August 2002

In this issue

3 Copying files and members between HFS directories and PDSs
11 Online batch
17 A simple DFHSM report writer
41 Editing a new member when in Option 3.4
42 Get current operating system version
44 SYSOUT Application Programming Interface
66 Matching a filename against a pattern
72 MVS news

© Xephon plc 2002
Copying files and members between HFS directories and PDSs

After working with Unix Systems Services for a while, I found that I needed a way to copy PDS members to HFS directories and files from HFS directories to a PDS. Unix Systems Services comes with the OCOPY, OGETX, and OPUTX commands to perform this function and the ISHELL ISPF application also provides ways to accomplish this. This is more than sufficient when one or two items need to be copied.

As I began to exploit USS more, I found that it was a convenient place to create software distribution packages for software that would be distributed on floppy disk or CD. Traditional Unix commands, like TAR and PAX, made this an easy proposition. I found that I regularly needed to copy entire PDSs to an HFS directory when the distribution contained traditional MVS components. Occasionally, I also had to refresh only certain subsets of files from the source PDSs. Using existing tools this was cumbersome for multiple copies and I wanted a batch equivalent to run unattended.

I first thought that I would write a REXX wrapper for OCOPY/OGETX/OPUTX that would satisfy my needs. This would have worked but became more difficult when I wanted to build in some bells and whistles using wildcards. Since I also wanted to move things back from the HFS directory to a PDS, my utility using OCOPY/OGETX/OPUTX became even more convoluted.

Then it struck me. I had abandoned the use of tools like IND$FILE long ago when I discovered the flexibility of MVS FTP using the MGET and MPUT commands. For those unfamiliar with FTP MGET and MPUT, this is a way to move the entire contents of a directory (or a wildcarded subset) from one host to another. When OS/390 is one of those hosts a couple of nice things happen for you. First, ASCII to EBCDIC translation can occur by using the default FTP ASCII transfer. Second, if a PDS is involved in an MGET or MPUT, the PDS members are moved to files or files are moved to PDS members (depending on which direction you are going). The only restriction is the eight-character
limitation on PDS member names. Since my primary direction was
from PDS to HFS, this was not a problem, but would be for an MPUT
from an HFS directory using file names longer than eight characters.

When using FTP with MGET or MPUT, FTP will default to prompt
mode. If you are moving many files or members, it is possible to invoke
FTP using the -i option to ignore prompting.

So, I decided to wrap this concept in a REXX EXEC for a nice self-
contained utility. MVS FTP will recognize whether you are on the MVS
side or USS side based on your first cd command. If there is a ‘/’ in the
target of the cd command, you are placed in the USS side; if not, you
are placed in the MVS side.

Normally, you would FTP from your host to or from another host. This
stumped me for a while. What am I FTPing to or from? Here is the
beauty of this solution. We are FTPing to ourselves. We can accomplish
everything we need using the standard TCP/IP Loopback address
(127.0.0.1). Once I realized this, my XCOPY utility was born.

XCOPY is a REXX EXEC that will read an input directory or PDS and
copy its contents to an output directory or PDS. This means it can copy
PDS to HFS, HFS to PDS, HFS to HFS, and PDS to PDS. XCOPY can
run interactively or in batch.

XCOPY parameters are:

- **FROM** – the source PDS or HFS directory.
- **TO** – the target PDS or HFS directory.
- **PW** – the password for the user performing the XCOPY. If run
  interactively, the user will be prompted if PW is missing.
- **PATTERN** – the wildcard pattern conforming to FTP MGET and
  MPUT rules. Defaulted to ‘*’ for all members (case sensitive from
  HFS).
- **REPLACE** – ‘Y’ or ‘N’. Defaults to ‘Y’.

One thing to remember (as always) – Unix is case sensitive. Therefore,
be sure to enter all HFS directory names in lower case when appropriate.
If running XCOPY in batch, you may want to use CAPS OFF in your
ISPF JCL edit session to retain lower case in the PARM string.

If XCOPY has a problem (usually a non-zero return code from FTP), it will list all the debugging information. If running under TSO/ISPF foreground, all FTP INPUT and OUTPUT dataset contents will be say’d to the screen. If running in background, SYSTSPRT will contain all the FTP INPUT and OUTPUT contents.

When XCOPY runs successfully under ISPF, all processing occurs and a simple ISPF message is issued with a success message and a count of the items that were copied. Pressing PF1/HELP will display the long message text, which will contain the list of members copied (until truncated, if a long list).

If you are processing a large number of items, batch is usually more appropriate. If XCOPY is running in background a report will be generated in SYSTSPRT identifying all items that were copied.

XCOPY

```plaintext
/********************************************************************************
/*  REXX */
********************************************************************************
/* Purpose: Copy HFS-->PDS PDS-->HFS PDS-->PDS HFS-->HFS */
/* Syntax: XCOPY from to password pattern */
/* */
/*Parms: from - The 'from' directory or PDS */
/* to - The 'to' directory or PDS */
/* pw - Your password for FTP */
/* pattern - The name pattern (defaults to '*' - all) */
/* replace - Replace existing files (Y or N - default Y) */
/* */
/* Notes: Uses FTP with the loopback address 127.0.0.1 so assumes */
/* you will use your ID and password. Can also be run in */
/* batch. All directories and/or PDS's must already exist. */
********************************************************************************
/* Change Log */
/* */
/* Standard entry */
********************************************************************************
parse upper source execenv . execname . execdsn .
signal on syntax name trap
signal on failure name trap
signal on novalue name trap
```
/***************************************************************************/
/* Initialize base values for required variables */
***************************************************************************/
FTPRC = Ø
EXITRC = Ø
ftpcount = Ø
ftplist = ''
ftpenv = sysvar('SYSENV')
x = time('r')

/******************************************************************************/
/* Start-up message */
/******************************************************************************/
startmsg = execname 'started' date() time()
if ftpenv = 'BACK' then
do
   say center(' startmsg ',78,'-')
say end
/******************************************************************************/
/* Accept parms */
/******************************************************************************/
parse arg from to pw pattern replace .
if from = '' then call rcexit 999 'Source Directory or PDS is missing'
if to = '' then call rcexit 999 'Target Directory or PDS is missing'
if pw = '' then pw = getpw()
if pattern = '' then pattern = '*'
if replace = '' then replace = 'Y'
if replace = 'Y' then replace = '(REPLACE)'
******************************************************************************/
/* Format the input and output to avoid double quoting */
******************************************************************************/
from = strip(from,'B','""
to = strip(to,'B','""
******************************************************************************/
/* Make sure the from and to exist */
******************************************************************************/
if pos('/',from) then
do
   "ALLOC F($FROM) PATH('"from"')"
   call rcexit RC 'Source' from 'does not exist'
   "FREE F($FROM)"
end
else
do
   from = ""from"
   if sysdsn(from) <> 'OK' then
call rcexit 12 'Source' from 'does not exist'
end
if pos('/',to) then
do
"ALLOC F($TO) PATH('"to"')"
  call rceexit RC 'Target' to 'does not exist'
"FREE F($TO)"
end
else
do
to = "'"to"'"
  if sysdsn(to) <> 'OK' then
    call rceexit 12 'Target' to 'does not exist'
  end

