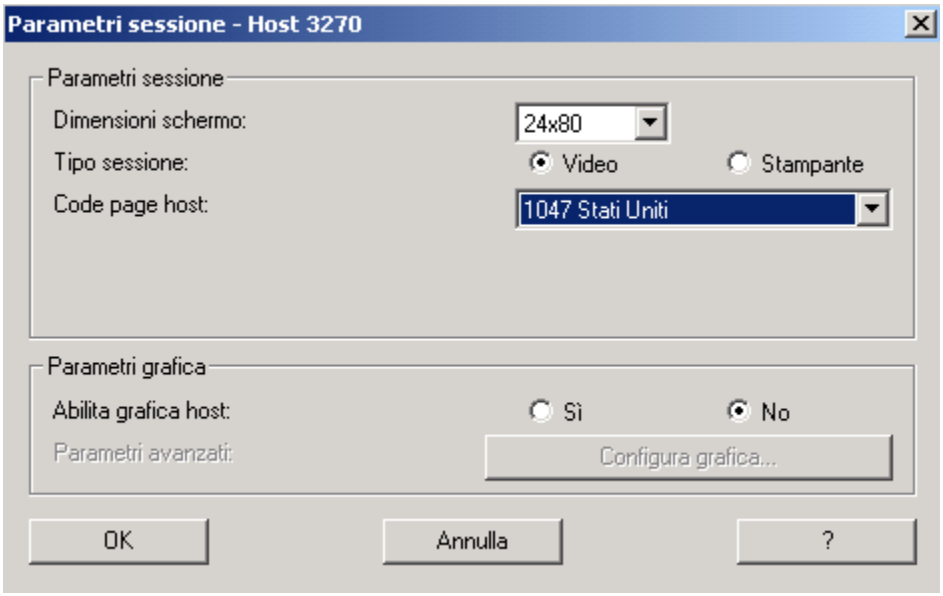


Figure 2: Emulation codepage

```
   with the correct code page (IBM-1047)
*/
address ispexec "vget (zosdsn zosmbr usspath ussfile" ,
  "usscpage ussibmcp ussp) profile"
if bpxwunix('cd 'usspath,,stack) ¬= 0
                                    /*Exec cd(change dir) USS command */
                  then do
                         address ISPEXEC "SETMSG MSG(PM002)"
                         return
                        end
dsn_input = zosdsn"("zosmbr")"          /* Input dataset */
if RIGHT(usspath,1) ¬= '/' then usspath = usspath||"/"
path_input = usspath||f||time('S')  /* temporary dataset */
if bpxwunix('touch 'path_input,,stack) ¬= 0 /* Create empty file */
                  then do
                         address ISPEXEC "SETMSG MSG(PM003)"
                         return
                        end
path_output = usspath||ussfile
conv_comm ="iconv -f "usscpage ,
              " -t "ussibmcp" "path_input">"path_output
chmod_comm="chmod "ussp" "path_output
call chk_alloc 'INPUT'       /* Check if DDname is already allocated */
call alloc_input_ddname      /* Alloc DDname and dataset */
```

```
call chk_alloc_err rc INPUT  /* Check allocation return code */
call chk_alloc 'PATHNAME'
call alloc_input_pathname
call chk_alloc_err rc PATHNAME
call copy_to_uss             /* Copy program from z/os to USS */
if rc ¬= Ø then do
                 address ISPEXEC "SETMSG MSG(PMØØ4)"
                 return
                end
if bpxwunix(conv_comm,,stack) ¬= Ø     /* Convert codepage zos -> USS */
                  then do
                         address ISPEXEC "SETMSG MSG(PMØØ5)"
                         return
                        end
if bpxwunix(chmod_comm,,stack) ¬= Ø /* Exec chmod USS command */
                  then do
                         address ISPEXEC "SETMSG MSG(PMØØ6)"
                         return
                        end
x = msg('off')
address tso "free da("dsn_input")"
x = msg('on')
call bpxwunix 'rm 'path_input,,out. /*Exec rm(remove file) USS command
*/
ussf = path_output
address ispexec "VPUT (ussf) profile"
address ISPEXEC "SETMSG MSG(PMØØ8)"
exit

chk_alloc:
ddname=arg(1)
x = msg('off')
rcconv = LISTDSI(ddname 'FILE')
   if rcconv ¬= Ø then "FREE DD("ddname")"
x = msg('on')
return

chk_alloc_err:
arg retcode ddname
if retcode ¬= Ø then do
                     address ISPEXEC "SETMSG MSG(PMØØ7)"
                     exit
                    end
return

alloc_input_ddname:
"ALLOC DD(INPUT) DATASET('"dsn_input"') SHR REU"
return

alloc_input_pathname:
```

```
 "ALLOC DD(PATHNAME) PATH('"path_input"')" ,
 "PATHMODE(SIRWXU,SIXGRP,SIRGRP) PATHOPTS(OCREAT, OWRONLY)"
 return

copy_to_uss:
 "OCOPY INDD(INPUT) OUTDD(PATHNAME) TEXT"  /*Copy datset to USS file */
 return
```

PMRUNUSS

This program manages option 4 to execute the script in the USS environment.

```
/* Rexx */
/* This program manages the option #4
   Exec the script to USS environment.
*/
address ispexec
"vget (zosdsn usspath ussfile) profile"
if RIGHT(usspath,1) ¬= '/' then usspath = usspath||"/"
command = usspath||ussfile
command = strip(command)
point = pos('.',zosdsn)
point = point - 1
alias = left(zosdsn,point)
tmpout = alias||".out"
call bpxwunix 'rm 'tmpout,,out.
call bpxwunix command '1>>'tmpout '2>>'tmpout,,out.
do while queued()>0      /* Empty the stack queue */
 pull x
end
address tso
"obrowse "tmpout
exit
call bpxwunix 'rm 'tmpout,,out.
```

PMSTART

This is the start-up program.

```
/* REXX */
dsn = "user.LIB.TEST"
address TSO "ALTLIB ACTIVATE APPLICATION(CLIST) DA("dsn")"
address ISPEXEC "LIBDEF ISPPLIB DATASET ID("dsn") STACK"
address ISPEXEC "LIBDEF ISPMLIB DATASET ID("dsn") STACK"
address ISPEXEC "SELECT PANEL(PMPPRIM) NEWAPPL(ISR) PASSLIB"
address ISPEXEC "LIBDEF ISPMLIB"
address ISPEXEC "LIBDEF ISPPLIB"
```

```
address TSO "ALTLIB DEACTIVATE APPLICATION(CLIST) "
exit Ø
```

PMUSSED

This program manages option 3 to edit the script from the USS environment with the correct code page (IBM-1047).

```
/* Rexx */
/* This program manages the option #3
   Edit the script from USS environment
   with the correct code page (IBM-1Ø47)
*/
address ispexec
"vget (zosdsn usspath ussfile) profile"
if RIGHT(usspath,1) ¬= '/' then usspath = usspath||"/"
dsname = usspath||ussfile
dsname = strip(dsname)
if bpxwunix('cat 'dsname,,stack) ¬= Ø
                  then do
                         address ISPEXEC "SETMSG MSG(PMØØ6)"
                         return
                        end
do while queued() > Ø   /* Empty stack queue */
 pull x
end
address tso
"oedit " dsname
```

PMZOSED

This program manages option 1 to edit or create the dataset, which can contain a sequence of Unix Perl commands.

```
/* Rexx */
/* This program manages the option #1
   Edit,create the dataset that can contain
   a sequence of unix,perl,..... command
*/
address ispexec
"vget (zosdsn zosmbr) profile"  /* get variables from panel */
x = msg('off')
if zosmbr ¬= ' '                /* The member is not specified */
 then dsname = zosdsn"("zosmbr")"
 else dsname = zosdsn
check_dsn = SYSDSN("'"dsname"'") /* Check if the dsname or the */
                                 /* Member name exist          */
```

```
if check_dsn = "OK"              /* Dsname ok */
         then do
                Address ispexec
                'LMINIT DATAID(DID) DATASET('zosdsn') ENQ(EXCLU)'
                'LMOPEN DATAID(&DID) OPTION(OUTPUT)'
                'LMPUT DATAID(&DID) MODE(INVAR) ' ,
                'DATALOC(LINE) DATALEN(8Ø)'
                if zosmbr = ' '
                          then do
                                'ISPEXEC LMMDISP DATAID(&DID) ' ,
                                'OPTION(DISPLAY) MEMBER(*)'
                                zosmbr = ZLMEMBER
                                address ispexec "VPUT (zosmbr) profile"
                                'LMFREE DATAID(&DID)'
                                exit
                               end
                           else do
                                address ISPEXEC  "EDIT DATASET("dsname")"
                                 exit
                                end
              end
         else do
                if check_dsn = "MEMBER NOT FOUND"  /* Member not found */
                then do
                       'LMINIT DATAID(DID) DATASET('zosdsn') ENQ(EXCLU)'
                       'LMOPEN DATAID(&DID) OPTION(OUTPUT)'
                       'LMPUT DATAID(&DID) MODE(INVAR) ' ,
                       'DATALOC(LINE) DATALEN(8Ø)'
                       'LMMADD DATAID(&DID) MEMBER('zosmbr')'
                       address ispexec "VPUT (zosmbr) profile"
                       'LMFREE DATAID(&DID)'
                        address ISPEXEC  "EDIT DATASET("dsname")"
                        exit
                      end
                else do
                address ISPEXEC "SETMSG MSG(PMØØ1)" /* Get the message */
                                                     /* from library */
                address ispexec "VPUT (zosmbr) profile" /* Save member */
                                                          /* name */
                      end
         end
```

PMPPRIM PANEL

```
 )attr default(%+_)
~ type(text) intens(high) caps(off) just(asis ) color(turq)
hilite(reverse)
% type(text) intens(high) color(white)
+ type(text) color(blue)
! type(text) color(red)
```

```
@ type(output) caps(off) color(blue)
_ type(input) caps(off)
# type(input) caps(on)
)Body Expand(\\)
%-\-\- ~UNIX System Services Script Manager%-\-\-
%
!Selection ===>_ZCMD +
%
%
+1%  Edit/Modify Z/OS file
     +Dsname .....#zosdsn                                          +
      +Member .....#zosmbr  +
 +2%  Convert codepage and copy program to USS environment
 +     USS Path ............. _usspath                                  +
 +     USS output file ...... _ussfile                              +
 +     USS CodePage  ........ _ussibmcp    +
 +     Emulation CodePage.... _usscpage    +
 +     USS file permission .. _ussp             +
 +
 +3%  Edit/Modify USS file .... @ussf
 +4%  Run USS file ............. @ussfr
 +x%  Exit
 +
 +
 +Userid: &ZUSER  +  Date: &ZDATE  +   Time: &ZTIME +
 )INIT
 .CURSOR = ZCMD
 VGET (zosmbr usscpage ussp ussibmcp) PROFILE
  IF  (&usscpage = ' ')
      &usscpage = 'IBM-280'
 IF  (&ussp = ' ')
      &ussp = '750'
 IF  (&ussibmcp = ' ')
      &ussibmcp = 'IBM-1047'
&zcmd = ' '
&ussfr= &ussf
)REINIT
&ussf = ' '
&message = ' '
)PROC
 VER (&zosdsn,NB)
 VER (&usspath,NB)
 VER (&ussfile,NB)
 VER (&usscpage,NB)
 VER (&ussp,NB)
 VER (&ussibmcp,NB)
 VPUT (zosdsn zosmbr usspath ussfile usscpage ussibmcp ussp) PROFILE
&zsel = trans(&ZCMD
              1,'CMD(PMZOSED)'
              2,'CMD(PMCOPY)'
```

```
              3,'CMD(PMUSSED)'
              4,'CMD(PMRUNUSS)'
              x,'EXIT'
              *,'?'
              )
)end
```

MESSAGES

```
PM00:             Message member where the error messages are stored.
PM001  'Invalid DSN/Member' .ALARM=YES
'PMZOSED: Invalid Dsname or Member'
PM002  'Invalid USS path' .ALARM=YES
'PMCOPY: USS path does not exist'
PM003  'Error - create tmp file' .ALARM=YES
'PMCOPY: Error during creation tmp file '
PM004  'Error - copy to USS' .ALARM=YES
'PMCOPY: Error during copy the program/script to USS'
PM005  'Error - file conversion' .ALARM=YES
'PMCOPY: Error during file conversion z/os to USS'
PM006  'Error - chmod' .ALARM=YES
'PMCOPY: Error during to change file owner to USS file'
PM007  'Error - alloc DDname' .ALARM=YES
'PMCOPY: DDname allocation error'
PM008  'All good - Rc=0' .ALARM=YES
'PMCOPY: The program has been copied succesfully'
```

SAMPLE UNIX SCRIPT COMMAND

```
#!/bin/sh
echo "Show current directory"
echo "---------------------\n"
pwd
echo "\nShow current directory files"
echo "----------------------------\n"
ls -la
echo "\nDisplay filesystem"
echo "------------------\n"
df
```

INSTALLATION

To install the interface copy all member into user.lib.test.

START-UP

To start the interface, from TSO option 6, type:

```
ex 'user.lib.test(pmstart)'
```

This produces the screen shown in Figure 3.

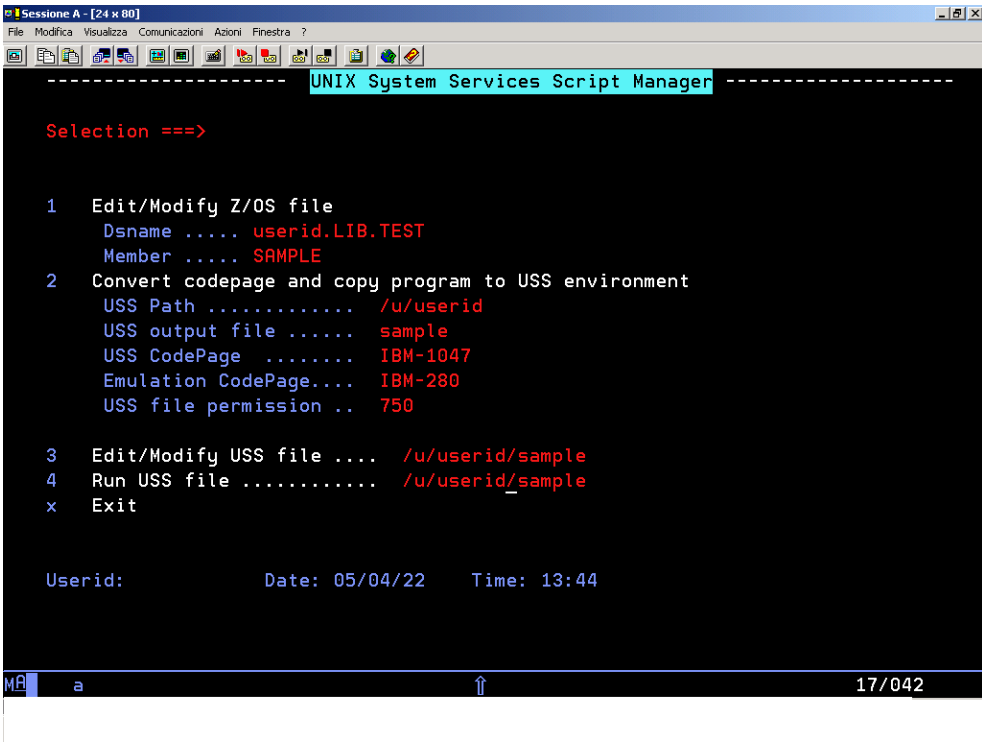


Figure 3: Initial screen

USER GUIDE

The interface manages only partitioned datasets, not sequential files.