/******************************************************/
/* Echo the requested action if running in background */
/******************************************************/
if ftpenv = 'BACK' then
do
  say execname 'will copy from' from 'to' to 'using pattern' pattern
  say
end
/******************************************************/
/* Setup FTP subcommands for the copy */
/******************************************************/
inpt.0 = 6
inpt.1 = userid(pw
inpt.2 = 'cd' from
inpt.3 = 'lced' to
inpt.4 = 'ascii'
inpt.5 = 'mget' pattern replace
inpt.6 = 'quit'
/******************************************************/
/* Allocate the FTP INPUT dataset */
/******************************************************/
"ALLOC F(INPUT) NEW CATALOG TRAKS DSORG(PS) UNIT(VIO)"
  call rceexit RC 'ALLOCATE error on VIO INPUT'
/******************************************************/
/* Allocate the FTP OUTPUT dataset */
/******************************************************/
"ALLOC F(OUTPUT) NEW CATALOG CYLINDERS DSORG(PS) UNIT(VIO)"
  call rceexit RC 'ALLOCATE error on VIO OUTPUT'
/******************************************************/
/* Write FTP subcommands to INPUT file */
/******************************************************/
"EXECIO * DISKW INPUT (STEM INPUT. FINIS"
  call rceexit RC 'EXECIO DISKW error in FTP INPUT'
/******************************************************/
/* Invoke FTP on the loopback address 127.0.0.1 */
/******************************************************/
"FTP -i 127.0.0.1 (EXIT"
FTPRC = RC
call rcsexit FTPRC 'FTP error on 127.0.0.1'

call rcsexit RC 'EXECIO error on FTP OUTPUT reporting but FTP RC='FTPRC

if ftpenv = 'BACK' then
  do
    say 'The following items were successfully copied:'
    say ftpmsg . ftpcmd file.
  end

do i=1 to output.Ø
  parse var output.i ftpmsg . ftpcmd file.
  if ftpcmd = 'RETR' then
    do
      ftpcount = ftpcount + 1
    end
  end

if ftpenv = 'BACK' then
  say file
else
  ftplist = ftplist file
end

if ftpenv = 'FORE' then
  do
    zedmsg = execname ftpcount 'items RC='FTPRC
    zedlmsg = execname 'from' from 'to' to 'pattern' pattern
    zedlmsg = zedlmsg ftpcount 'items RC='FTPRC 'copied:' ftplist
    address ISPEXEC "SETMSG MSG(ISRZ000)"
  end
else
  do
    say ftpcount 'items copied'
  end
end

/* Shutdown */
/* If errors print the contents of the FTP INPUT (FTP Commands) */
if FTPRC <> Ø then
  do
    say
    say 'FTP input statements:'
    say
    do i=1 to input.Ø
      say input.i
    end
    say
    say 'FTP output messages:'
    say
  end
end
/* Cleanup the "temporary" datasets */
call outtrap "output", Ø
"FREE F(INPUT)"
"FREE F(OUTPUT)"
/* Shutdown message */
endmsg = execname 'ended' date() time() time('e') 'RC='EXITRC
if ftpenv = 'BACK' then
  do
    say
    say center(' 'endmsg' ',78,'-')
  end
  exit(EXITRC)
/* Subroutines */
/* GETPW - Ask the user for their password (if foreground) */
getpw: if sysvar('SYSENV') = 'BACK' then
call rcexit 913 'Missing Password'
do while pw = '' | substr(pw,1,1) = ' ',
  say 'Please enter your password'
  parse pull pw
end
return pw
/**
 * RCEXIT - Exit on non-zero return codes
 */
rcexit: parse arg EXITRC zedlmsg
   if EXITRC <> 0 then
     do
       /* If execution environment is ISPF then VPUT ZISPFRC */
       if execenv <> 'OMVS' then
         do
           "ISPQRY"
           if RC = 0 then
             do
               zispfrc = EXITRC
               address ISPEXEC "VPUT (ZISPFRC)"
             end
           end
       end
       /* If a message is provided, wrap it in date, time and EXITRC */
       if zedlmsg <> '' then
         do
           zedlmsg = date() time() execname zedlmsg 'RC='EXITRC
       end
       /* If execution environment is ISPF SETMSG. If not, say the message */
       if sysvar('SYSENV') = 'BACK' | execenv = 'OMVS' then
         say zedlmsg
       else
         do
           "ISPQRY"
           if RC = 0 then
             address ISPEXEC "SETMSG MSG(ISRZ000)"
           else
             say zedlmsg
           end
       end
       /* Signal SHUTDOWN. SHUTDOWN label MUST exist in the program */
       signal shutdown
     end
   else
     return
   /* Issue a common trap error message using rcexit */
   trap: trapt = condition('C')
     if trapt = 'SYNTAX' then
Online batch

THE PROBLEM

As we all know, the main differences between batch and online programs are:

- Duration
- Complexity
- Amount of data
- Parameter dependency
- Schedule dependency.

Batch programs run much longer than online ones because of their larger complexity and demands for scheduling (not only processing but
data too). Also online programs deal with a limited set of data in one moment and do it at random moments of time, while batch programs deal with huge amounts of data, most often sequentially, in a limited period of time.

The aim of this article is to look at the fourth difference in the list – parameter dependency. Batch programs mostly deal with predefined parameters that are known before submitting a batch job. This can cause problems in situations where we need to change such parameters very often. If we leave the changing of the parameters to the operators there is a danger of mistakes being made, especially if there are dates or some long strings. So, a better approach is to create separate JCL, one for each set of parameters, and then schedule them appropriately. Or there is another solution, which I will describe here.

A SOLUTION

I want to show you how to create batch programs that are aware of parameter values online, which means at the time they are running, not from an operator or as predefined values. I found that the best way is to create parameter values directly in my programs. So I will need to use two passes. First I need to find out what my parameter values are, and in the next pass I will use those values for processing the data. But how will my JCL be aware of those values? Well, I will create my JCL on-the-fly, directly in the program.

The only restriction will be that I need to have several JCL. The first one is static and it has a step that will create the content of the next one, with real parameter values inside. A good approach would be to put these dynamic JCL in a separate PDS library, to avoid any accidental updates, because I must have permission to update that library from my programs.

The next technique that we can use is the usage of JCL procedures, separate members that we can call in an EXEC statement – something like a JCL module. PROCs can have parameters of their own, so we can call one PROC with different parameter values.

As I can create JCL on-the-fly, I can also create any other necessary parameter sources and some of them you will see in my examples.
EXAMPLES

Where do we need to change parameters for batch programs in an online manner? Well, I found the following situations.

Working with an unknown number of datasets

As a matter of fact this is the example where I used this ‘online’ batch technique for the first time. I had to divide the input dataset into an unknown number of datasets, named by the value of one field from the dataset record. The problem was to design a solution to work with an unknown number of output datasets, not knowing the names of these datasets before running the program. So parameters in this situation were values taken from fields in the input record.

I needed to divide this problem into two parts. First I wrote a program to process the input dataset and to produce a JCL member having as many steps as the input dataset demanded to have output datasets. Every step will be a call to a general PROC member, which will process the input dataset for one value of the parameter. The JCL looks like this:

```bash
//MANYOUTD JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
/*
//ESYLIB JCLIB ORDER=(APPHELP.UNIT.PROCLIB)
/*
//STEP1 EXEC PRONEVAL,VALUE='VALUE1',PRI=5,SEC=3
//STEP2 EXEC PRONEVAL,VALUE='VALUE2',PRI=25,SEC=5
//STEP3 EXEC PRONEVAL,VALUE='VALUE3',PRI=500,SEC=50
//...
//STEPn EXEC PRONEVAL,VALUE='VALUEn',PRI=5,SEC=3
//...
//STEP125 EXEC PRONEVAL,VALUE='VALUE125',PRI=15,SEC=5
```

VALUE1, VALUE2, etc are synonyms for actual values of parameter VALUE.

So my first program processes the input dataset and, for every different value in a specific field, saves that value, calculates the necessary size of the output dataset, and writes one line in the JCL member which is the output for that program.

So there are as many steps as possible values in the input dataset. This can be different every time we run the program, and that’s why it is online batch. Every step calls a procedure with three PROC parameters:
VALUE, which will be the value that we are currently working on, and PRI and SEC as values for number of units for allocation. PRI and SEC are a helpful side effect in this example because we now know exactly what space we need for our output datasets.

In the PDS library APPHLQ.UNIT.PROCLIB, I have the following member with the name PRONEVAL (PRocess ONE VALue). It is actually my PROC, which will process the input dataset for the value of the field in the input record, given in the PROC parameter VALUE:

```plaintext
/* PROC DATE 011101

** PROC FOR PROCESSING INPUT DATASET FOR ONE POSSIBLE FIELD VALUE
** PRONEVAL PROC VALUE=,PRI=,SEC= */
*/
/* STEPPGM EXEC PGM=EXTRACT,PARM='&VALUE.' */
/* STEPLIB DD DSN=APPHLQ.UNIT.LOADLIB,DISP=SHR */
/* INPUT DD DSN=APPHLQ.INPUT.FILE,DISP=SHR */
/* OUTPUT DD DSN=APPHLQ.&VALUE..FILE,DISP=(NEW,CATLG,DELETE), */
/* DCB=(LRECL=600,RECFM=FB,BLKSIZE=0), /*
/* SPACE=(CYL,(&PRI,&SEC),RLSE) */
/* SYSPRINT DD * */
/* PEND */

Program EXTRACT must have the input parameter defined through the PARM option, because this is the only way that we can pass parameters to programs using PROCs.

Sort conditions

There are few programmers who like writing report programs. From a client’s point of view it is a very important part of the application – it is the way they see the data. But not all of them like to see data in the same way. Different people like the data in a different order. Some like to have data ordered by invoice number, some by product name, or in some more complex order. We satisfy different order types by sorting our datasets. But if we have many orders we will have lots of JCL with different conditions in the SORT step. Or, as I stated before, someone will need to change that every time the program runs. But what happens if we wish to give our clients the freedom to choose the sorting order themselves? They can ask for a different order every time.
One of the solutions is to include online parameters in our batch application. We can ask our client to define sort order in some way, which must be easy for them. Then we will have a program which will process that order, and we will produce standard SORT order statements in some dataset or member in a PDS library. Then we will use it in the SORT step. Something like this:

```
//REPORTS JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
/srcproducingreportswithdifferentorders
//input dd ds=apphlq.order.file,disp=shr
//output dd ds=apphlq.sortord.file,disp=shr
//sysprint dd sysout=* /*
//report2 exec pgm=prodrep
//input dd ds=apphlq.sortinp.file,disp=shr
//output dd ds=apphlq.report.file,disp=shr
//sysprint dd sysout=*
/*
```

In this JCL:

- **PRODCOND** is the program that processes the sort requirements from the client.

- **PRODREP** is the program that produces the report on the sorted data as the client asked.

If APPHLQ.ORDER.FILE is empty, PRODCOND produces the report using the most common order for sorting.

SENDING E-MAILS

Sending e-mails from a host can be a very useful tool for informing someone about events on the system. It can also be a standard requirement from the client to have some kind of automatic communication about ‘where’ the processing of his data is at any moment. But the recipient
of our e-mail can be unknown before the actual data comes to us. Actually, the e-mail address of the recipient can be part of the data that is sent by the client. The situation with our program is the same as for an online program when it gets data from an operator: it happens while the program runs, and there is no way of finding it before.

The simplest way to send e-mails from a host is to use IEBGENER:

```
//SENDMAIL JOB MSGCLASS=Z,CLASS=B,NOTIFY=&SYSUID
/***************************
//** SENDING MAIL FROM HOST USING IEBGENER
***************************/
//SENDMAIL EXEC PGM=IEBGENER
//SYSUT1 DD DSN=APPHLQ.HEADER.MSG,DISP=SHR
// DD DSN=APPHLQ.BODY.MSG,DISP=SHR
//SYSUT2 DD SYSOUT=(B,SMTP)
//SYSPRINT DD SYSOUT=* 
//SYSSIN DD DUMMY
/**

The content of APPLQ.HEADER.MSG is something like this:

HELO SMTP_SERVER
MAIL FROM:<senderid@company.com>
RCPT TO: <emailid@client.com>
DATA
SUBJECT: MESSAGE TITLE

So, we can produce dataset APPLQ.HEADER.MSG in our program for sending an e-mail, with all the relevant data. We know the SMTP_server and who the sender is, we can define a message title in the program, and we can enter the recipient’s address from our input data. Our batch program will work in an online mode.

CONCLUSION

These were just some examples of how we can make batch behave as if it were online, to receive or produce input parameters, and then change the results, as the program works or just before it starts to work. Certainly there are many others and maybe you already use some. I hope this article will help you to think in a different and new way.

Predrag Jovanovic
Project Developer
Pinkerton Computer Consultants Inc (USA) © Xephon 2002
PROBLEM
As is commonly known, DFHSM is a rather complex software package responsible for performing automated space management and availability in a storage media hierarchy.

In order to make sure that the DFHSM system at a given installation is working as desired, one must be aware that there is simply no substitute for frequent analysis of detailed information about the functions DFHSM performs. However, extracting, processing, and analysing this information requires experienced personnel as well as a great deal of time and effort, and there is a real danger that the DFHSM system will not receive the attention it deserves. Not having a full and comprehensive overview of DFHSM’s operation could put an installation’s data at risk of loss, to put it mildly.