To execute the Unix script file there are three steps to perform and they are sequential.

Environment set-up

Before we start we must set up the USS environment.

Option 1

This option is used in order to create a new script command or to modify one that already exists.

To modify a script, insert in both the dataset name and member fields. If you enter only the dataset name field (no member field), you will be shown the directory from which you can select the member you want.

```
1   Edit/Modify Z/OS file
     Dsname ..... userid.LIB.TEST
     Member ..... SAMPLE
```

Option 2

- In the USS path field insert your Unix directory from which you are copying the Unix script file. You don't have to insert a '/' as the last character – */u/userid/* or */u/userid*.
- In the USS output file field insert your Unix script file name.
- In the USS codepage insert your USS code page. The default is IBM-1047.
- In the emulation codepage insert your emulator codepage.
- In the USS file permission insert the Unix script permission file.

  The default is 750 and means:

  ```
  user -> rwx group -> r-x other ---
  ```

  – The user can read, modify, and execute the file.

  – The group can only read and execute the file.

  – Other can't read, modify, or execute the file.

The user that starts the REXX program must have Open Edition Segment (OMVS).

The OMVS segment is created, in the case of the RACF security system, with the following commands:

```
ALTUSER userid  OMVS(UID(1111) GID(2222) HOME('/u/userid') PROGRAM('/
bin/sh'))
```

where *1111* is the ID assigned to the user and *2222* is the ID assigned to the group.

Option 2 is used in order to copy the z/OS input dataset

userid.LIB.TEST (SAMPLE) in the Unix directory */u/userid* and convert the codepage from IBM-280 to IBM-1047:

```
2   Convert codepage and copy program to USS environment
     USS Path ............  /u/userid
     USS output file ......  sample
     USS Code Page ........  IBM-1047
     Emulation Code Page ..  IBM-280
     USS file permission ..  750

3   Edit/Modify USS file ....  /u/userid/sample
4   Run USS file ............  /u/userid/sample
```

If there aren't any problems, in the right corner of the screen the following message will be shown:

```
All good - Rc=0
```

and the field to the right of the options number 3 and 4 automatically will be filled up with the complete name of the program.

Option 3

This option is not mandatory and can be used to show the script file:

```
3   Edit/Modify USS file ....  /u/userid/sample
```

The result could be the following:

```
EDIT       /u/userid/sample                          Columns 00001 00072
Command ===>                                           Scroll ===> HALF
****** ************************* Top of Data ***************************
000001 £é/bin/sh
000002 echo "Show current directory"
000003 echo "---------------------çn"
000004 pwd
000005 echo "çnShow current directory files"
000006 echo "----------------------------çn"
000007 ls -la
000008 echo "çnDisplay filesystem"
000009 echo "------------------çn"
000010 df
```

Option 4

Option 4 is used to run the script program and to show the report:

```
 4   Run USS file ............  /u/userid/sample
```

The result could be the following:

```
Show current directory
----------------------

/u/userid

Show current directory files
----------------------------

total 608
drw-------   2 userid  usergrp      8192 Apr 22 06:33 .
drwxrwxrwx  44 unistc  30          8192  Apr 20 07:50 ..
-rw-------   1 unistc  usergrp      2367 Apr 18 07:53 .sh_history
-rw-rw----   1 unistc  unixgrp        16 Mar 29  2004 PRDELSI.userid.ERR
-rw-r-----   1 unistc  unixgrp       371 Mar 29  2004 PRDELSI.userid.LOG
-rw-rw----   1 unistc  unixgrp        84 Mar 29  2004 PRDELSI.userid.LSI
-rw-rw----   1 unistc  unixgrp        11 Mar 29  2004 PRDELSI.userid.LST
-rw-rw-rw-   1 userid  usergrp       117 Apr 22 06:33 userid.out
-rwxr-x---   1 userid  usergrp       810 Apr 22 06:33 sample
-rwxr-x---   1 userid  usergrp       567 Apr 21 12:58 test03

Display filesystem
------------------

Mounted on                        Filesystem                 Avail/Total
Files      Status
/DEV2/etc/licmgmt/ilmadb      (OMVS.DEV2.DEVLPLEX.ILMSPDM)      28624/
28800     4294967294 Available
/DEV2/etc/licmgmt/certs       (OMVS.DEV2.DEVLPLEX.ILMUC)        28624/
28800     4294967294 Available
/DEVE/var                     (OMVS.DEVE.DEVLPLEX.VAR)          13296/
14400     4294967265 Available
/DEVE/etc                     (OMVS.DEVE.DEVLPLEX.ETC)          48992/
86400     4294967036 Available
/R0TB2B                       (OMVS.R4TA1A.SVILPLEX.ROOT)       1263392/
3556800 4294958306 Available
/DEV2/dev                     (OMVS.DEV2.SVILPLEX.DEV)          14080/
14400     4294967279 Available
/DEV2/tmp                     (OMVS.DEV2.SVILPLEX.TMP)          27528/
43200     4294967169 Available
/DEV2/var                     (OMVS.DEV2.SVILPLEX.VAR)          13296/
14400     4294967253 Available
/DEV2/etc                     (OMVS.DEV2.SVILPLEX.ETC)          83976/
86400     4294967024 Available
/DEV2                         (OMVS.DEV2.HFSSYSS)               28552/
28800     4294967284 Available
/APPLIC/cicsmq                (OMV000.DEVL.CICSMQ)              2840/
```

```
21600      4294966967 Available
/APPLIC/ellips                (OMV000.DEVL.ELLIPS)              15456/
20544     4294967269 Available
/APPLIC/netview               (OMV000.DEVL.NETVIEW)             63808/
86400     4294967229 Available
/APPLIC/java                  (OMV000.DEVL.JAVA131)             435576/
1036800  4294966105 Available
/APPLIC/PRDserve              (OMV000.DEVL.PRDSERVE.HFS)        798816/
835200   4294967162 Available
/u                            (OMV000.DEVL.USER)                1524272/
2259840 4294966101 Available
/APPLIC/npm                   (OMV000.DEVL.NPM)                 224/480
4294967288 Available
```

Magni Mauro
Systems Engineer (Italy)

Enclave resource accounting – part 2

This month we publish a description of the enclave report writer and the code.

In order to provide a starting point from which one can begin to measure and analyse enclave resource usage I have coded a sample report writer.

Several reports are generated by this report writer. The first one is *Enclave CPU report*, which is in fact an overview of various processor timings (total, TCB, SRB, I/O, and RCT) for address space, pre-emptable SRBs, and enclaves (independent/dependent):

```
                      -- PROCESSOR TIMINGS (sec.)--
                            Enclaves   Preemp.  TCB distribution (%)
Jobid   Job     #Ind  Total Indpt. Dep. Srb    Asid Indep. Dep. Preemt.
-------------------------------------------------------------------------
STC2906 DSNDIST  148   15.6  14.30  0   0.08   1.30  98.15  0   0.55
STC4715 DSNDIST  151   17.1  15.41  0   0.09   1.71  97.72  0   0.57
STC5533 DSNDIST  311    4.9   1.24  0   0.20  17.24  71.26  0  11.49
STC6228 DBTDIST  243  250.2 244.99  0   0.14   0.19  99.76  0   0.06
STC7406 DSNDIST  523   69.2  62.81  0   0.33   0.94  98.54  0   0.52
```

```
STC7437 DBTDIST  403   53.6  48.21  0   0.25   0.68  98.81  0   0.51
STC6696 DBLDIST  371   31.6  23.18  0   0.27  14.38  84.63  0   0.99
STC7848 BBOASR2  452   44.0  41.93  0      0   4.01  95.99  0      0
STC8046 DBLDIST 1320   51.2  32.94  0   0.94  21.50  76.32  0   2.18
STC0111 BBOASR2  269   40.4  38.51  0      0   3.94  96.06  0      0
STC0197 BBOASR2  356   40.1  38.12  0      0   4.03  95.97  0      0
STC0265 BBOASR2  137   35.1  33.43  0      0   3.96  96.04  0      0
STC0108 BBOSMS  6413  208.8 180.24  0      0  13.11  86.89  0      0
```

This report can be used to determine which enclaves are the heaviest consumers of CPU.

The *I/O Activity Report* is a detailed report that shows calculated delays (dataset delay and allocation delay), the count of blocks transferred, DASD I/O connect time, and enclave DASD I/O timing for both independent/dependent enclaves including connect, disconnect, and pending time:

```
                          DASD I/O timing     I/O distribution (%)
Average (sec.)
Jobid  Job name #Ind   conn  disc   pend   conn   disc   pend    conn
disc      pend
-------------------------------------------------------------------------
-----------------
STC2906 DSNDIST  148  0.330  1.895  0.047   8.37  35.86  35.78  0.00223
0.01280  0.00031
STC4715 DSNDIST  151  0.716  0.770  0.233  18.82  26.39  78.99  0.00474
0.00509  0.00154
STC5533 DSNDIST  311  0.440  1.076  0.157  12.10  27.85  73.67  0.00141
0.00346  0.00050
STC6228 DBTDIST  243  0.107  0.114  0.032   3.82   6.58  34.59  0.00044
0.00046  0.00013
STC7406 DSNDIST  523  2.021  3.948  0.480  38.88  56.70  83.14  0.00386
0.00754  0.00091
STC7437 DBTDIST  403  1.156  0.926  0.271  29.64  34.46  84.32  0.00286
0.00229  0.00067
STC6696 DBLDIST  371  0.081  0.238  0.018  17.99  68.85  19.52  0.00021
0.00064  0.00004
STC7848 BBOASR2  452  1.988  2.175  0.412  55.50  95.18  54.54  0.00439
0.00481  0.00091
STC8046 DBLDIST 1320  1.094  1.833  0.235  75.30  95.28  77.43  0.00082
0.00138  0.00017
STC0111 BBOASR2  269  2.616  0.796  0.574  62.70  90.69  63.10  0.00972
0.00295  0.00213
STC0197 BBOASR2  356  2.850  0.997  0.629  64.61  79.81  63.63  0.00800
0.00280  0.00176
STC0265 BBOASR2  137  2.578  0.825  0.569  63.00  80.46  63.49  0.01881
0.00602  0.00415
```

```
STC0108 BBOSMS  6413 14.430  6.417  2.891  75.28  76.07  74.95  0.00225
0.00100  0.00045
```

Performance Report is a detailed report on service units consumed by each independent enclave transaction timing (how long a transaction was active and how long it was real-storage resident) and WLM-related information (workload, service class, resource group, reporting class and performance related notes):

```
                                    SRM timing         SRM Service
Jobid  Job name #Ind Active  %Active  Avg.tm  Tcb su(K) %SRM su  Avg.su
-------------------------------------------------------------------------
STC2906 DSNDIST 148   81.70   0.819   0.552    87.50    98.22    605.43
STC1865 DSNDIST   1    0.68   0.000   0.680     0.15     2.89    158.00
STC4002 DSNDIST   3  459.52   0.860 153.173     0.40    18.62    137.00
STC4715 DSNDIST 151  872.56   5.069   5.779    94.42    97.84    640.32
STC6754 DSNDIST  98 1103.58   1.560  11.261  1812.05    99.89  18934.04
STC6777 DBTDIST  99 1224.48   1.935  12.368  1363.50    99.82  14103.26
STC7406 DSNDIST 523 1684.02   4.439   3.220   385.53    98.64    754.83
STC7437 DBTDIST 403  186.43   0.442   0.463   295.67    98.90    751.29
STC6617 BBOASR2   3   69.28  54.011  23.093     6.72    47.28   2293.67
STC6609 BBOSMS    1   13.63   0.260  13.630     0.87     2.72    888.00
STC6812 BBOASR2  23   85.76   5.808   3.729   135.52    94.40   6033.65
STC6816 BBOASR2   6   67.43   0.080  11.238    65.19    88.98  11126.50
STC6700 BBOSMS   79 1125.69   0.338  14.249    63.15    64.28    818.58
STC6906 BBOASR2  12   64.89   0.075   5.408    14.92    58.89   1273.00
STC6903 BBOSMS   20  308.56   0.356  15.428    16.05    35.07    821.80
STC6988 BBOSMS  123 1781.35   0.343  14.483    97.37    74.33    810.59
```

There are also two additional reports: *Multisystem Enclave System Report* and *Enclave Summary Report*.