It should be noted that there are just a few elementary reports available from within DFHSM itself, which can hardly provide any insight into its functioning. In the past few years several DFHSM add-on products have appeared on the market trying to bridge this gap. These products vary greatly in complexity, functionality, and reporting capabilities, as well as in flexibility.

SOLUTION
In order to find out what DFHSM was doing, a simple report writer was produced. This report writer tries to solve the problems mentioned above with its ability to access and to process the wealth of information DFHSM writes to SMF records. I chose to process the SMF records because the DFHSM log file is not formatted in a user-friendly way, and analysis of control datasets (CDS) is a bit difficult with the standard IBM tool set currently available. Prior to writing this tool, our storage administrator had to spend hours looking through huge DFHSM logs, in an attempt to track and document the functions and completion status of work performed by DFHSM.
The amount of data DFHSM collects and writes to SMF is enormous, and this report writer is an easy way to access what one needs from that data. The sets of reports it provides to technical staff give information and analysis needed to monitor and correct DFHSM operations. These reports quickly identify work that has been completed and, more importantly, not completed by DFHSM. Some of the information reported on is where the data was and where it was moved to, along with much more pertinent information.

As already said, this report writer uses SMF records produced by DFHSM. However, before using this program, one has to determine the SMF identification number related to DFHSM functional statistical records (FSR), which is the only source utilized. In a typical situation, DFHSM writes two types of SMF records: default SMF IDs are 240 for daily and volume statistic records (DSR, VSR), and 241 is reserved for functional statistics records (FSR and WWFSR). If you are not certain which ID is assigned to DFHSM, a simple `QUERY SETSYS` will tell you that:

```
ARC0150I JOURNAL={NONE | SPEED | RECOVERY}, LOG={YES | NO | HELD},
TRACE={YES | NO}, SMFID={smfid | NONE}, DEBUG={YES | NO}
```

Note that SMFID=NONE means there are no records being collected!

In general, there are twenty functional subtypes of FSR that DFHSM can write to an SMF record. What is really written depends on a particular function being performed on one dataset, so, if DFHSM’s function is not utilized, one will not encounter its subtype record. When a DFHSM function is executed, only selected fields within the FSR record are set. Which fields are actually set depends on the function being performed and the method used to request the function. Function subtypes 1-14 and 17-20 are FSR records, while function subtypes 15/16 are ABARS WWFSR records and are beyond the scope of this report writer.

A detailed description of the layout of an FSR record and its fields can be obtained from the DFSMS manual `DFSMShsm Implementation and Customization Guide`.

Eleven sets of reports are produced by this report writer, each providing in-depth information on functions DFHSM performed:
• FSRSTAT is a DFHSM functions overall report providing information such as number of functions completed (with and without errors), various timings and timeframes, the functions performed by various parameters (by age of datasets – which is function specific but very useful for identifying thrashing conditions, by request, by date, or by one hour slot; this can help to identify tasking level or drives needed), extent reduction, expiration of datasets statistics, and thrashing analysis.

• FUNSUMM contains a set of summary reports on each function performed – as one drills down to more specific functions, statistics get more interesting and revealing.

• ERRORS is a detailed report of functions ending with a non-zero return code.

• MGMTCLAS reports describe activity of DFHSM against management classes, datasets managed by each class (by age criteria), and thrashing.

• BKRCVY provides a detailed report on backed up and recovered datasets.

• BKEDEX is a detailed report on expired and deleted incremental back-up datasets.

• MIGREC provides detailed reports on migrated and recalled datasets.

• MIGDELEX reports provide a list of deleted and expired migrated (or primary) datasets as well as a report on the partial release of unused space.

• RECYCLE reports lists of recycled tapes and datasets.

• DUMPS report is a volume dump list.

• TAPES is a report on tapes used by DFHSM’s functions.

It should be noted that this report writer is not comprehensive but nevertheless is an open-ended program that allows users to modify and customize reports generated so as to meet an installation’s needs or requirements. Reporting can be done on both current and historical
information provided that an appropriate database was created.

The report writer was written in SAS language and was kept very simple in order to maintain compatibility across various versions of SAS software. The code was tested on SAS Version 5.16 and SAS Version 6.06, using only functions of the BASE component.

PROGRAM CODE

```
//FSR241   EXEC  SAS
//SYSPRINT DD SYSOUT=X
//FSRSTAT DD SYSOUT=X
//FUNSUMM DD SYSOUT=X
//ERRORS  DD SYSOUT=X
//MGMTCLAS DD SYSOUT=X
//BKRCVDY DD SYSOUT=X
//BKDELEX DD SYSOUT=X
//MIGREC  DD SYSOUT=X
//MIGDELEX DD SYSOUT=X
//RECYCLE DD SYSOUT=X
//DUMPS   DD SYSOUT=X
//TAPES   DD SYSOUT=X
//SMF     DD DSN=your.smf.file,DISP=SHR
//SYSSN   DD *
PROC FORMAT;

VALUE FSRFMT
  1='MIGRATE PR->L1'
  2='MIGRATE L1->L2'
  3='MIGRATE PR->L2'
  4='RECALL L1->PR'
  5='RECALL L2->PR'
  6='DELETE MIG. DS'
  7='DAILY BACKUP'
  8='SPILL BACKUP'
  9='RECOVERY'
10='RECYCLE BACKVOL'
11='DELETE BY AGE'
12='RECYCLE ML2'
13='VOLUME DUMP'
14='RESTORE'
15='ABACKUP'
16='ARECOVER'
17='EXPRIE PR/L1-2'
18='PARTREL'
19='EXPRIE INCR. BK'
20='DELETE INCR. BK';

VALUE MIGRAT
  1='95'
```
2='95'
3='95'
4='94'
5='94'
7='93'
8='93'

VALUE TPT
2='98'
3='98'
5='97'
7='98'
8='98'
9='97'
10='97'
12='97'

VALUE TIMEFMT
0='00:00 -> 00:59'
1='01:00 -> 01:59'
2='02:00 -> 02:59'
3='03:00 -> 03:59'
4='04:00 -> 04:59'
5='05:00 -> 05:59'
6='06:00 -> 06:59'
7='07:00 -> 07:59'
8='08:00 -> 08:59'
9='09:00 -> 09:59'
10='10:00 -> 10:59'
11='11:00 -> 11:59'
12='12:00 -> 12:59'
13='13:00 -> 13:59'
14='14:00 -> 14:59'
15='15:00 -> 15:59'
16='16:00 -> 16:59'
17='17:00 -> 17:59'
18='18:00 -> 18:59'
19='19:00 -> 19:59'
20='20:00 -> 20:59'
21='21:00 -> 21:59'
22='22:00 -> 22:59'
23='23:00 -> 23:59'

VALUE AGEFMT
0='0'
1='1'
2='2'
3='3'
4='4'
5='5'
6-HIGH='6+';

VALUE ELAPFMT (FUZZ=.5)
Ø-9='Ø-9'
1Ø-19='1Ø-19'
2Ø-29='2Ø-29'
3Ø-39='3Ø-39'
4Ø-49='4Ø-49'
5Ø-59='5Ø-59'
6Ø-HIGH='6Ø+'

VALUE EXPFMT
1='RECOVERY REQUEST'
2='ML1 DS EXPIRED'
3='ML2 DS EXPIRED'
4='BACKUP VER. BEING EXPIRED'
5='TAPE BACKUP VER. DELETED'
6='DELETED BY EXPDT/MGTCM';

VALUE $FMTREQ
'ØØ0'='AUTOMATIC-NOWAIT'
'Ø01'='AUTOMATIC-WAIT'
'11Ø'='TSO-NOWAIT'
'111'='TSO-WAIT'
'010'='BATCH-NOWAIT'
'011'='BATCH-WAIT';

OPTIONS NOCENTER SOURCE2 LINESIZE=132;

DATA FSR (DROP = ID MVSXA OFFSMF MVSXAFLG
SYSTEM EOFSTRNG FSRID FSRDEV
FSRABCC FSRARC FSRGRP FSRRAF FSRQON
FSRADTR FSRTIMRH FSRTIMMR FSRTIMRS
FSRTIMS FSRTIME FSRTIMA
FSRDATE FSRTSD FSRUSER FSRWAIT
FSRTIMSH FSRTIMMS FSRTIMSS I J X
FSRTIMEH FSRTIMEM FSRTIMES FSRDAT1
FSRTIMAH FSRTIMAM FSRTIMAS FSRXXX
FSRDORG FSRRCFM FSROPTCD
FSRDCL2 FSRDCL3 FSRDCL4 FSRDCL5 ) ;

INFILE SMF STOPOVER LENGTH=LENGTH COL=COL RECFM=VBS LRECL=32768
JFCB=SMFJFCB START=BEGINCPY;
LENGTH ID MVSXA OFFSMF 2;
FORMAT
MVSXAFLG HEX2.
SMFJFCB $HEX200.
SYSTEM $4.
;
IF OFFSMF=0 THEN DO;
IF SUBSTR(SMFJFCB,100,1)="....1..."B THEN OFFSMF=4;
ELSE OFFSMF=0;
BEGINCPY=OFFSMF+1;
RETAIN BEGINCPY OFFSMF SYSTEM;
END;
IF OFFSMF=4 THEN DO;
  INPUT @5 EOFSTRING $CHAR7. @;
  IF EOFSTRING='SMF EOF'
    OR EOFSTRING='SMFEOF' THEN STOP;
END;
INPUT @1+OFFSMF MV SXAFLG   PIB1.
   @2+OFFSMF ID         PIB1.
   @11+OFFSMF SYSTEM     $4.
@;
MV SXA=0;
IF MV SXAFLG='.....1..'B THEN MV SXA=1;
IF ID=241;

INPUT @011 FSR SID       $4.
   @015 FSR JBN     $8.
   @031 FSR UID     $8.
   @039 FSR TYPE    PIB1.
   @040 FSR FLAGS   $1.
   @041 FSR DSN     $44.
   @085 FSR TVOL    $6.
   @091 FSR DEVT    PIB4.
   @095 FSR FVOL    $6.
   @105 FSR RC      PIB4.
   @109 FSR REAS    PIB4.
   @113 FSR ABC     PIB4.
   @117 FSR DARC    PIB2.
   @119 FSR GRP     $8.
   @127 FSR RACF    IB1.
   @129 FSR RQN     IB4.
   @133 FSR DAT1    PIB1.
   @133 FSR DATR    PD4.
   @137 FSR TIMR    PK1.
   @138 FSR TMR     PK1.
   @139 FSR TMR S   PK2.2
   @141 FSR TIM SH  PK1.
   @142 FSR TIM SM  PK1.
   @143 FSR TIM S   PK2.2
   @145 FSR TIME H  PK1.
   @146 FSR TIM EM  PK1.
   @147 FSR TIMES   PK2.2
   @149 FSR TIM AH  PK1.
   @150 FSR TIM AM  PK1.
   @151 FSR TIM AS  PK2.2
   @153 FSR DLU ?? PD4.
   @157 FSR DL M1   PIB1.
   @157 FSR DL M    PD4.
   @161 FSR BY TR   IB4.
   @165 FSR BY TW  IB4.
@169 FSRTRKR IB2.
@171 FSRTRKW IB2.
@173 FSRDORG PIB2.
@175 FSRFLG2 PIB2.
@176 FSRXXX PIB2.
@177 FSRCPU PIB4.2
@181 FSRAGE IB2.
@183 FSRRECFM IB1.
@184 FSROPTCD IB1.
@211 FSRMGTC $8.
@219 FSRFLG3 PIB1.
@237 FSRNENT1 IB2.
@239 FSRNENT2 IB2.
@241 FSRDCOPR PIB2.
@243 FSRDCOPF PIB2.
@245 FSRDCL1 $CHAR8.
@253 FSRDCL2 $CHAR8.
@261 FSRDCL3 $CHAR8.
@269 FSRDCL4 $CHAR8.
@277 FSRDCL5 $CHAR8. @;

IF FSRTYPE LT 1 THEN DELETE;
IF FSRDAT1 < 2 THEN FSRDATR = FSRDATR + 1900000;

IF FSRFLG2='....1............'B THEN INPUT
   @101 FSRADE ?? PD4. @;
ELSE INPUT
   @101 FSRGEN PIB4. @;

IF FSRFLG2= '.1............'B THEN FSRTSO = '1';
ELSE
   FSRTSO = '0';
IF FSRFLG2= '.1............'B THEN FSRUSER='1';
ELSE
   FSRUSER='0';
IF FSRFLG2= '.1............'B THEN FSRWAIT='1';
ELSE
   FSRWAIT='0';

MATRIX= FSRTSO !! FSRUSER !! FSRWAIT;

IF FSRXXX = '....1............'B THEN EXREDU = 'YES';
ELSE
   EXREDU = 'NO';
IF FSRXXX = '....1............'B THEN CONVER = 'YES';
ELSE
   CONVER = 'NO';

IF FSRFLG3='1............'B THEN FSREX=1;
IF FSRFLG3='1............'B THEN FSREX=2;
IF FSRFLG3='1............'B THEN FSREX=3;
IF FSRFLG3='1............'B THEN FSREX=4;
IF FSRFLG3='1............'B THEN FSREX=5;
IF FSRLG3='.....1..'B THEN FSrex=6;

IF FSRNENT1 GT 0 THEN  DO I = 1 TO FSRNENT1;
   INPUT @297 TAPEVOL $6.
      @303 FSRTFLGS PIB1.
      @305 FSRTBYBK PIB4.  @;
   IF FSRNENT2 GT 0 AND FSRTYPE =10 OR FSRTYPE=12
      THEN DO J = 1 TO FSRNENT2;
         INPUT @297 TAPEVOL2 $6.
            @305 FSRTBYB2 PIB4.  @;
         FSRTVOL = TAPEVOL2 ;
         OUTBLKS= FSRTBYB2 ;
      END;
   END;

IF FSRTAT1 < 2 THEN FSRTATR = FSRTATR + 1900000;

FUNCTION=PUT(FSRTYPE,FSRTFMT.);
TIME =PUT(FSRTIMRH,TIMEFMT.);
AGE =PUT(FSRAGE,AGEFMT.);
REQ =PUT(MATRIX,$FMTREQ.);
EXPIRE =PUT(FSREX,$EXPFMT.);

DATE=INPUT(PUT(FSRTATR,7.),JULIAN7.);
FORMAT DATE DATE.