CODE

```
//*--------------------------------------
//* Clean-up step: delete work files
//*--------------------------------------
//DEL        EXEC PGM=IDCAMS//SYSPRINT   DD SYSOUT=X
//SYSIN      DD *
    DELETE hlq.R304.DATA
    DELETE hlq.R97.DATA
    SET MAXCC=0
/*
//*-------------------------------------------------------------
//* The SMF extract job.
//* This job will extract the SMF records type 30.4 and 97 from
```

```
//* the SMF weekly dataset. Set interval (date & time) as you wish
//*-------------------------------------------------------------------
//DUMPSMF  EXEC PGM=IFASMFDP,REGION=0M
//INDD     DD DSN=your.smf.weekly.ds,DISP=SHR
//OUTDD304 DD DSN=&&SMF304,DISP=(NEW,PASS),
//         SPACE=(CYL,(25,5)),DCB=(your.smf.weekly.ds)
//OUTDD97  DD DSN=&&SMF97,DISP=(NEW,PASS),
//         SPACE=(CYL,(25,5)),DCB=(your.smf.weekly.ds)

//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
     DATE(yyyyddd,yyyyddd)
     START(hhmm)
     END(hhmm)
     INDD(INDD,OPTIONS(DUMP))
     OUTDD(OUTDD304,TYPE(30(4)))
     OUTDD(OUTDD97,TYPE(97))
/*
//*-------------------------------------------------------------------
//* Further data filtering & sorting
//*-------------------------------------------------------------------
//COPYSMF  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAW304   DD DSN=&&SMF304,DISP=(,PASS)
//RAW97    DD DSN=&&SMF97,DISP=(,PASS)
//SMF304   DD DSN= hlq.R304.DATA,
//         SPACE=(CYL,(5)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//SMF97    DD DSN= hlq.R97.DATA,DISP=(,PASS)
//         SPACE=(CYL,(5)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN   DD *
  SORT FROM(RAW304) TO(SMF304) USING(SMF4)
  SORT FROM(RAW97) TO(SMF97) USING(SMF9)
//SMF4CNTL DD *
  OPTION SPANINC=RC4,VLSHRT
* Select only the records with valid enclave CPU data
* Processor Accounting Section offset is: 470 + 1 = 471
* 69+471: 539,4 smf30asr - Preemptable/Client SRB
* 72+471: 543,4 smf30enc - Independent enclave CPU time
* 76+471: 547,4 smf30det - Dependent enclave CPU time
*
  INCLUDE COND=(6,1,BI,EQ,30,AND,23,2,BI,EQ,4,AND,
               471,4,BI,NE,0,AND,
               (539,4,BI,NE,0,OR,543,4,BI,NE,0,OR,547,4,BI,NE,0))
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
//SMF9CNTL DD *
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,97)
```

```
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//REX       EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50
//SYSEXEC   DD DISP=SHR,DSN=your.rexx.lib
//SYSTSPRT  DD SYSOUT=*
//SMF30     DD DD DSN= hlq.R304.DATA,DISP=SHR
//SMF97     DD DD DSN= hlq.R97.DATA,DISP=SHR
//SYSTSIN   DD *
prof nopref
%ENC304
%ENC97
/*
```