FSRTIMR=HMS(FSRTIMRH,FSRTIMRM,FSRTIMRS);
FORMAT FSRTIMR TIME8.  .
FSRTIM=HMS(FSRTIMSH,FSRTIMSM,FSRTIMSS);
FORMAT FSRTIM TIME8.  .
FSRTIME=HMS(FSRTIMEH,FSRTIMEM,FSRTIMES);
FORMAT FSRTIME TIME8.  .
FSRTIMA=HMS(FSRTIMAH,FSRTIMAM,FSRTIMAS);
FORMAT FSRTIMA TIME8.  .

IF FSRTIM GE FSRTIMR THEN DELAY = FSRTIM - FSRTIMR;
   ELSE DELAY = FSRTIM + '24.00.00'T - FSRTIMR;
   FORMAT DELAY TIME8.  .

IF FSRTIMA GE FSRTIM THEN MOUNT = FSRTIMA - FSRTIM;
   ELSE MOUNT = FSRTIMA + '24.00.00'T - FSRTIM;
   FORMAT MOUNT TIME8.  .

IF FSRTIME GE FSRTIMR THEN DURATION = FSRTIME - FSRTIMR;
   ELSE DURATION = FSRTIME + '24.00.00'T - FSRTIMR;
   ELAPS =PUT(DURATION,ELAPFMT.);
   FORMAT DURATION TIME8.  .

IF FSRTIMA GE FSRTIMR THEN PENDING = FSRTIMA - FSRTIMR;
ELSE PENDING = FSRTIMA + '24.00.00'T - FSRTIMR;
FORMAT PENDING TIME8.;
X = DURATION - PENDING;
FORMAT X 5.;
XP =PUT(X,ELAPFMT.);

IF FSRRC EQ 0 THEN RETCODE='OK ';
ELSE RETCODE='ERROR' ;

IF FSRMGTCL=' ' THEN FSRMGTC1='NON SMS' ;

LABEL
FSRAGE = 'DAYS SINCE*LAST*REF'
FSRDLM = 'TIME LAST*MOVED'
FSRDLU = 'LAST REF*DATE'
FSRRC = 'RETURN*CODE'
FSRREAS = 'REASON*CODE'
FSRTIMR = 'REQUEST*TIME'
FSRFVOL = 'FROM*VOLUME'
FSRTVOL = 'TO*VOLUME'
FSRDSN = 'DATASET*NAME'
FSRMGTCL='MANAGEMENT*CLASS'
FSRCPU = 'CPU TIME*USED'
FSRTIMR = 'TIME OF*REQUEST'
DATE = 'DATE OF*REQUEST'
FSRDCL1 = 'DUMP*CLASS'
FSRDCPF='DUMP*COPIES*FAILED'
FSRDCCP='DUMP*COPIES*REQUESTED'
FSRGEN = 'BACKUP*GEN*NUMBER'
TAPEVOL = 'TAPE VOLUME*SERIAL'
FSRNENT1='NUMBER TAPES*USED'
FUNCTION='DFHSM*FUNCTION'
DATE = 'DATE OF*REQUEST'
DURATION='ELAPSED*TIME'
RETCODE = 'RETURN*CODE' ;

PROC SORT DATA=FSR; BY FSRTYPE RETCODE;

*----- OVERALL DFHSM FUNCTION SUMMARY -----

PROC SUMMARY DATA=FSR NWAY;
CLASS RETCODE;
BY FSRTYPE;
VAR DURATION PENDING FSRCPU;
OUTPUT OUT=K
MEAN(DURATION PENDING FSRCPU)= AVGRESP AVGPEND AVGCPU
N=COUNT ;

DATA SUMMARY; SET K;
FUNCTION=PUT(FSRTYPE,FSRFMT.);
LABEL COUNT = 'TIMES*FUNCTION*EXECUTED'
  AVGRESP = 'AVG*ELAPSED*TIME'
  AVGPEND = 'AVG*PENDING*TIME'
  FUNCTION = 'DFHSM*FUNCTION'
  AVGCPU = 'AVG*CPU*TIME';

PROC PRINTTO PRINT=FSRSTAT; OPTIONS PAGENO=1;

PROC PRINT DATA=SUMMARY UNIFORM NOOBS SPLIT="*";
  VAR FUNCTION RETCODE COUNT AVGRESP AVGPEND AVGCPU;
  FORMAT AVGRESP AVGPEND AVGCPU TIME8 . ;
  TITLE1"DFHSM FUNCTION OVERALL REPORT";

*----- FUNCTIONS IN ERROR: RC > 0-----;

DATA ERROR(KEEP=DATE FSRDSN FSRTYPE FUNCTION FSRAGE
  FSRRC FSRREAS FSRTVOL FSRFVOL FSRTIMR);
SET FSR; IF FSRRC NE 0;

PROC SORT DATA=ERROR NODUP; BY DATE FSRTIMR FSRTYPE;
PROC PRINTTO PRINT=ERRORS; OPTIONS PAGENO=1;

PROC PRINT DATA=ERROR UNIFORM NOOBS SPLIT="*";
  TITLE1"DFHSM FUNCTION ERRORS DETAILS REPORT";
  VAR FSRTIMR FUNCTION FSRFVOL FSRTVOL
    FSRRC FSRREAS FSRDSN ;
    BY DATE;

DATA OK; SET FSR; IF FSRRC GT 0 THEN DELETE;

*----------- MIGRATION -----------;

DATA MIG; SET OK;
MIG=PUT(FSRTYPE,MIGRAT.);
IF MIG='95';
TP=PUT(FSRTYPE,TPT.);
IF TP='98' THEN FSRTVOL=TAPEVOL ;

IF FSRFLAGS = '....1....'B THEN KREAD = FSRBYTR;
   ELSE KREAD = (FSRBYTR / 1024);
IF FSRTLGS = '1....1....'B THEN KBWRITE = FSRTBYBK;
   ELSE KBWRITE = (FSRBYBK / 1024);

IF FSRTYPE = 1 THEN DO;
   IF FSRFLAGS = '....1....'B THEN KBWRITE= FSRBYTW;
      ELSE KBWRITE= (FSRBYTW / 1024);
   END;

MBREAD=(KREAD/1024);  MBWRITE=(KBWRITE/1024);
TOTKB=KBREAD+KBWRITE;  COMPR=KBREAD/KBWRITE;
XFERTIME=DURATION - PENDING;
KBSEC=TOTKB/XFERTIME;  TOTMB=(KBREAD+KBWRITE)/1024;

IF FSRDLM1 < 2 THEN FSRDLM = FSRDLM + 1900000;
   DLM  = INPUT(PUT(FSRDLM,7.),JULIAN7.);

LABEL FSRAGE = 'DAYS*ON*PRIMARY/ML1'
   DLM  = 'MIGRATE*DATE'
   FSRTIMR = 'MIGRATE*TIME'
   KBREAD = 'DS SIZE*IN KB'
   KBSEC  = 'TRANSFER RATE'
   COMPR  = 'COMPRESS RATE';

*----- MIGRATION DATAIL REPORT -----

PROC SORT DATA=MIG NODUP;
   BY DATE FSRTIMR FUNCTION;
PROC PRINTTO PRINT=MIGREC; OPTIONS PAGENO=1;

PROC PRINT DATA=MIG UNIFORM NOOBS SPLIT="*";
TITLE1"DFHS MIGRATION ACTIVITY";
BY DLM;
FORMAT DLM DATE.;
VAR FSRTIMR FUNCTION FSRAGE FSRMGTCL
   KBREAD COMPR FSRFVOL FSRTVOL FSRDSN;
FORMAT FSRCPUS TIME11.;

*----- MIGRATION SUMMARY REPORT -----

PROC SUMMARY DATA=MIG NWAY;
   CLASS FUNCTION DATE;
VAR DURATION PENDING TOTMB MBREAD COMPR KBSEC FSRCPUS;
OUTPUT OUT=S
MEAN(DURATION PENDING TOTMB MBREAD COMPR KBSEC FSRCPUS)=
   AVGRESP AVGPEND AVGMB AVGSIZE AVGCOMP AVGKBSEC AVGCPU
N=COUNT ;

PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC SORT DATA=S; BY DATE FUNCTION;

PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
TITLE1"DFHS MIGRATION ACTIVITY - SUMMARY";
ID DATE FUNCTION;
VAR COUNT AVGRESP AVGPEND AVGCPU
   AVGBM AVGSIZE AVGCOMP AVGKBSEC;
FORMAT AVGRESP AVGPEND AVGCPU 6.2 ;
   LABEL AVGBM = 'AVG MB*XFERED*(READ + WRITE)' FUNCTION = 'DFHS*MFUNCTION'
DATE = 'DATE OF REQUEST'
AVGSIZE = 'AVG DS*SIZE*IN MB'
AVGCOMP = 'AVG*COMPRESS*RATIO'
AVGCPU = 'AVG*CPU*TIME(SEC)'
AVGRESP = 'AVG*ELAPSED*TIME(SEC)'
AVGPEND = 'AVG*PENDING*TIME(SEC)'
AVGKBSEC = 'AVG*TRANSFER RATE*(KB/SEC)'
FUNCTION = 'DFHS*FUNCTION'
COUNT = 'DATASETS*PROCESSED';

*--- DELETION OF MIGRATED DATASETS ---;

DATA DELMIG; SET OK;
IF FSRTYPE = 6;
IF FSRNENT1 GT 0 THEN FSRFVOL = TAPEVOL;
IF FSRTFLOG = '1.......B THEN KBFREE = FSRTBYBK;
   ELSE KBFREE = (FSRTBYBK / 1024);

IF FSRTFLAGS = '...1....B THEN KFREE = FSRTBYR;
   ELSE KFREE = (FSRTBYR / 1024);
IF KFREE EQ 0 THEN KFREE = KBFREE;
LABEL KFREE = 'KB FREED';

*--- DELETION OF MIGRATED DATASETS DETAIL REPORT ---;

PROC SORT DATA=DELMIG NODUP; BY DATE FSRTIMR FSRTYPE;
PROC PRINTTO PRINT=MIGDELEX; OPTIONS PAGENO=1;
PROC PRINT DATA=DELMIG UNIFORM NOOBS SPLIT=’*’;
VAR FSRTIMR FUNCTION FSRFVOL KFREE
FSRAGE FSRTMCL FSRTSN;
BY DATE;
TITLE1"DELETE MIGRATED DATASETS - DETAIL";

*--- DELETION OF MIGRATED DATASETS SUMMARY REPORT ---;

PROC SUMMARY DATA=DELMIG NWAY;
   CLASS FUNCTION DATE;
   VAR DURATION PENDING KFREE FSRCPU;
   OUTPUT OUT=S
   MEAN(DURATION PENDING KFREE FSRCPU)=
   AVGRESP AVGPEND AVGFR AVGCPU
   SUM(KFREE)= TOTFR
   N=COUNT;

PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT=’*’;
TITLE1"DELETE MIGRATED DATASETS - SUMMARY";
VAR DATE FUNCTION COUNT AVGFR TOTFR AVGRESP AVGPEND AVGCPU;
FORMAT AVGRESP AVGPEND AVGCPU 5.2;
LABEL AVGRESP = 'AVG*ELAPSED*TIME(SEC)' AVGPEND = 'AVG*PENDING*TIME(SEC)' AVGCPU = 'AVG*CPU*TIME(SEC)' FUNCTION = 'DFHSM*FUNCTION'
DATE = 'DATE OF REQUEST'
COUNT = 'DATASETS*PROCESSED'
TOTFR = 'TOTAL KB*FREED'
AVGFR = 'AVG*KB FREED';

*---- EXPIRATION OF PRIMARY OR MIGRATED DATASETS ----;

DATA EXPMIG; SET OK;
IF FSRTYPE =17;
IF FSRNENT1 GT 0 THEN FSRFVOL =TAPEVOL;

IF FSRTFLGS = '1.......'B THEN KBFREE = FSRTBYBK;
   ELSE KBFREE = (FSRTBYBK / 1024);
IF FSRTFLGS = '....1....'B THEN KFREE = FSRBYTR;
   ELSE KFREE = (FSRBYTR / 1024);
K = 55;  /* DASD TRACK SIZE IN KB - DEVICE DEPENDENT */
IF FSRTKR GE 0 THEN KFREE=(FSRTKR*K);
IF KFREE EQ 0 THEN KFREE=KBFREE;
LABEL KFREE = 'KB FREED'
EXPIRE = 'DATASET*TYPE' ;

*-- EXPIRATION OF PRIMARY/MIGRATED DATASETS DETAIL REPORT --;

PROC SORT DATA=EXPMIG NODUP; BY DATE FSRTIMR EXPIRE;
PROC PRINTTO PRINT=MIGDELEX; OPTIONS PAGENO=1;
PROC PRINT DATA=EXPMIG UNIFORM NOOBS SPLIT="*";
   VAR FSRTIMR EXPIRE FSRFVOL
      KFREE FSRRAGE FSRMGTCL FSRDSN ;
BY DATE;
TITLE1"EXPIRED PRIMARY/MIGRATED DATASETS - DETAIL";