ENC304 EXEC

```
/* REXX EXEC to read and format SMF 30.4 records                */
/* for Enclave resource accounting.                             */
/* The accounting for resources consumed by an enclave          */
/* depends on whether it is an independent, dependent, or a     */
/* foreign enclave                                              */
Address TSO
Numeric digits 64
userid=SYSVAR(SYSUID)
r30cp  =userid||'.r30cp.rep'         /* Processor usage Report  */
r30io  =userid||'.r30io.rep'         /* I/O Activity report     */
r30pr  =userid||'.r30pr.rep'         /* Performance Report      */
r30ms  =userid||'.r30ms.rep'         /* Multisystem Enclave Rpt.*/
r30se  =userid||'.r30se.rep'         /* Summary Report          */
x = MSG('OFF')
If SYSDSN(r30cp) = 'OK'       /* check CPU report dsn validity */
Then "DELETE "r30cp" PURGE"
  "ALLOC FILE(s30cp) DA("r30cp")",
  "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
  "REUSE LRECL(185) RECFM(F B) "

 If SYSDSN(r30io) = 'OK'          /* check report dsn validity */
 Then "DELETE "r30io" PURGE"
   "ALLOC FILE(s30io) DA("r30io")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
   "REUSE LRECL(230) RECFM(F B) "

 If SYSDSN(r30pr) = 'OK'          /* check report dsn validity */
 Then "DELETE "r30pr" PURGE"
   "ALLOC FILE(s30pr) DA("r30pr")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
   "REUSE LRECL(230) RECFM(F B) "

 If SYSDSN(r30ms) = 'OK'          /* check report dsn validity */
 Then "DELETE "r30ms" PURGE"
```

```
   "ALLOC FILE(s3Øms) DA("r3Øms")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
   "REUSE LRECL(85) "

 If SYSDSN(r3Øse) = 'OK'          /* check report dsn validity */
 Then "DELETE "r3Øse" PURGE"
   "ALLOC FILE(s3Øse) DA("r3Øse")",
   "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
   "REUSE LRECL(4ØØ) "

Cpu.1  =left(' ',8,' ')||'Enclave CPU report '||left(' ',15,' ')
Cpu.2  ='  '
Cpu.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø),
        ||left(' ',25,' ')||left('-',8,'-')||left(' ',7,' '),
        ||left('  PROCESSOR TIMINGS  ',21)||left(' ',8,' '),
        ||left('-',8,'-')
Cpu.4  =left(' ',126,' ')||left('Enclaves',18)||left('Preemptable',11),
        ||left(' ',2,' ')||left('TCB distribution (%)',22)
Cpu.5  =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',1Ø)||left('Step name',11),
        ||left('#Ind',7)||left('Clock',7)||left('Total',8),
        ||left('TCB',7)||left('SRB',7),
        ||left('I/O',8)||left('RCT',7)||left('HSP',6),
        ||left('Asid',6)||left('Indpt.',7) ,
        ||left('Avg ind.',9)||left('Dep.',7) ,
        ||left('srb',6)||left('Asid',5),
        ||left('Indep.',7)||left('Dep.en',7)||left('Preemt.',7)
Cpu.6   =left(' ',4,' ')||left('-',177,'-')
   "EXECIO * DISKW s3Øcp (STEM Cpu.)"
 Io.1  =left(' ',8,' '),
        ||'Enclave I/O Activity Report '||left(' ',15,' ')
 Io.2  ='  '
 Io.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
 Io.4  =left(' ',147,' ')||left('-',17,'-')||left(' ',2,' '),
        ||left('Independent enclave DASD I/O timing',35)||left('-',14,'-
')
 Io.5  =left('',81,' ')||left('-- Delays --',14)||left(' ',8,' '),
        ||left('Blocks',9)||left('Connect',7)||left(' ',2, ' '),
        ||left('Asid + dep.enclave',31)||left(' ',3Ø,' ') ,
        ||left('I/O distribution (%)',3Ø)||left('Average',8)
 Io.6  =left('OBS',4)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11)||left('#Ind',7),
        ||left('Clock',7)||left('ds enqueue',13)||left('allocation',13),
        ||left('transfer',1Ø)||left('time',8),
```

```
         ||left('connect',1Ø)||left('disconn.',1Ø)||left('pending',11),
         ||left('connect',1Ø)||left('disconn.',1Ø)||left('pending',9) ,
         ||left('conn',6)||left('discon',8)||left('pend',8),
         ||left('conn',9)||left('discon',11)||left('pend',8)
  Io.7  =left(' ',4,' ')||left('-',225,'-')
   "EXECIO * DISKW s3Øio (STEM Io.)"

Prf.1  =left(' ',8,' '),
        ||'Enclave performance report '||left(' ',15,' ')
Prf.2  ='  '
Prf.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
Prf.4  =left(' ',78,' ')||left("Transaction timings (msec)",28),
        ||left(' ',2,' ')||left('-',4,'-')||left(' ',2,' '),
        ||left('Service units (in K)',2Ø),
        ||left('-',4,'-')||left(' ',18,' '),
        ||left('Independent enclave',25)||left(' ',37,' '),
        ||left('Perf.flags',11)
Prf.5   =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('User id',9)||left('Job id',8),
        ||left('Job name',9)||left('Step name',11)||left('#Ind',7),
        ||left('Clock',12)||left('Active',11)||left('RS resident',14),
        ||left('Total',1Ø)||left('Tcb',11)||left('Srb',1Ø),
        ||left('I/O',1Ø)||left('Mso',9),
        ||left('Active tm',11)||left('% Active',1Ø)||left('Avg.tm',8),
        ||left('Tcb su(K)',1Ø)||left('%SRM su',9)||left('Avg.su',8),
        ||left('SClass',7)||left('pf1 pf2',9)
Prf.6   =left(' ',4,' ')||left('-',222,'-')
   "EXECIO * DISKW s3Øpr (STEM Prf.)"

Mse.1  =left('Multisystem Enclave remote system Report',5Ø)
Mse.2  ='  '
Mse.3  =left(' ',8,' ')||'Report produced on',
        ||left(' ',1,' ')||left(date(),11),
        ||left(' ',1,' ')||left('at ',3,' ')||left(time(),1Ø)
Mse.4   =left('OBS',4,)||left('Date',12)||left('Time',9),
        ||left('System name',13)||left('Cpu adj.factor',14),
        ||left('Dep. enc. CPU',16)||left('Indep.enc. CPU',16)
Mse.5   =left(' ',4,' ')||left('-',78,'-')
   "EXECIO * DISKW s3Øms (STEM Mse.)"
Xu.1 =  left('Enclave summary report',3Ø)
Xu.2 =  left(' ',1,' ')
   "EXECIO * DISKW s3Øse(stem Xu.)"
   'EXECIO *  DISKR SMF3Ø (STEM x. FINIS'
    Do i = 1 to x.Ø
    smf3Ørty = c2d(SUBSTR(x.i,2,1))           /* SMF record type   */
    smf3Østp = c2d(SUBSTR(x.i,19,2))          /* Record subtype    */
 IF smf3Ørty = '3Ø'  Then
    Do
```

```
  smf30dte = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)  /* Unpack SMF date   */
  smf30tme = smf(c2d(SUBSTR(x.i,3,4)))         /* Decode SMF time   */
  term     = c2d(SUBSTR(x.i,3,4))              /* Termination time  */
  smf30sid = SUBSTR(x.i,11,4)                  /* System ide.       */
  smf30wid = SUBSTR(x.i,15,4)                  /* Subsystem id      */
smf30sof = c2d(SUBSTR(x.i,21,4))      /* Offset to subsys section   */
smf30sln = c2d(SUBSTR(x.i,25,2))      /* Lenght to subsys section   */
smf30son = c2d(SUBSTR(x.i,27,2))      /* Number to subsys sections  */
smf30iof = c2d(SUBSTR(x.i,29,4))      /* Offset to job/sess.sec.    */
smf30iln = c2d(SUBSTR(x.i,33,2))      /* Length of job/sess.sec.    */
smf30ion = c2d(SUBSTR(x.i,35,2))      /* Number of job/sess.sec.    */
smf30uof = c2d(SUBSTR(x.i,37,4))      /* Offset to I/O section      */
smf30uln = c2d(SUBSTR(x.i,41,2))      /* Length of I/O section      */
smf30uon = c2d(SUBSTR(x.i,43,2))      /* Number of I/O sections     */
smf30tof = c2d(SUBSTR(x.i,45,4))      /* Offset to CC section       */
smf30tln = c2d(SUBSTR(x.i,49,2))      /* Length of CC section       */
smf30ton = c2d(SUBSTR(x.i,51,2))      /* Number of CC sections      */
smf30cof = c2d(SUBSTR(x.i,53,4))      /* Offset to CPU section      */
smf30cln = c2d(SUBSTR(x.i,57,2))      /* Length of CPU section      */
smf30con = c2d(SUBSTR(x.i,59,2))      /* Number of CPU sections     */
smf30aof = c2d(SUBSTR(x.i,61,4))      /* Offset to accn.section     */
smf30aln = c2d(SUBSTR(x.i,65,2))      /* Length of single accn.sec. */
smf30aon = c2d(SUBSTR(x.i,67,2))      /* Number of vl.text segments.*/
smf30rof = c2d(SUBSTR(x.i,69,4))      /* Offset to STOR section     */
smf30rln = c2d(SUBSTR(x.i,73,2))      /* Length of STOR section     */
smf30ron = c2d(SUBSTR(x.i,75,2))      /* Number of STOR sections    */
smf30pof = c2d(SUBSTR(x.i,77,4))      /* Offset to PERF. section    */
smf30pln = c2d(SUBSTR(x.i,81,2))      /* Length of PERF. section    */
smf30pon = c2d(SUBSTR(x.i,83,2))      /* Number of PERF. sections   */
smf30oof = c2d(SUBSTR(x.i,85,4))      /* Offset to operator sec.    */
smf30oln = c2d(SUBSTR(x.i,89,2))      /* Length of the operator sec */
smf30oon = c2d(SUBSTR(x.i,91,2))      /* Number of operator sec.    */
smf30eof = c2d(SUBSTR(x.i,93,4))      /* Offset to EXCP section     */
smf30eln = c2d(SUBSTR(x.i,97,2))      /* Length of EXCP section     */
smf30eon = c2d(SUBSTR(x.i,99,2))      /* Number of EXCP sections    */
smf30eor = c2d(SUBSTR(x.i,101,2))     /* Number of EXCP next sec.   */
smf30eos = c2d(SUBSTR(x.i,105,4))     /* Number of EXCP next sec.   */
smf30dro = c2d(SUBSTR(x.i,109,4))     /* Offset to APPC/MVS res.sec.*/
smf30drl = c2d(SUBSTR(x.i,113,2))     /* Length of APPC/MVS res.sec.*/
smf30drn = c2d(SUBSTR(x.i,115,2))     /* Number of APPC/MVS res.sec.*/
smf30aro = c2d(SUBSTR(x.i,117,4))     /* Offset to APPC/MVS cum.res */
smf30arl = c2d(SUBSTR(x.i,121,2))     /* Length of APPC/MVS cum.res */
smf30arn = c2d(SUBSTR(x.i,123,2))     /* Number of APPC/MVS cum.res */
smf30opo = c2d(SUBSTR(x.i,125,4))     /* Offset to USS section      */
smf30opl = c2d(SUBSTR(x.i,129,2))     /* Length of USS section      */
smf30opn = c2d(SUBSTR(x.i,131,2))     /* Number of USS sections     */
smf30opm = c2d(SUBSTR(x.i,133,4))     /* Next no. of USS sections   */
smf30udo = c2d(SUBSTR(x.i,137,4))     /* Offset to first usage data */
smf30udl = c2d(SUBSTR(x.i,141,2))     /* Length of each usage data  */
smf30udn = c2d(SUBSTR(x.i,143,2))     /* Number of usage data sec.  */
```

```
smf30uds = c2d(SUBSTR(x.i,145,4))     /* Next no. of usage data     */
smf30rmo = c2d(SUBSTR(x.i,149,4))     /* Offset to first ARM sec.   */
smf30rml = c2d(SUBSTR(x.i,153,2))     /* Length of ARM section      */
smf30rmn = c2d(SUBSTR(x.i,155,2))     /* Number of ARM sections     */
smf30rms = c2d(SUBSTR(x.i,157,4))     /* Number of ARM in next      */
smf30mof = c2d(SUBSTR(x.i,161,4))     /* Offset to MSys Enclave sec.*/
smf30mln = c2d(SUBSTR(x.i,165,2))     /* Length of MSys Enclave sec */
smf30mno = c2d(SUBSTR(x.i,167,2))     /* Number of MSys Enclave     */
smf30mos = c2d(SUBSTR(x.i,169,4))     /* Next no. of MSys Enclave   */
/*-------------------------------------------------------------------*/
/* Subsystem section                                                 */
/* This section contains general record and system identification   */
/* information that you can use to determine the level of info       */
/* on the rest of the record.                                        */
/* Triplet information: this section is located on the record using */
/* the following triplet fields, which are located in the            */
/* header/self-defining  section: offset smf30sof; length smf30sln; */
/* number smf30son - this field will always be 1   as this           */
/* section is on each of the Type 30 records that is generated.      */
/*-------------------------------------------------------------------*/
 IF smf30sof <> 0 AND smf30sln <> 0 Then do
    sof= smf30sof -3
    smf30typ = c2d(SUBSTR(x.i,sof,2))    /*  Sub-type identification*/
 Select
   When smf30typ = 1 then rec='job start'
   When smf30typ = 2 then rec='interval record'
   When smf30typ = 3 then rec='step termination delta'
   When smf30typ = 4 then rec='step total'
   When smf30typ = 5 then rec='job termination'
   When smf30typ = 6 then rec='system address space'
   otherwise nop
  End
    smf30rvn = SUBSTR(x.i,sof+2,4)            /*       Product level*/
  Select
    When smf30rvn = '05'  then ver='mvs/sp version 5'
    When smf30rvn = '04'X then ver='mvs/sp version 4'
    When smf30rvn = '03'X then ver='mvs/sp version 3'
    When smf30rvn = '04'X then ver='mvs/sp version 2'
    When smf30rvn = '01'X then ver='vs2'
    otherwise nop
   End
    smf30pnm = SUBSTR(x.i,sof+6,8)             /* Product name       */
    smf30osl = SUBSTR(x.i,sof+14,8)            /* MVS product level  */
    smf30syn = SUBSTR(x.i,sof+22,8)            /* Current system name*/
    smf30syp = SUBSTR(x.i,sof+30,8)            /* Sysplex name       */
 END
/*-------------------------------------------------------------------*/
/* Job / Session Identification Section                              */
/* This section contains general address space and user              */
/* information which can be used to identify the address space       */
```

```
/* that the data is being reported for or to merge                    */
/* this record with other records that are generated for this         */
/* address space.                                                     */
/* Triplet information: this section is located on the record using   */
/* the following triplet fields, which are located in the             */
/* header/self-defining  section: offset smf30iof; length smf30iln;   */
/* number smf30ion - this field will always be 1 because this         */
/* section is on each of the Type 30 records that are generated.      */
/*--------------------------------------------------------------------*/
 IF smf30iof <> 0 AND smf30iln <> 0 Then do
    iof = smf30iof -3
smf30jbn = SUBSTR(x.i,iof,8)                  /* Job or session name */
smf30pgm = SUBSTR(x.i,iof+8,8)                /* Program name        */
    rsd = SUBSTR(c2x(SUBSTR(x.i,iof+12,4)),3,5)
smf30stm = SUBSTR(x.i,iof+16,8)               /* Step name           */
smf30uif = SUBSTR(x.i,iof+24,8)               /* User identification */
smf30jnm = SUBSTR(x.i,iof+32,8)               /* JES job identifier  */
smf30stn = c2d(SUBSTR(x.i,iof+40,2))          /* Step number         */
smf30cls = c2d(SUBSTR(x.i,iof+42,1))          /* Job class           */
smf30pgn = c2d(SUBSTR(x.i,iof+44,2))          /* Performance group   */
                                              /* Valid only if WLM:  */
                                              /* compatibility mode  */
                                              /* Zero if WLM in goal */
                                              /* mode                */
smf30jpt = c2d(SUBSTR(x.i,iof+46,2))          /* JES input prty.     */
smf30ast = smf(c2d(SUBSTR(x.i,iof+48,4)))     /* Start of alloc event*/
   alloc =     c2d(SUBSTR(x.i,iof+48,4))      /* Start of allocation */
smf30pps = smf(c2d(SUBSTR(x.i,iof+52,4)))     /* Program start time  */
    load =     c2d(SUBSTR(x.i,iof+52,4))      /* Program fetch event */
smf30sit = smf(c2d(SUBSTR(x.i,iof+56,4)))     /* Init selected time  */
    init=     c2d(SUBSTR(x.i,iof+56,4))       /* Step initiate time  */
smf30std = SUBSTR(c2x(SUBSTR(x.i,iof+60,4)),3,5)  /*       Init date */
smf30rst = smf(c2d(SUBSTR(x.i,iof+64,4)))     /* Reader end time     */
    read =     c2d(SUBSTR(x.i,iof+64,4))      /* Read-in or logon tm.*/
smf30rsd = SUBSTR(c2x(SUBSTR(x.i,iof+68,4)),3,5)  /*  Reader date    */
smf30ret = smf(c2d(SUBSTR(x.i,iof+72,4)))     /* Reader end time     */
    rend=      c2d(SUBSTR(x.i,iof+72,4))      /* Unformatted ret     */
smf30red = SUBSTR(c2x(SUBSTR(x.i,iof+76,4)),3,5)  /*  Reader date    */
smf30usr = SUBSTR(x.i,iof+80,20)              /* Programmer's name   */
smf30grp = SUBSTR(x.i,iof+100,8)              /* RACF group id.      */
smf30rud = SUBSTR(x.i,iof+108,8)              /* RACF user id.       */
smf30tid = SUBSTR(x.i,iof+116,8)              /* RACF terminal id.   */
smf30tsn = SUBSTR(x.i,iof+124,8)              /* Terminal symb. name */
smf30psn = SUBSTR(x.i,iof+132,8)              /* Step name           */
smf30cl8 = SUBSTR(x.i,iof+140,8)              /* Job class           */
smf30iss = c2d(SUBSTR(x.i,iof+148,8))         /* Interval Start Time */
smf30iet = c2d(SUBSTR(x.i,iof+156,8))         /* Interval End Time   */
smf30ssn = c2d(SUBSTR(x.i,iof+164,4))         /* Sub-Step Number     */
smf30exn = SUBSTR(x.i,iof+168,16)             /*Exec'ed program name */
    ttp = SUBSTR(c2x(SUBSTR(x.i,iof+168,16)),15,2)
```

```
       SELECT
        when ttp = '4Ø' then pgmtype='MVS'
        otherwise pgmtype='USS'
       END
/* time calc */
    dsenqtm = smf(alloc-init)            /* ds enqueue delay         */
    aloctm  = smf(load-alloc)            /* allocation delay duration*/
    exectm  = smf(term-load)             /* execution duration       */
    elapstm = cross(term,init)           /* step elapsed duration    */
 END
/*-------------------------------------------------------------------*/
/* I/O activity section                                              */
/* This section contains the summary I/O information at the          */
/* address space level. This differs from the I/O information        */
/* in the EXCP sections of the record which present the data         */
/* at a DD name/device level.                                        */
/* Triplet information: this section is located on the record        */
/* using the following triplet fields, which are located in the      */
/* header/self-defining  section: offset smf30uof; length smf30uln ; */
/* number smf30uon - reports the number of I/O activity sections     */
/* on the current record. Because only one I/O activity section      */
/* can appear on the record, this field is 1 (if the                 */
/* section exists) or Ø (if it doesn t).                             */
/*-------------------------------------------  ----------------------*/
 IF smf30uof <> Ø AND smf30uln <> Ø Then do
    uof=  smf30uof -3
smf30inp = c2d(SUBSTR(x.i,uof,4))         /* Card reads in DD Data   */
smf30tep = c2d(SUBSTR(x.i,uof+4,4))       /* Total blocks transferred*/
smf30tpt = c2d(SUBSTR(x.i,uof+8,4))       /* No.of tputs for a TSO   */
smf30tgt = c2d(SUBSTR(x.i,uof+12,4))      /* No.of tgets for a TSO   */
smf30rdr = c2d(SUBSTR(x.i,uof+16,1))      /* Reader device class     */
smf30rdt = c2d(SUBSTR(x.i,uof+17,1))      /* Reader device type      */
smf30tcn = c2d(SUBSTR(x.i,uof+18,4))      /*Total device connect time*/
     tcn = format(smf30tcn*128E-6,6,3)
smf30dcf = c2d(SUBSTR(x.i,uof+22,4))      /* Device connect flags    */
smf30trr = c2d(SUBSTR(x.i,uof+28,4))      /* Total asid re-read count*/
smf30aic = c2d(SUBSTR(x.i,uof+32,4))      /* DASD I/O connect time + */
     aic = format(smf30aic*128E-6,6,3)    /* dependent enclaves      */
smf30aid = c2d(SUBSTR(x.i,uof+36,4))      /* DASD I/O disconnect time*/
     aid = format(smf30aid*128E-6,6,3)    /* + dependent enclaves    */
smf30aiw = c2d(SUBSTR(x.i,uof+40,4))      /* DASD I/O pending time + */
     aiw = format(smf30aiw*128E-6,6,3)    /* CU queue time + dep.enc */
smf30ais = c2d(SUBSTR(x.i,uof+44,4))      /* DASD I/O start count +  */
     ais = format(smf30ais*128E-6,6,3)    /* dependent enclaves      */
/*-------------------------------------------------------------------*/
/* Independent enclaves DASD fields:                                 */
/* smf30eic,  smf30eid, smf30eiw, smf30eis                           */
/*-------------------------------------------------------------------*/
smf30eic = c2d(SUBSTR(x.i,uof+48,4))      /* DASD I/O connect time   */
     eic = format(smf30eic*128E-6,6,3)    /* for independent encl.   */
```

```
smf30eid = c2d(SUBSTR(x.i,uof+52,4))      /* DASD I/O disconnect time*/
     eid = format(smf30eid*128E-6,6,3)    /* for independent encl.   */
smf30eiw = c2d(SUBSTR(x.i,uof+56,4))      /* DASD I/O discon. + CU   */
     eiw = format(smf30eiw*128E-6,6,3)    /* queue time for ind.encl.*/
smf30eis = c2d(SUBSTR(x.i,uof+56,4))      /* DASD I/O start subchan. */
     eis = format(smf30eis*128E-6,6,3)    /* for independent enclave */
totalcon = smf30aic + smf30eic            /* Total conn. time: asid +*/
     con = format(totalcon*128E-6,6,3)    /* dep.enc + indep.enc.    */
totaldis = smf30aid + smf30eid
     dis = format(totaldis*128E-6,6,3)
totalpen = smf30aiw + smf30eiw
     pen = format(totalpen*128E-6,6,3)
totalsch = smf30ais + smf30eis
     sch = format(totalsch*128E-6,6,3)
/* Independent enclaves DASD distribution */
Select
  when smf30eic > 0 then
  d1 = format(smf30eic/totalcon*100,5,2)
  otherwise d1 = 'n.a'
End
Select
  when smf30eid > 0 then
  d2 = format(smf30eid/totaldis*100,5,2)
  otherwise d2 = 'n.a'
End
Select
  when smf30eiw > 0 then
  d3 = format(smf30eiw/totalpen*100,5,2)
  otherwise d3 = 'n.a'
End
Select
  when smf30eis > 0 then
  d4 = format(smf30eis/totalsch*100,5,2)
  otherwise d4 = 'n.a'
End
  END
/*-------------------------------------------------------------------*/
/* Completion code section                                           */
/* This section contains the completion information for the step     */
/* on the Step Termination record (Subtype-4) and for the job        */
/* on the Job Termination record (Subtype-5).                        */
/* This section does not appear on the Job Initialization            */
/* (Subtype-1) or Interval(Subtype-2 and 3) records.                 */
/* Note: the smf30sti field also contains some general record        */
/* indicator flags which are not necessarily completion in           */
/* nature. The system may fail a step or job                         */
/* even if the return code is zero. This could happen, for           */
/* example, as a result of specifying CATLG_ERR FAILJOB(YES)         */
/* and incurring that type of post execution error. (A return        */
/* code is generated by the application program and                  */
```

```
/* is never changed by the operating system.) A user can deduce      */
/* that a step failed due to a post execution error if bit           */
/* smf30sye in the two-byte smf30sti field in the smf30 subtype 4    */
/* record is on.                                                     */
/* Triplet information: this section is located on the record        */
/* using the following triplet fields, which are located in the      */
/* header/self-defining section: offset smf30tof; length smf30tln;   */
/* number smf30ton - reports the number of completion sections       */
/* on the current record. Because only one completion section        */
/* can appear on the record, this field is 1 (if the section exists) */
/* or 0 (if it doesn't).                                             */
/*-------------------------------------------------------------------*/
 IF smf30tof <> 0 AND smf30tln <> 0 Then do
    tof=smf30tof -3
smf30scc = c2d(SUBSTR(x.i,tof,2))            /* Step completion code1*/
smf30sti = x2b(c2d(SUBSTR(x.i,tof+2,1)))     /* Step term.indicator  */
smf30sab = substr(smf30sti,1,1)              /* system abend         */
smf30ujv = substr(smf30sti,2,1)              /* iefujv    cancelled  */
smf30uji = substr(smf30sti,3,1)              /* iefuji    cancelled  */
smf30usi = substr(smf30sti,4,1)              /* iefusi    cancelled  */
smf30trt = substr(smf30sti,5,1)              /* iefactrt  cancelled  */
smf30srs = substr(smf30sti,6,1)              /* step restarted       */
smf30abd = substr(smf30sti,7,1)              /* 0=ok; 1=abend        */
smf30flh = substr(smf30sti,8,1)              /* 0=ok; 1=jcl error    */
smf30stt = x2b(c2d(SUBSTR(x.i,tof+3,1)))     /* Step term.indicator2 */
smf30exf = substr(smf30stt,1,1)              /* excp counts incorrect*/
smf30isk = substr(smf30stt,2,1)              /* interval recordnot ok*/
smf30nex = substr(smf30stt,3,1)              /* excp sec.not merged  */
smf30sye = substr(smf30stt,4,1)              /* post exec error      */
smf30ere = substr(smf30stt,5,1)              /* open mvs end request */
smf30cde = substr(smf30stt,6,1)              /* jcl cond= condition  */
 END
/*-------------------------------------------------------------------*/
/* Processor Section (CPU accounting segment)                        */
/* This section contains various Processor times for the address     */
/* space for the period that the record represents.                  */
/* Triplet information: this section is located on the               */
/* record using the following triplet fields, which are located in   */
/* the header/self-defining sec.: offset smf30cof; length smf30cln;  */
/* number smf30con - reports the number of processor accounting      */
/* sections on the current record. Because only one processor        */
/* accounting section can appear on the record, this field is        */
/* 1 (if the section exists) or 0 (if it doesn t).                   */
/*-------------------------------------------------------------------*/
 IF smf30cof <> 0 AND smf30cln <> 0 Then do
    cof=smf30cof -3
smf30pty = c2d(SUBSTR(x.i,cof,2))        /* Dispatching  priority    */
smf30tfl = x2b(c2d(SUBSTR(x.i,cof+2,1))) /* timer flags - part 1     */
smf30tff = x2b(c2d(SUBSTR(x.i,cof+3,1))) /* timer flags - part 2     */
smf30tfo = substr(smf30tfl,1,1)          /* timer flags are used:y/n */
```

```
smf30ctf = substr(smf30tfl,2,1)          /* smf30cpt invalid         */
smf30csf = substr(smf30tfl,3,1)          /* smf30cps invalid         */
smf30vuf = substr(smf30tfl,4,1)          /* smf30jvu invalid         */
smf30vaf = substr(smf30tfl,5,1)          /* smf30jva invalid         */
smf30isf = substr(smf30tfl,6,1)          /* smf30isb invalid         */
smf30icf = substr(smf30tfl,7,1)          /* smf30icu invalid         */
smf30ivf = substr(smf30tfl,8,1)          /* smf30ivu invalid         */
smf30iaf = substr(smf30tff,1,1)          /* smf30iva invalid         */
smf30iif = substr(smf30tff,2,1)          /* smf30iip invalid         */
smf30hpf = substr(smf30tff,3,1)          /* smf30hpt invalid         */
smf30rcf = substr(smf30tff,4,1)          /* smf30rct invalid         */
smf30asf = substr(smf30tff,5,1)          /* smf30asr invalid         */
smf30enf = substr(smf30tff,6,1)          /* smf30enc invalid         */
smf30def = substr(smf30tff,7,1)          /* smf30det invalid         */
smf30cpt = c2d(SUBSTR(x.i,cof+4,4))      /* Step cpu time under TCB  */
smf30cps = c2d(SUBSTR(x.i,cof+8,4))      /* Step cpu time under SRB  */
smf30icu = c2d(SUBSTR(x.i,cof+12,4))     /* init cpu time (tcb)      */
smf30isb = c2d(SUBSTR(x.i,cof+16,4))     /* init cpu time (srb)      */
smf30jvu = c2d(SUBSTR(x.i,cof+20,4))     /* step vector usage time   */
smf30ivu = c2d(SUBSTR(x.i,cof+24,4))     /* init vector usage time   */
smf30jva = c2d(SUBSTR(x.i,cof+28,4))     /* step vector aff.usage tm.*/
smf30iva = c2d(SUBSTR(x.i,cof+32,4))     /* init vector aff.usage tm.*/
smf30ist = c2d(SUBSTR(x.i,cof+36,4))     /* interval start time      */
smf30idt = c2d(SUBSTR(x.i,cof+20,4))     /* interval start date      */
smf30iip = c2d(SUBSTR(x.i,cof+44,4))     /* I/O interrupts time      */
smf30rct = c2d(SUBSTR(x.i,cof+48,4))     /* Region control task time */
smf30hpt = c2d(SUBSTR(x.i,cof+52,4))     /* Hiperspace transfer time */
smf30csc = c2d(SUBSTR(x.i,cof+56,4))     /* Crypto service count     */
smf30dmi = c2d(SUBSTR(x.i,cof+60,4))     /* # pages moved:ADMF/write */
smf30dmo = c2d(SUBSTR(x.i,cof+64,4))     /* # pages moved:ADMF/read  */
smf30asr = c2d(SUBSTR(x.i,cof+68,4))     /* Preemptable/Client SRB   */
smf30enc = c2d(SUBSTR(x.i,cof+72,4))     /* Independent encl.CPU time*/
smf30det = c2d(SUBSTR(x.i,cof+76,4))     /* Dependent encl. CPU time */
smf30cep = c2d(SUBSTR(x.i,cof+80,4))     /* CPU time while enqueue   */
smf30cep = smf30cep * 1024E-6            /* CPU time while enqueue   */
initcpu  = smf30icu + ,                  /* init cpu time (tcb)      */
           smf30isb                      /* init cpu time (srb)      */
vectorcpu= smf30jvu + ,                  /* step vector usage time   */
           smf30ivu + ,                  /* init vector usage time   */
           smf30jva + ,                  /* step vector aff.usage tm.*/
           smf30iva                      /* init vector aff.usage tm.*/
/*-------------------------------------------------------------------*/
/* note: smf30cpt includes:       -                                  */
/*       smf30enc (cpu time used by the enclave for the step or job) */
/*       smf30det (dependent enclave cpu time for the stepor job)    */
/*       smf30asr (additional CPU time accumulated by the            */
/*                 preemptable SRBs and client SRBs for this job)    */
/*-------------------------------------------------------------------*/
totcpu= smf30cpt + smf30cps + smf30iip + smf30rct + smf30hpt,
      + smf30cep + initcpu  + vectorcpu
```

```
/*-------------------------------------------------------------------*/
/* Get ASID only CPU TCB (ie exclude enclave & pre-emt. SRB timing  */
/*-------------------------------------------------------------------*/
asidtcb = smf30cpt - (smf30enc + smf30det + smf30asr)
/*-------------------------------------------------------------------*/
/* Independent encl.CPU time / Step CPU time under TCB ratio         */
/*-------------------------------------------------------------------*/
select
  when smf30enc > 0 then ,
  p1 = format((smf30enc/smf30cpt)*100,5,2)
  otherwise              p1 = 'n.a'
end
/*-------------------------------------------------------------------*/
/* Dependent encl. CPU time  / Step CPU time under TCB ratio         */
/*-------------------------------------------------------------------*/
select
  when smf30det > 0 &  smf30cpt  > 0 then ,
  p2 = format((smf30det/smf30cpt)*100,5,2)
  otherwise              p2 = '0'
end
/*-------------------------------------------------------------------*/
/* Preemptable/Client CPU / Step CPU time under TCB ratio            */
/*-------------------------------------------------------------------*/
select
  when smf30asr > 0 & smf30cpt  > 0 then ,
  p3 = format((smf30asr/smf30cpt)*100,5,2)
  otherwise              p3 = '0'
end
/*-------------------------------------------------------------------*/
/* ASID only CPU TCB / Step CPU time under TCB ratio                 */
/*-------------------------------------------------------------------*/
select
  when smf30cpt > 0 then ,
  p4 = format((asidtcb/smf30cpt)*100,5,2)
  otherwise              p4 = 'n.a'
end
 END
/*-------------------------------------------------------------------*/
/* Storage and Paging Section                                        */
/* This section contains the statistics on the use of different      */
/* kinds of storage by the address space and the different kinds     */
/* of paging activity for the address space.                         */
/* Triplet information: this section is located on the record using  */
/* the following triplet fields, which are located in the            */
/* header/self-defining section:offset smf30rof; length smf30rln;    */
/* number smf30ron - reports the number of storage and paging        */
/* sections on the current record. Because only one storage and      */
/* paging section can appear on the record, this field is 1          */
/* (if the section exists) or 0 (if it doesn t).                     */
/*-------------------------------------------------------------------*/
```

```
 IF smf30rof <> 0 AND smf30rln <> 0 Then do
    rof=smf30rof -3
smf30sfl = c2d(SUBSTR(x.i,rof+2,1))     /* Storage flags            */
smf30spk = c2d(SUBSTR(x.i,rof+3,1))     /* Storage protect key      */
smf30prv = c2d(SUBSTR(x.i,rof+4,2))     /* Getmain size from bottom */
smf30sys = c2d(SUBSTR(x.i,rof+6,2))     /* Getmain size from top    */
smf30pgi = c2d(SUBSTR(x.i,rof+8,4))     /* unblk. pages: in from aux*/
smf30pgo = c2d(SUBSTR(x.i,rof+12,4))    /* unblk. pages: out to aux */
smf30cpm = c2d(SUBSTR(x.i,rof+16,4))    /* read data from expanded  */
smf30nsw = c2d(SUBSTR(x.i,rof+20,4))    /* no.ASID swap sequence    */
smf30psi = c2d(SUBSTR(x.i,rof+24,4))    /* pages swapped:in from aux*/
smf30pso = c2d(SUBSTR(x.i,rof+28,4))    /* pages swapped: out to aux*/
smf30vpi = c2d(SUBSTR(x.i,rof+32,4))    /* number of vio page ins   */
smf30vpo = c2d(SUBSTR(x.i,rof+36,4))    /* number of vio page outs  */
smf30vpr = c2d(SUBSTR(x.i,rof+40,4))    /* number of vio reclaims   */
smf30cpi = c2d(SUBSTR(x.i,rof+44,4))    /* common area page-ins     */
smf30hpi = c2d(SUBSTR(x.i,rof+48,4))    /* hiper page ins           */
smf30lpi = c2d(SUBSTR(x.i,rof+52,4))    /* lpa area page-ins        */
smf30hpo = c2d(SUBSTR(x.i,rof+56,4))    /* hiper page outs          */
smf30pst = c2d(SUBSTR(x.i,rof+60,4))    /* pages stolen away        */
smf30psc = c2d(SUBSTR(x.i,rof+64,8))    /* no.of cpu page seconds   */
smf30rgb = c2d(SUBSTR(x.i,rof+72,4))    /* private area size <16M   */
smf30erg = c2d(SUBSTR(x.i,rof+76,4))    /* private area size >16M   */
smf30arb = c2d(SUBSTR(x.i,rof+80,4))    /* max vir.lsqa and swa <16M*/
smf30ear = c2d(SUBSTR(x.i,rof+84,4))    /* max vir.lsqa and swa >16M*/
smf30urb = c2d(SUBSTR(x.i,rof+88,4))    /* max vir.user subpools<16M*/
smf30eur = c2d(SUBSTR(x.i,rof+92,4))    /* max vir.user subpools>16M*/
smf30rgn = c2d(SUBSTR(x.i,rof+96,4))    /* region size              */
smf30dsv = c2d(SUBSTR(x.i,rof+100,4))   /* data space storage used  */
smf30pie = c2d(SUBSTR(x.i,rof+104,4))   /* unblk. pages: in from es */
smf30poe = c2d(SUBSTR(x.i,rof+108,4))   /* unblk. pages: out to es. */
smf30bia = c2d(SUBSTR(x.i,rof+112,4))   /* blocks paged: in from aux*/
smf30boa = c2d(SUBSTR(x.i,rof+116,4))   /* blocks paged: out to aux */
smf30bie = c2d(SUBSTR(x.i,rof+120,4))   /* blocked pages: in from es*/
smf30boe = c2d(SUBSTR(x.i,rof+124,4))   /* blocked pages: out to es.*/
smf30kia = c2d(SUBSTR(x.i,rof+128,4))   /* blocks pages: in from aux*/
smf30koa = c2d(SUBSTR(x.i,rof+132,4))   /* blocks pages: out to aux */
smf30psf = c2d(SUBSTR(x.i,rof+144,8))   /* cpu page seconds         */
smf30pai = c2d(SUBSTR(x.i,rof+152,4))   /* shared pages paged in AUX*/
smf30pei = c2d(SUBSTR(x.i,rof+156,4))   /* shared pages paged in EXP*/
smf30ers = c2d(SUBSTR(x.i,rof+160,8))   /* EXP residency time       */
smf30mem = c2d(SUBSTR(x.i,rof+168,8))   /* MEMLIMIT value in MB     */
smf30mls = c2d(SUBSTR(x.i,rof+170,1))   /* MEMLIMIT source          */
smf30psc = smf30psc * 1024E-6           /* % 1000000 for sec. value */
 page = smf30pgi + smf30pgo + smf30bia + smf30boa + smf30kia ,
       +smf30koa + smf30pie + smf30poe + smf30bie + smf30boe ,
       +smf30cpi + smf30lpi + smf30hpi + smf30hpo

 swap  = smf30psi + smf30pso               /* total swap count      */
 vio   = smf30vpi + smf30vpo + smf30vpr    /* total vio count       */
```

```
 auxin = smf30pgi + smf30bia + smf30kia    /* page in from aux.     */
 auxout= smf30pgo + smf30boa + smf30koa    /* page out to aux.      */
 esin  = smf30pie + smf30bie               /* page in from es.      */
 esout = smf30poe + smf30boe               /* page out to es.       */
 END
/*-------------------------------------------------------------------*/
/* Performance Section                                               */
/* This section contains the SRM service units used by the           */
/* address space for the period being reported on. For more          */
/* information on SRM service units, see z/OS "MVS Initialization    */
/* and Tuning Guide"                                                 */
/* Triplet information: this section is located on the record        */
/* using the following triplet fields, which are located in the      */
/* header/self-defining section: offset smf30pof; length smf30pln;   */
/* number smf30pon - reports the no. of performance sections on the */
/* current record. Because only one performance section can appear   */
/* on the record, this field = 1 (if the section exists) or Ø (if it*/
/* doesn t).                                                         */
/*-------------------------------------------------------------------*/
 IF smf30pof <> Ø AND smf30pln <> Ø Then do
    pof=smf30pof -3
smf30srv = c2d(SUBSTR(x.i,pof,4))       /* Total service units      */
smf30csu = c2d(SUBSTR(x.i,pof+4,4))     /* CPU service units        */
smf30srb = c2d(SUBSTR(x.i,pof+8,4))     /* SRB service units        */
smf30io  = c2d(SUBSTR(x.i,pof+12,4))    /* I/O service units        */
smf30mso = c2d(SUBSTR(x.i,pof+16,4))    /* MSO service units        */
smf30tat = c2d(SUBSTR(x.i,pof+2Ø,4))    /* SRM transaction time     */
smf30res = c2d(SUBSTR(x.i,pof+28,4))    /* SRM residency time       */
smf30trs = c2d(SUBSTR(x.i,pof+32,4))    /* SRM transactions count   */
smf30wlm = SUBSTR(x.i,pof+36,8)         /* Workload Name            */
smf30scn = SUBSTR(x.i,pof+44,8)         /* Service Class Name       */
smf30grn = SUBSTR(x.i,pof+52,8)         /* Resource Group Name      */
smf30rcn = SUBSTR(x.i,pof+6Ø,8)         /* Reporting Class Name     */
smf30eta = c2d(SUBSTR(x.i,pof+68,4))    /* Independent Enclave time */
smf30esu = c2d(SUBSTR(x.i,pof+72,4))    /* Independent Enclave cpusu*/
smf30etc = c2d(SUBSTR(x.i,pof+76,4))    /* Independent Enclave count*/
smf30pfl =   SUBSTR(x.i,pof+8Ø,16)      /* Scheduling env. name     */
smf30jqt = c2d(SUBSTR(x.i,pof+96,4))    /* JCL conversion time      */
smf30rqt = c2d(SUBSTR(x.i,pof+1ØØ,4))   /* hold time - affinity     */
smf30hqt = c2d(SUBSTR(x.i,pof+1Ø4,4))   /* hold time - no affinity  */
smf30sqt = c2d(SUBSTR(x.i,pof+1Ø8,4))   /* execution eligible time  */
smf30pf1 = c2x(SUBSTR(x.i,pof+112,1))   /* Performance flags - 1    */
     SELECT
      when smf30pf1 = 8Ø then note1 ='Reset before initiation          '
      when smf30pf1 = 4Ø then note1 ='Reset after  initiation          '
      when smf30pf1 = 2Ø then note1 ='Immediate initiation             '
      when smf30pf1 = 1Ø then note1 ='Job has been restarted           '
      when smf30pf1 = Ø8 then note1 ='Remote Sys. Data incomplete      '
      when smf30pf1 = Ø4 then note1 ='Job executing in a WLM batch init'
      when smf30pf1 = Ø2 then note1 ='Serv. class CPU-critical         '
```

```
      when smf30pf1 = 01 then note1 ='Serv. class stg-critical         '
      otherwise               note1 ='OK'
     END
smf30pf2 = c2x(SUBSTR(x.i,pof+113,1))   /* Performance flags - 2    */
     SELECT
      when smf30pf2 = 80 then note2 ='AS designated stg-critical   '
     when smf30pf2 = 40 then note2 ='AS cannot be managed to trxn goals'
     when smf30pf2 = 20 then note2 ='AS is currently CPU-protected     '
     when smf30pf2 = 10 then note2 ='AS is currently stg-protected     '
      otherwise               note2 ='OK'
     END
smf30jpn = SUBSTR(x.i,pof+116,8)        /*    Subsys collection name */
                                        /*  % 1000000 for sec. value */
    active  = format(smf30tat*1024E-6,7,2)  /* SRM  active time      */
    eactive = format(smf30eta*1024E-6,7,2)  /* Enc. active time      */
    reside  = format(smf30res*1024E-6,7,2) /*resident in real storage*/
    sqt1   = smf30sqt*1024E-6
smf30tot = smf30srv                      /* Total service units      */
smf30srv = format(smf30srv/1024,9,2)
smf30csu1= format(smf30csu/1024,9,2)
smf30srb = format(smf30srb/1024,9,2)
smf30io  = format(smf30io/1024,9,2)
smf30mso = format(smf30mso/1024,9,2)
smf30esu1= format(smf30esu/1024,9,2)
/*-------------------------------------------------------------------*/
/* CPU service unit ratio: Independent enclave vs. ASID CPU su.      */
/*-------------------------------------------------------------------*/
Select
  when smf30esu < smf30csu then ,
  e1 = format(smf30esu/smf30csu*100,5,2)
  otherwise e1 = 'n.a'
End
/*-------------------------------------------------------------------*/
/* Transaction active time ratio: Independent enc. vs. SRM           */
/*-------------------------------------------------------------------*/
Select
  when smf30tat > smf30eta then ,
  e2 = format(eactive/active*100,5,3)
  otherwise e2 = 'n.a'
End
/*-------------------------------------------------------------------*/
/* Avg. Independent enclave transaction active time                  */
/*-------------------------------------------------------------------*/
Select
  when smf30tat > smf30eta then e4 = format(eactive/smf30etc,7,3)
  otherwise e4 = 'n.a'
End
/*-------------------------------------------------------------------*/
/* Avg. Independent enclave CPU service units                        */
/*-------------------------------------------------------------------*/
```

```
Select
  when smf30esu < smf30csu  then ,
  e3 = format(smf30esu/smf30etc,7,2)
  otherwise e3 = 'n.a'
End
/*-------------------------------------------------------------------*/
/*      Performance Report                                           */
/*-------------------------------------------------------------------*/
  c30pr.1 = right(i,3,'0')    right(Date('N',smf30dte,'J'),11),
          left(smf30tme,8)    right(smf30rud,8)  left(smf30jnm,8),
          left(smf30jbn,8)    left(smf30stm,8)   right(smf30etc,5),
          left(elapstm,8)     right(active,12)   right(reside,12),
          right(smf30srv,9)   right(smf30csu1,9) right(smf30srb,9),
          right(smf30io,9)    right(smf30mso,9)                  ,
          right(eactive,12)   right(e2,8)        right(e4,8)     ,
          right(smf30esu1,9)  right(e1,8)        right(e3,9)     ,
          left(smf30scn,6)    left(smf30pf1,3)   left(smf30pf2,3)
  "EXECIO 1 DISKW s30pr(stem c30pr.)"

   END
/*-------------------------------------------------------------------*/
/* Execute Channel Program (EXCP) Section                            */
/* This section contains the I/O information for a specific          */
/* DD Name/Device address pair for the address space. There          */
/* can be multiple EXCP sections for a given address space.          */
/* Triplet information: this section is located on the record using */
/* the following triplet fields, which are located in the            */
/* header/self-defining section: offset smf30eof; length smf30eln;  */
/* number smf30eon - reports the number of excp sections on the      */
/* current record.                                                   */
/* This section also has additional control fields in the            */
/* header/self-defining section:                                     */
/* - smf30eos reports the number of excp sections for the current    */
/* period on subsequent type 30 records. These are known as the      */
/* chained type 30 records.                                          */
/* - smf30eor also reports this same information but is only a       */
/* 2-byte field which can overflow so smf30eos is the preferred      */
/* field for processing this data.                                   */
/*-------------------------------------------------------------------*/
 IF smf30eof <> 0 AND smf30eln <> 0 AND smf30eon <> 0 Then do
    eof=smf30eof -3
    totexcp.i = 0
  do k = 0 to smf30eon -1
    incr = (eof + (k*smf30eln))
    ddn.k = SUBSTR(x.i,incr+4,8)
    blk.k = c2d(SUBSTR(x.i,incr+12,4))
    totexcp.i =totexcp.i + blk.k
   end
   end
  END
```

```
/*-------------------------------------------------------------------*/
/* USS section                                                       */
/* Reports on Open MVS processes' use of Open MVS services           */
/*-------------------------------------------------------------------*/
 IF smf30opo <> 0 AND sm30opl <> 0 and smf30opn <> 0 then do
    totio.i = 0
    totcal.i = 0
  do w = 0 to smf30opn -1
    opo1   = (smf30opo + (w*smf30opl)) - 3
smf30opi   = c2d(SUBSTR(x.i,opo1,4))     /* Proc. ID                 */
smf30opp.w = c2d(SUBSTR(x.i,opo1+72,4))  /* Parent process ID no.    */
smf30opg   = c2d(SUBSTR(x.i,opo1+4,4))   /* Process Group ID         */
smf30oui   = c2d(SUBSTR(x.i,opo1+8,4))   /* Process User ID          */
smf30oug   = c2d(SUBSTR(x.i,opo1+12,4))  /* Process User Group ID    */
smf30osi   = c2d(SUBSTR(x.i,opo1+16,4))  /* Process Session ID       */
smf30osc   = c2d(SUBSTR(x.i,opo1+20,4))  /* Number of USS Syscalls   */
smf30ost   = c2d(SUBSTR(x.i,opo1+24,4))  /* Total CPU time           */
smf30odr.w = c2d(SUBSTR(x.i,opo1+28,4))  /* No.dir. I/O blocks read p*/
smf30ofr.w = c2d(SUBSTR(x.i,opo1+32,4))  /* No.dir. I/O blocks read s*/
smf30ofw.w = c2d(SUBSTR(x.i,opo1+36,4))  /* No.dir. I/O blocks wrt. s*/
smf30opr.w = c2d(SUBSTR(x.i,opo1+40,4))  /* No.dir. I/O blocks read i*/
smf30opw.w = c2d(SUBSTR(x.i,opo1+44,4))  /* No.dir. I/O blocks wrt. i*/
smf30osr.w = c2d(SUBSTR(x.i,opo1+48,4))  /* No.dir. I/O blocks read c*/
smf30osw.w = c2d(SUBSTR(x.i,opo1+52,4))  /* No.dir. I/O blocks wrt. c*/
smf30okr.w = c2d(SUBSTR(x.i,opo1+76,4))  /* I/O blocks read -Remote  */
smf30okw.w = c2d(SUBSTR(x.i,opo1+80,4))  /* I/O blocks wrt. -Remote  */
smf30oll.w = c2d(SUBSTR(x.i,opo1+56,4))  /* Look calls - path log.   */
smf30olp.w = c2d(SUBSTR(x.i,opo1+60,4))  /* Look calls - path phys.  */
smf30ogl.w = c2d(SUBSTR(x.i,opo1+64,4))  /* Gen. calls - path log.   */
smf30ogp.w = c2d(SUBSTR(x.i,opo1+68,4))  /* Gen. calls - path phys.  */
smf30oms.w = c2d(SUBSTR(x.i,opo1+84,4))  /* Message queues BY. sent  */
smf30omr.w = c2d(SUBSTR(x.i,opo1+88,4))  /* Message queues BY. rec.  */
smf30osy.w = c2d(SUBSTR(x.i,opo1+92,4))  /* No.sync() function calls */
   time = SUBSTR(smftime,1,7)
totio.i = totio.i + smf30odr.w + smf30ofr.w + smf30ofw.w ,
        + smf30opr.w + smf30opw.w ,
        + smf30osr.w + smf30osw.w ,
        + smf30okr.w + smf30okw.w
totcal.i= totcal.i   + smf30oll.w + smf30olp.w + ,
          smf30ogl.w + smf30ogp.w + smf30osy.w
 END
/*-------------------------------------------------------------------*/
/* Multisystem Enclave Remote System Data Section                    */
/* This section contains remote system data for each system          */
/* that executed work under a multisystem enclave.                   */
/* Triplet information: this section is located on the record using */
/* the following triplet fields, which are located in the            */
/* header/self-defining section: offset smf30mof; length smf30mln;  */
/* number smf30mno                                                   */
/*-------------------------------------------------------------------*/
```

```
 IF  smf30mno <> 0 then do
       do s = 0 to smf30nmo -1
   meoff  = (smf30mof + (s*smf30mln)) - 3
smf30mrs  =   SUBSTR(x.i,meoff,8)       /* System name
*/
smf30mra  = c2d(SUBSTR(x.i,meoff+8+4))
                                  /* Adjustment factor for CPU rate */
                                  /* - no. of sixteenths of one CPU */
                                  /* microsecond per CPU serv. unit */
smf30mrd  = c2d(SUBSTR(x.i,meoff+12,4))
                                   /* CPU Time used by dep. enclaves */
smf30mrd  = smf30mrd*0.01          /* that executed on named system  */
                                   /* in 0.01 of a second.           */
smf30mri  = c2d(SUBSTR(x.i,meoff+16,4)) /* CPU Time used by indep.    */
smf30mri  = smf30mri*0.01          /* enclaves that executed on named*/
                                   /* system in 0.01 of a second     */
smf30mr = smf30mrd + smf30mri
mrpct   = format((smf30mr/smf30cpt)*100,5,2)
mrp1    = format(smf30mr/smf30mno,5,3)
End
/*--------------------------------------------------------------------*/
/*   Multisystem Enclave Remote System Report                         */
/*--------------------------------------------------------------------*/
m30rs = right(i,3,'0')         right(Date('N',smf30dte,'J'),11)    ,
        left(smf30tme,8)       left(smf30mrs,8)  right(smf30mra,6) ,
        right(smf30mrd,6)      right(smf30mri,6)
  PUSH m30rs
      "EXECIO 1 DISKW s30ms"
  end
 END
/*--------------------------------------------------------------------*/
/* Avg. CPU TCB per independent enclave                               */
/*--------------------------------------------------------------------*/
Select
 when smf30enc > 0 then do
    aiec= (smf30enc/smf30etc)*0.01
    avgincpu = format(aiec,5,3)
    end
 otherwise avgincpu = '-'
End
/*--------------------------------------------------------------------*/
/*      Enclave CPU report                                            */
/*--------------------------------------------------------------------*/
  c30cp = right(i,3,'0')          right(Date('N',smf30dte,'J'),11),
          left(smf30tme,8)        right(smf30rud,8)  left(smf30jnm,8),
          left(smf30jbn,8)        left(smf30stm,8)   right(smf30etc,5),
          left(elapstm,8) right(totcpu*0.01,7)  right(smf30cpt*0.01,6) ,
          right(smf30cps*0.01,6)  right(smf30iip*0.01,6) ,
          right(smf30rct*0.01,6)  right(smf30hpt*0.01,6) ,
          right(asidtcb*0.01,6)   right(smf30enc*0.01,6) ,
```

```
          right(avgincpu,6)       right(smf30det*0.01,6) ,
          right(smf30asr*0.01,6)  right(p4,6)            ,
          right(p1,6)             right(p2,6) right(p3,6)
  PUSH c30cp
    "EXECIO 1 DISKW s30cp"
/* Independent enclaves DASD averages */
Select
  when smf30eic > 0 & smf30etc > 0 then
  da1 = format(eic/smf30etc,8,6)
  otherwise da1 = 'n.a'
End
Select
  when smf30eid > 0 & smf30etc > 0 then
  da2 = format(eid/smf30etc,8,6)
  otherwise da2 = 'n.a'
End
Select
  when smf30eiw > 0 & smf30etc > 0 then
  da3 = format(eiw/smf30etc,8,6)
  otherwise da3 = 'n.a'
End
Select
  when smf30eis > 0 & smf30etc > 0 then
  da4 = format(eis/smf30etc,8,6)
  otherwise da4 = 'n.a'
End
/*-------------------------------------------------------------------*/
/*      I/O Report                                                   */
/*-------------------------------------------------------------------*/
  c30io = right(i,3,'0')      right(Date('N',smf30dte,'J'),11),
          left(smf30tme,8)    right(smf30rud,8)  left(smf30jnm,8),
          left(smf30jbn,8)    left(smf30stm,8)   right(smf30etc,5),
          left(elapstm,8)     right(dsenqtm,12)  right(aloctm,12),
          right(smf30tep,8)   right(tcn,7)    ,
          right(aic,9)        right(aid,9)       right(aiw,9),
          right(eic,9)        right(eid,9)       right(eiw,9),
          right(d1,6)         right(d2,6)        right(d3,6) ,
          right(da1,9)        right(da2,9)       right(da3,9)
 PUSH c30io
   "EXECIO 1 DISKW s30io"
/*-------------------------------------------------------------------*/
/*   Enclave summary report                                          */
/*-------------------------------------------------------------------*/
 Select
  when smf30mno = 0 then do
    smf30mrd = 0; smf30mri = 0; mrp1 = 0;
    smf30mr  = 0; mrpct = 0; end;
 otherwise nop; end
  su.1 = left('Part 1: Address space accounting',40)
  su.2 = left(' ',1,' ')
```

```
  su.3 = left('OBS',4)||left('Date',12)||left('Time',9),
         ||left('Job id',8)||left('Job name',11),
         ||left('#Ind',9)||left('EXCP',7)||left('CONN',7),
         ||left('Tot CPU',9)||left('TCB',7)||left('SRB',6),
         ||left('Clock',13)||left('Serv',5)
   "EXECIO * DISKW s3Øse(stem su.)"
  c3Øse = right(i,3,'Ø')          right(Date('N',smf3Ødte,'J'),11),
          left(smf3Ømte,8)        left(smf3Øjnm,8)       ,
          left(smf3Øjbn,8)        right(smf3Øetc,5)      ,
          right(smf3Øtep,8)       right(tcn,7)           ,
          right(totcpu*Ø.Ø1,7)    right(smf3Øcpt*Ø.Ø1,6) ,
          right(smf3Øcps*Ø.Ø1,6)  left(elapstm,8)        ,
          right(smf3Øtot,1Ø)
 PUSH c3Øse
   "EXECIO 1 DISKW s3Øse"
  eu.1 = left(' ',1,' ')
  eu.2 = left('Part 2: Enclave resource accounting',4Ø)
  eu.3 = left(' ',1,' ')
  eu.4 = left('Workload Name:',15)||left(smf3Øwlm,8),
         ||left('Service Class Name:',21)||left(smf3Øscn,8)
  eu.5 = left('Resource Group Name:',23)||left(smf3Øgrn,8)
  eu.6 = left('Reporting Class Name:',23)||left(smf3Ørcn,8)
  eu.7 = left(' ',1,' ')
  eu.8 = left('Enclave count ',15)||left(' ',2,' ')||,
         left('Tcb time',1Ø)||left('% total ',9)||,
         left('Enc avg',7)
  eu.9 = left('Independent:',12)||right(smf3Øetc,4)||left(' ',3,' '),
         ||right(smf3Øenc*Ø.Ø1,6)||left(' ',2,' ')||right(p1,6),
         ||left(' ',3,' ')||right(avgincpu,6)
  eu.1Ø= left('Dependent  : n.a',16)||left(' ',3,'
')||right(smf3Ødet*Ø.Ø1,6),
         ||left(' ',1,' ')||right(p2,6)||left(' ',6,' ')||left('n.a',4)
  eu.11= left('Preemptable: n.a',16)||left(' ',3,' '),
         ||right(smf3Øasr*Ø.Ø1,6),
         ||left(' ',1,' ')||right(p3,6),
         left(' ',5,' ')||left('n.a',4)
  eu.12= left('Multi system:',12)||right(smf3Ømno,4)||left(' ',3,' '),
         ||right(smf3Ømr,6)||left(' ',1,' ')||right(mrpct,6),
         ||left(' ',3,' ')||right(mrp1,6)
   "EXECIO * DISKW s3Øse(stem eu.)"
  ew.1 = left(' ',1,' ')
  ew.2 = left('Part 2.1: Independent Enclave resource accounting',5Ø)
  ew.3 = left(' ',1,' ')
  ew.4 = left('Independent enclave DASD I/O timing',6Ø)
  ew.5 = left(' ',2Ø,' ')||left('% of Total I/O',14)||left('  Enc
avg',9)
  ew.6 = left('Connect   :',11)||right(eic,9)||left(' ',4,'
')||right(d1,6),
         ||left(' ',4,' ')||right(da1,9)
  ew.7 = left('Disconnect:',11)||right(eid,9)||left(' ',4,'
```

```
')||right(d2,6),
         ||left(' ',4,' ')||right(da2,9)
  ew.8 = left('Pending   :',11)||right(eiw,9)||left(' ',4,'
')||right(d3,6),
         ||left(' ',4,' ')||right(da3,9)
   "EXECIO * DISKW s30se(stem ew.)"
  ez.1 = left(' ',1,' ')
  ez.2 = left('Independent enclave SRM timing & Service',60)
  ez.3 = left('Active time:',12)||right(eactive,8)||left(' ',4,' '),
         ||left('Tcb su   :',10)||right(smf30esu,8)
  ez.4 = left('% Active   :',12)||right(e2,8)||left(' ',4,' '),
         ||left('%SRM su  :',10)||right(e1,8)
  ez.5 = left('Avg.time   :',12)||right(e4,8)||left(' ',4,' '),
         ||left('Avg.su   :',10)||right(e3,9)
  ez.6 = left(' ',1,' ')
   "EXECIO * DISKW s30se(stem ez.)"
END
/* Close & free all allocated files  */
  "EXECIO 0 DISKW S30CP(FINIS "
  "EXECIO 0 DISKW S30IO(FINIS "
  "EXECIO 0 DISKW S30PR(FINIS "
  "EXECIO 0 DISKW S30MS(FINIS "
  "EXECIO 0 DISKW S30SE(FINIS "
  say
  say 'Processor usage Report .....:'r30cp
  say 'I/O Activity Report ........:'r30io
  say 'Performance Report .........:'r30pr
  say 'Multisystem Enclave Report .:'r30ms
  say 'Summary Report .............:'r30se
  say
  "FREE FILE(SMF30 S30CP S30IO S30PR S30MS S30SE)"
 exit

CROSS: procedure
/* ------------------------------------------------------------*/
/*        Cover the midnight crossover                         */
/* ------------------------------------------------------------*/
arg endtime,startime
  select
   when endtime > startime then nop
   otherwise  endtime = endtime + 8640000
  end
diftm = smf(endtime - startime)
return diftm
SMF: procedure
/* REXX - convert a SMF time to hh:mm:ss:hd format   */
arg time
    time1 = time % 100
    hh    = time1 % 3600
    hh    = RIGHT("0"||hh,2)
```

```
    mm    = (time1 % 60) - (hh * 60)
    mm    = RIGHT("0"||mm,2)
    ss    = time1 - (hh * 3600) - (mm * 60)
    ss    = RIGHT("0"||ss,2)
    fr    = time //  1000
    fr    = RIGHT("0"||fr,2)
    rtime = hh||":"||mm||":"||ss||":"||fr
    return rtime
```

ENC97 EXEC

```
/* REXX EXEC to read and format SMF 97 records.                      */
/* WLM mapping for SMF Record Type 97: SYS1.MACLIB(IWMSMF97)         */
/* When an enclave is exported to one or more supporting systems,    */
/* the CPU time consumed in those foreign enclaves is accumulated in */
/* SMF type 97 records. There will be one type 97 record on each     */
/* supporting system that imported the enclave.                      */
Address TSO
Numeric digits 64
userid=SYSVAR(SYSUID)
f97en  =userid||'.r97cp.rep'            /* Foreign enclave report    */
x = MSG('OFF')
If SYSDSN(f97en) = 'OK'                 /* Check report dsn validity */
Then "DELETE "f97en" PURGE"
 "ALLOC FILE(F97) DA("f97en")",
 "UNIT(SYSALLDA) NEW TRACKS SPACE(4,2) CATALOG",
 "REUSE LRECL(80) RECFM(F B) "
/*-------------------------------------------------------------------*/
/* Print header for Foreign enclave report                           */
/*-------------------------------------------------------------------*/
rpt.1 = left('Foreign enclave report',50)
rpt.2 = left(' ',1,' ')
rpt.3 = left('Interval start TOD',20)||left('Interval end TOD',20),
        ||left('duration'10)||left('System',9),
        ||left('Dep.CPU',8)||left('Ind.CPU',8)
rpt.4 = ||left('-',78,'-')
  "EXECIO * DISKW F97(stem rpt.)"
  'EXECIO * DISKR SMF97 (STEM x. FINIS'
   Do i = 1 to x.0
/*-------------------------------------------------------------------*/
/*   Header Section                                                  */
/*-------------------------------------------------------------------*/
 smf97rty = c2d(substr(x.i,2,1))             /* Record Type - 97     */
 smf97tme = smf(c2d(substr(x.i,3,4)))        /* Record Written Time  */
 smf97dte = substr(c2x(substr(x.i,7,4)),3,5) /* Record Written Date  */
 smf97sid = substr(x.i,11,4)                 /* System Id.           */
 smf97ssi = substr(x.i,15,4)                 /* Subsystem ID = STC   */
 smf97stp = c2d(substr(x.i,19,2))            /* Record Subtype       */
/*-------------------------------------------------------------------*/
```

```
/*   Self defining section                                          */
/*-----------------------------------------------------------------*/
 smf97sdl = c2d(substr(x.i,21,4))      /* Length of Self Defining    */
                                       /* Section                    */
 smf97pof = c2d(substr(x.i,25,4))      /* Offset to Product Section  */
 smf97pln = c2d(substr(x.i,29,2))      /* Length of Product Section  */
 smf97pon = c2d(substr(x.i,31,2))      /* Number of Product Sections */
 smf97eof = c2d(substr(x.i,33,4))      /* Offset to Enclave resource */
                                       /* data section               */
 smf97eln = c2d(substr(x.i,37,4))      /* Length of Enclave resource */
                                       /* data section               */
 smf97eon = c2d(substr(x.i,41,4))      /* Number of Enclave resource */
                                       /* data sections              */
 smf97eos = c2d(substr(x.i,45,4))      /* Number of Enclave resource */
                                       /* data sections in subsequent*/
                                       /* records                    */
/*-----------------------------------------------------------------*/
/*   Product section                                               */
/*-----------------------------------------------------------------*/
IF smf97pon <> Ø Then do
   feo = smf97pof -3
 smf97rvn = SUBSTR(x.i,feo,2)        /* Record Version Num - 'Ø1'    */
 smf97pnm = SUBSTR(x.i,feo+4,8)      /* Product Name - 'SCWLM   '    */
 smf97osl = SUBSTR(x.i,feo+12,8)     /* MVS Product Level            */
 smf97syn = SUBSTR(x.i,feo+2Ø,8)     /* Local System Name (from      */
                                     /* SYSNAME PARMLIB option)      */
 smf97ist = ,                        /* Reporting interval start     */
 smf(c2d(SUBSTR(x.i,feo+28,4)))      /* time (local, Ø.Ø1 of seconds */
 start    = ,                        /* from midnight).              */
 c2d(SUBSTR(x.i,feo+28,4))           /* First record will report the */
                                     /* "IPL" time.                  */
 smf97isd = ,                        /* Reporting interval start date*/
 substr(c2x(substr(x.i,feo+32,4)),3,5)  /*    in the form ØcyydddF,  */
                                     /* where F is the sign.         */
                                     /* First record will report the */
                                     /* "IPL" date.                  */
 smf97iet = ,                        /* Reporting interval end time  */
 smf(c2d(SUBSTR(x.i,feo+36,4)))      /*  - local, Ø.Ø1 of a second   */
 end      = ,                        /* from midnight.               */
 c2d(SUBSTR(x.i,feo+36,4))
 smf97ied = ,                        /* Reporting interval end date  */
 substr(c2x(substr(x.i,feo+4Ø,4)),3,5)  /* in the form ØcyydddF,     */
                                     /* where F is the sign.         */
 smf97caf = ,                        /* Copy of RmctAdjc when this   */
 c2d(SUBSTR(x.i,feo+44,4))           /* record was produced, measures*/
                                     /* the number of sixteenths of  */
                                     /* one microsecond of CPU time  */
                                     /* per CPU service unit.        */
elaps = cross(end,start)             /* Reporting interval duration  */
end
```

```
/*-------------------------------------------------------------------*/
/*  Foreign enclave resource data section (describing foreign        */
/*  enclaves resource data)                                          */
/*  Note: type 97 records are written on an SMF interval basis.      */
/*  The CPU time is broken down by originating system - in other     */
/*  words, there is one section for each originating system that     */
/*  exported one or more enclaves that were then imported by this    */
/*  system during the interval. Note that this section will          */
/*  reflect the total CPU time consumed by all foreign enclaves      */
/*  imported from this one particular originating system.            */
/*  To identify the specific jobs that consumed CPU time, you        */
/*  need to review the SMF type 3Ø records on the originating        */
/*  system. Note that for any specific SMF interval, the sum         */
/*  of the type 3Ø CPU time may not exactly match the type 97        */
/*  CPU time, as the data for type 3Ø records is collected           */
/*  asynchronously.                                                  */
/*-------------------------------------------------------------------*/
IF smf97eon <> Ø Then do
    rof = smf97eof -3
    do k = Ø to smf97eon -1
      rff  = (rof + (k*smf97eln))
 smf97fsn = SUBSTR(x.i,rff,8)          /* Name of the system that     */
                                       /* exported the enclaves which */
                                       /* used services of the local  */
                                       /* system                      */
 smf97fcd = ,                          /* CPU time used by foreign    */
 c2d(SUBSTR(x.i,rff+8,4))*Ø.Ø1         /* dependent enclaves in Ø.Ø1  */
                                       /* of a second - conv. to sec. */
 smf97fci = ,                          /* CPU time used by foreign    */
 c2d(SUBSTR(x.i,rff+12,4))*Ø.Ø1        /* independent enclaves in Ø.Ø1*/
                                       /* of a second - conv. to sec. */
   end
  end
 f97pr.1 = right(Date('N',smf97isd 'J'),11)  left(smf97ist,8),
           right(Date('N',smf97ied 'J'),11)  left(smf97iet,8),
           left(elaps,11)                    left(smf97fsn,8),
           right(smf97fcd,7)                right(smf97fci,7)
  "EXECIO * DISKW F97(stem f97pr.)"
END
  "EXECIO Ø DISKW F97(FINIS "
 say
 say 'Foreign enclave report .....:'f97en
 say
 "FREE FILE(SMF97 F97)"
exit
CROSS: procedure
/* ------------------------------------------------------------ */
/*        Cover the midnight crossover                          */
/* ------------------------------------------------------------ */
arg endtime,startime
```

```
  select
   when endtime > startime then nop
   otherwise  endtime = endtime + 8640000
  end
diftm = smf(endtime - startime)
return diftm
SMF: procedure
/* REXX - convert a SMF time to hh:mm:ss:hd format   */
arg time
    time1 = time % 100
    hh    = time1 % 3600
    hh    = RIGHT("0"||hh,2)
    mm    = (time1 % 60) - (hh * 60)
    mm    = RIGHT("0"||mm,2)
    ss    = time1 - (hh * 3600) - (mm * 60)
    ss    = RIGHT("0"||ss,2)
    fr    = time //  1000
    fr    = RIGHT("0"||fr,2)
    rtime = hh||":"||mm||":"||ss||":"||fr
    return rtime
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Get generation dataset association

PURPOSE

This tool can be used to fetch and send back the full dataset name (with absolute generation number) of the GDG associations for a given base and version.

USAGE

Most REXX developers have come across a situation in which they want to find out the full dataset name for a given GDG base and the version number.

This tool has been created as a subroutine that REXX

developers can call from their REXX source programs by passing the base name and the relative generation number.

The subroutine, GetGen, LISTCATs the given base and tries to fetch the generation for the given version number.

As explained above, the routine requires two parameters:

1 The base name.

2 The version number should be a whole number. Valid numbers are 0 and all positive numbers. So, if 0 is supplied, the routine would get the full dataset name of the 0th generation of the base. If 1 is supplied, it would assume that -1st generation of the basename is required. If 5 is supplied, it would assume that -5th generation of the basename is required.

GETGEN REXX

```
 /*rexx*/
/*
   GETVER (Getversion) REXX routine examines the given GDGBASE and
   the generations existing for GDGBASE under consideration.

   This module is a sub-routine which can be called from any other REXX
   routine.

   TWO Parameters are required by this routine.
       Parm 1 = The BASE name for which the required generation has to
                be identified.
       Parm 2 = positive relative generation number

*/
"ispexec control errors return"
Parse arg ds ver
ds = strip(ds,B,"'")
x   = outtrap(msg.)
"listcat entries('"ds"') "
lc_rc = rc
x   = outtrap(off)
if lc_rc = 0 then
  do
    if msg.0 = 2 then
       do
         ccgetgen = 2
```

```
          dsgetgen = ''
        end
     else
        if msg.Ø > (ver * 2 + 2) then
           do
             ccgetgen = Ø
             tmp = msg.Ø - ((ver * 2) + 1)
             dsgetgen  = word(msg.tmp,3)
           end
        else
           do
             ccgetgen = 4
             dsgetgen  = ''
           end
  end
else
  do
    ccgetgen = 8
    dsgetgen  = ''
  end
"ispexec vput (ccgetgen, dsgetgen) shared"
Return
```

SET-UP

Copy the above source code to a member of your REXX PDS (which is also allocated to your profile whenever you log-on) and name that member GETGEN.

EXECUTION

The following example shows GETGEN being called from a different REXX routine (which would become the calling program for GETGEN):

```
/*……SOME REXX PROGRAM…….*/
…….
……

…       ….       …       …       …

…       ….       …       …       …

Call GETGEN arg1 arg2
"ispexec Vget (ccgetgen, dsgetgen) shared"

Say 'Return code from Getgen ' ccgetgen
```

```
Select
when ccgetgen = Ø then
     Say '-'arg2'th generation DS name is ='dsgetgen
when ccgetgen = 2 then
     Say arg1' contains No generations'
when ccgetgen = 4 then
     Say '-'arg2'th generation does not exist for 'arg1
when ccgetgen = 8 then
     Say 'Error returned from the base 'arg1
otherwise
     nop
end
 …       ….       …       …       …

…       ….       …       …       …
```

RETURN CODES

The return codes and their meanings are shown below:

1 Execution was successful.

dsgetgen will contain the full dataset name with absolute generation number.

2 GDG base supplied does not have any active generations in the catalog.

4 The required version does not exist.

8 Execution failed. Check the base name and possible typographic errors.

Suresh Kumar Murugesan
Systems Analyst
Cognizant Technology Solutions (USA)

Please note that the correct contact address for Xephon Inc is PO Box 550547, Dallas, TX 75355, USA. The phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

DFSORT's OUTFIL control statement features

OUTFIL is a very versatile DFSORT control statement. It enables you to create one or more output datasets for a sort, copy, or merge application from a single pass over one or more input datasets. It provides many features that can be used with a single output dataset or multiple output datasets, the whole aim being to eliminate programming work.

The format of the OUTFIL command and its operands are shown below:

- FILES=(FILEID,FILEID) – one output file.
- ,FNAMES=(ddname,ddname) – output defined by files.
- ,HEADER1=(field,field) – no report heading.
- ,HEADER2=(field,field) – no page headings.
- ,INCLUDE=ALL|(COMPARISONS)|NONE – include records
- ,OMIT=ALL|(COMPARISONS)|NONE – exclude records.
- ,LINES=n|ANSI|ANSI,n – report format.
- ,NODETAIL– detailed report.
- ,STARTREC – start processing with first record.
- ,ENDREC – end processing with last record.
- ,SAVE – omitted records not saved for output.
- ,OUTREC=(field,field) – record unchanged.
- ,CONVERT – record format unchanged.
- ,SPLIT – no split output.
- ,SECTIONS=(field,field) – no sections.
- ,TRAILER1=(field,field) – no report trailer.

- ,TRAILER2=(field,field) – no page trailers.

The operands *FNAMES* and *FILES* specify the DDnames of the output datasets that need to be coded in the JCL. *FNAMES* specifies one or more full 8-byte DDnames. *FILES* specifies one or more 2-character suffixes (xx) for SORTOFxx DDnames. SORTOUT is used as the DDname for an OUTFIL statement without *FNAMES* or *FILES* coded.

The following statements give examples of these operands when used with OUTFIL:

```
OUTFIL FNAMES=OUTFIL1
OUTFIL FNAMES=(DISKO,TAPEO)
OUTFIL FILES=(Ø5,AB),FNAMES=TEMP
OUTFIL STARTREC=3,ENDREC=99
```

The first OUTFIL statement uses a DDname of OUTFIL1.

The second OUTFIL statement uses DDnames of DISKO and TAPEO.

The third OUTFIL statement uses DDnames of SORTOF05, SORTOFAB, and TEMP.

The fourth OUTFIL statement would write output to the default DDname of SORTOUT.

The OUTFIL operands *INCLUDE*, *OMIT*, and *SAVE* can be used to select the records to be included in each output dataset. The *INCLUDE* and *OMIT* operands provide all of the capabilities that can be used with SORT. These operands are very powerful and enable complex comparisons and extractions to be created without having to do any programming at all.

The following shows how subsets of records can be extracted from input files:

```
OUTFIL  INCLUDE=(1,6,CH,EQ,C'DEBITS'),FNAMES=OP1
OUTFIL  INCLUDE=(8,6,CH,EQ,C'ABSENT'),FNAMES=OP2
OUTFIL INCLUDE=(8,6,SS,EQ,C'ADMIN ,OPERAT'),FNAMES=OP3
OUTFIL SAVE,FNAMES=CATCH
```

Records with *DEBITS* in positions 1–6 are written to the output dataset named *OP1*.

Records with *ABSENT* in positions 8–13 are written to the output dataset *OP2*.

Records with *ADMIN* or *OPERAT* in positions 8–13 are written to *OP3*. Records that do not match the first three criteria are captured by the fourth OUTFIL statement and will be written to the file defined by DDname *CATCH*.

OUTREC can be used to reformat the records in each output dataset. The OUTREC operand provides many reformatting options. OUTFIL OUTREC also provides one capability not available with the INREC or OUTREC statement, it lets you use /, n/ or /.../ to create multiple output records from each input record. You can split an input record into different parts and then use these as input fields for one or more output records.

Take the following:

```
OUTFIL FNAMES=BITS,OUTREC=(1,20,/,21,60)
OUTFIL FNAMES=TSPACE,OUTREC=(1,80,2/)
```

Two output records will be written to *BITS* for each input record.

The first record will contain input positions 1–20 followed by 40 blanks. The second record will contain input positions 21–80.

Each input record will be written to *TSPACE* followed by two blank records – triple spacing.

The OUTFIL operand *VLFILL* allows DFSORT to continue processing if a variable-length record is too short to contain all the specified OUTFIL OUTREC fields. Missing bytes are replaced with the specified characters or hexadecimal fill bytes so the filled fields can be processed.

```
OUTFIL FNAMES=OUT1,OUTREC=(1,4,11,20,2X,35,10),VLFILL=C' '
```

For the *OUT1* output dataset, missing bytes in positions 11–30 or 35–44 of the variable-length input records will be replaced with blanks.

Another operand, *VLTRIM*, can be used to remove trailing

bytes of the specified type from the end of any variable-length records. DFSORT decreases the length of the record by the number of trailing trim bytes, effectively removing the trim bytes.

By using operands *VTOF* (or its alias *CONVERT*) and OUTREC you can change a variable-length input file's records to fixed-length output records. *VTOF* indicates that conversion is to be performed and OUTREC defines the reformatted records. All output datasets that *VTOF* is used to create have to be RECFM=FB.

The following statement will convert the input file to a fixed file and the latter will be written to DDNAME *VBFB*.

```
OUTFIL FNAMES=VBFB,VTOF,OUTREC=(5,14,32,8,2Z,22,6)
```

The fixed-length output records for the *VBFB* dataset will contain positions 5–18 of the variable input records, positions 32–39 of the variable input records, two binary zeros, and positions 22–27 of the variable input records.

When you specify *VTOF*, by default DFSORT automatically uses:

```
VLFILL=C' '
```

So the OUTFIL statement above is actually equivalent to:

```
OUTFIL FNAMES=VBFB,VTOF,OUTREC=(5,14,32,8,2Z,22,6),VLFILL=C' '
```

Any records that have missing bytes in the positions we are extracting from will be padded with blanks. Obviously if you coded VLFILL=C'A', then missing characters would be padded with As.

The OUTFIL operand *FTOV* performs the reverse of *VTOF* and converts a fixed-length input records file into a variable-length output records file. If *FTOV* is specified without OUTREC, the entire fixed-length record is used to build the variable-length record. If *FTOV* is specified with OUTREC, the specified fields from the fixed-length record are used to build the variable-length record. Output records will consist of a 4-byte

record descriptor word that is followed by the fixed-length data. All output datasets that *FTOV* is coded for will have to be RECFM=VB.

Here's an example of FB to VB conversion:

```
OUTFIL FNAMES=FBVBA,FTOV
OUTFIL  FNAMES=FBVBB,FTOV,OUTREC=(1,1Ø,C'&',21,1Ø)
```

The variable-length output records for the FBVBA will contain a 4-byte RDW followed by the fixed-length input record.

The variable-length output records for the FBVBB will contain a 4-byte RDW followed by the characters from input positions 1–10, an '&' character, and the characters from input positions 21–30.

VLTRIM=byte can be used with *FTOV* to remove trailing bytes from the end of the variable-length output records. Here's an example:

```
OUTFIL  FNAMES=FBVBC,FTOV,VLTRIM=C'+'
```

The variable-length output records for the FBVBC dataset will contain a 4-byte RDW followed by the fixed-length input records without any trailing plus signs.

The OUTFIL operands *STARTREC* and *ENDREC* can be used to select a range of records to be included in each output dataset. *STARTREC* starts processing at a specific input record, while *ENDREC* ends processing at a specific input record.

The OUTFIL operand *SAMPLE* can be used to sample records in a number of different ways. The sample consists of the first *m* records in every *n*th interval. *STARTREC* and *ENDREC* can be used with SAMPLE to select a range of records for sampling.

```
SAMPLE=n
```

writes every *n*th record, starting at the STARTREC record and ending at or before the ENDREC record.

```
SAMPLE=(n,m)
```

writes *m* records every *n*th record, starting at the STARTREC record and ending at or before the ENDREC record.

This is best shown by the following examples:

```
OUTFIL FNAMES=OUTF1,SAMPLE=5
OUTFIL FNAMES=OUTF2,SAMPLE=(1000,2),ENDREC=2500
OUTFIL FNAMES=OUTF3,STARTREC=23,ENDREC=75,SAMPLE=25
```

OUTF1 would contain records 1, 6, 11, 16, 21, 26, etc.

OUTF2 would contain 1, 2, 1001, 1002, 2001, 2002. The ENDREC stops it progressing to 3001 and 3002.

OUTF3 would contain 23, 48, and 73. Again ENDREC will stop any further processing.

The OUTFIL operands *SPLIT*, *SPLITBY*, *FNAMES*, and *FILES* can be used to split records in rotation among the output datasets. SPLIT writes the first output record to the first output dataset, the second output record to the second output dataset, and so on. When each output dataset has one record, the rotation starts again with the first output dataset. SPLITBY=n writes the first *n* output records to the first output dataset, the second *n* output records to the second data set, and so on. When each output dataset has *n* records, the rotation starts again with the first output dataset.

Finally the operands *LINES*, *HEADER1*, *TRAILER1*, *HEADER2*, *TRAILER2*, *SECTIONS*, *NODETAIL*, *REMOVECC*, and *OUTREC* can be used to create highly-detailed three-level (report, page, and section) reports containing a variety of report elements. It is possible to produce reports with the following included automatically:

- Current date
- Current time
- Page number
- Character strings
- Blank lines.

You can also produce headings etc from input records. Full details of how to code these operands can be obtained from the *DFSORT Application Programming Guide*.

You could if you wished also use the ICETOOL DISPLAY command instead of OUTFIL to produce reports. Both perform some reporting functions that the other does not; in general, the difference between them is flexibility. OUTFIL allows you to have more control over the way reports look, but it is likely you will need to expend more effort in achieving your result. ICETOOL allows most of the effort to be performed by the program but you will not have as much control over formatting, etc.

John Bradley
Systems Programmer
Meerkat Computer Services (UK)

October 2003 – September 2005 index

Items below are references to articles that have appeared in *MVS Update* since issue 205, October 2003. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

Mainstar Software has announced a new release of its Mirroring Solutions Suite, MS/VCR. The new version has enhanced flexibility and more automation.

MS/VCR provides access to datasets on 'cloned' or 'replicated' volumes created with point-in-time copies of fast data replication tools (FlashCopy and SnapShot) or splits of continuous mirror tools (EMC TimeFinder, IBM PPRC, HDS ShadowImage, Softek Replicator, and Fujitsu Equivalent Copy). The problem with the clones is that the target volume label, internal data, and dataset names all reflect the source volume name. MSVCR solves this problem.

The new version has several new commands and new return code options. These allow users to specify exactly what they want.

For further information contact:
URL: www.mainstar.com/products/ms/msvcr/index.asp.

* * *

Novell has announced a European SUSE Linux pricing package designed to simplify Linux procurement and licence management for IBM mainframe customers seeking to extend the use of Novell SUSE LINUX Enterprise Server 9 across additional eServer architectures.

Novell's SEAL (Strategic Enterprise Agreement for Linux) package for IBM customers offers discounts of up to 50% off the list price of Novell SUSE Linux Enterprise Server 9.

Customers expanding their use of Linux on the mainframe or migrating from Unix and Windows to SUSE Linux on any eServer platform will be eligible for a sliding scale of long-term contract discounts, which increase with the number of new Linux workloads.

For further information contact:
URL: www.novell.com/linux/ibm.

* * *

NEON Systems has announced support for CICS Transaction Server Version 3.1 through NEON's mainframe Web services solution, Shadow z/Services.

The company claim that Shadow RTE can simplify the deployment of Service-Oriented Architectures (SOA) for organizations using CICS to run their mission critical business services. Shadow's ease of development, implementation, and operation, the company says, can be used to extend all the new facilities and features available with CICS TS 3.1.

For further information contact:
URL: www.neonsys.com/newsroom/press_releases/2005/20050627.asp.

* * *

IBM has announced the z/9 mainframe, which can be configured with up to 54 processing engines and each processing engine can run at 600MIPS.

For further information contact your local IBM representative.

* * *