PROC SUMMARY DATA=EXPMIG NWAY;
   CLASS EXPIRE DATE;
VAR DURATION PENDING FSRRAGE KFREE FSRCPU;
OUTPUT OUT=S
MEANC(DURATION PENDING FSRRAGE KFREE FSRCPU)=
   AVGRESP AVGPEND AVGAGE AVGFR AVGCPU
SUM(KFREE) = TOTFR
N=COUNT ;

*---- EXPIRATION OF PRIMARY/MIGRATED DATASETS SUMMARY REPORT ----;

PROC SORT DATA=S; BY DATE EXPIRE;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOODB SPLIT="*";
TITLE1"EXPIRED PRIMARY/MIGRATED DATASETS - SUMMARY";
VAR DATE EXPIRE COUNT AVGFR TOTFR AVGAGE AVGRESP AVGPEND AVGCPU;
FORMAT AVGRESP AVGPEND AVGCPU 5.2;
FORMAT AVGAGE 3.;
LABEL EXPIRE = 'DATASET*TYPE'
  COUNT = 'DATASETS*PROCESSED'
  TOTFR = 'TOTAL KB* Freed'
  AVGFR = 'AVG*KB Freed'
  AVGAGE = 'AVG AGE*OF DS'
  AVGCPU = 'AVG*CPU*TIME'
  DATE = 'DATE OF REQUEST'
  AVGRESP = 'AVG*ELAPSED*TIME(SEC)'
  AVGPEND = 'AVG*PENDING*TIME(SEC)';

*---- PARTIAL RELEASE -------; 
DATA PAR; SET OK;
IF FSRTYPE =18;
LABEL FSRTKR = 'TRACKS*Freed';
PROC SORT DATA=PAR; BY DATE FUNCTION;

*---- PARTIAL RELEASE DETAILED REPORT -------;
PROC PRINTTO PRINT=MIGDELEX; OPTIONS PAGENO=1;
PROC PRINT DATA=PAR UNIFORM NOOB S SPLIT="*";
VAR FSRTMR FUNCTION FSRAGE
  FSRTKR FSRMGTCF FSRFVOL FSRSBN;
BY DATE;
TITLE1"PARTIAL RELEASE OF UNUSED SPACE - DETAILED";
PROC SUMMARY DATA=PAR NWAY;
  CLASS FUNCTION DATE;
VAR FSRTKR;
OUTPUT OUT=S MIN(FSRTKR) = MINTRK
  MEAN(FSRTKR) = AVGTRK
  MAX(FSRTKR) = MAXTRK
  SUM(FSRTKR) = SUMTRK
N=COUNT;

*---- PARTIAL RELEASE SUMMARY REPORT -------;
PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
VAR DATE FUNCTION COUNT SUMTRK MINTRK AVGTRK MAXTRK;
TITLE1"PARTIAL RELEASE OF UNUSED SPACE - SUMMARY";
LABEL COUNT = 'DATASETS*PROCESSED'
SUMTRK = 'TOTAL TRACKS*FREED'
DATE = 'DATE OF REQUEST'
FUNCTION= 'DFHSM*FUNCTION'
MINTRK = 'MIN TRACKS*FREED'
AVGTRK = 'AVG TRACKS*FREED'
MAXTRK = 'MAX TRACKS*FREED';

*----- RECALLING MIGRATED DATASETS -----;

DATA RECALL; SET OK;
MIG=PUT(FSRTYPE,MIGRAT.);
IF MIG='94';
TP=PUT(FSRTYPE,TPT.);
IF TP='97' THEN FSRFVOL=TAPEVOL;

IF FSRFLAGS = '....1....'B THEN KWRITE = FSRBYTW;
   ELSE KWRITE = (FSRBYTW / 1024);

IF FSRTYPE = 4 THEN DO; /* L1--PR */
   IF FSRFLAGS = '....1....'B THEN KBREAD = FSRBYTR;
   ELSE KBREAD = (FSRBYTR / 1024);
   END;

IF FSRTYPE = 5 THEN DO; /* L2---PR */
   IF FSRTFLGS = '1........'B THEN KBREAD = FSRTBYBK;
   ELSE KBREAD = (FSRTBYBK / 1024);
   END;

IF FSRLDM1 < 2 THEN FSRLDM = FSRLDM + 1900000;
   DLM = INPUT(PUT(FSRLDM,7.),JULIAN7.);

MBREAD=(KBREAD/1024); MBWRITE=(KWRITE/1024);
TOTKB=KBREAD+KWRITE; COMPR=KWRITE/KBREAD;
XFERTIME=DURATION - PENDING;
KBSEC=TOTKB/XFERTIME; TOTMB=(KBREAD+KWRITE)/1024;
LABEL KWRITE='DS SIZE*IN KB'
   KBSEC = 'TRANSFER RATE'
   FSRTIMR='RECALL TIME'
   FSRAJE = 'DAYS ON*ML1/ML2';

PROC SORT DATA=RECALL NODUP; BY DATE FUNCTION;

*---- RECALL DETAIL ----;

PROC PRINTTO PRINT=MIGREC; OPTIONS PAGENO=1;
PROC PRINT DATA=RECALL UNIFORM NOOBS SPLIT="*";
TITLE1'"DFHSM RECALL ACTIVITY"';
VAR FSRTIMR FUNCTION FSRAJE FSRMGTCL
   KBWRITE KBSEC FSRFVOL FSRTVOL FSRDSN;
BY DATE;

PROC SORT DATA=RECALL NODUP;
   BY DATE FSRTIMR FUNCTION;

*---- RECALL SUMMARY ----;

PROC SUMMARY DATA=RECALL NWAY;
   CLASS FUNCTION DATE;
VAR DURATION PENDING TOTMB MBWRITE COMPR KBSEC FSRCPU;
OUTPUT OUT=S
MEAN(DURATION PENDING TOTMB MBWRITE COMPR KBSEC FSRCPU)=
   AVGRESP AVGPEND AVGMB AVGSIZE AVGCOMP AVGKBSEC AVGCPU
N=COUNT ;

PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
TITLE1"DFHSM RECALL ACTIVITY - SUMMARY";
VAR DATE FUNCTION COUNT AVGRESP AVGPEND AVGCPU
   AVGMB AVGSIZE AVGCOMP AVGKBSEC;
FORMAT AVGRESP AVGPEND AVGCPU 6.2 ;
LABEL AVGMB = 'AVG MB*XFERED*(READ + WRITE)' 
   AVGSIZE = 'AVG DS*SIZE*IN MB' 
   AVGCOMP = 'AVG*COMPRESS*_RATIO' 
   AVGCPU = 'AVG*CPU*TIME' 
   DATE = 'DATE OF REQUEST' 
   FUNCTION = 'DFHSM*FUNCTION' 
   COUNT = 'DATASETS*PROCESSED' 
   AVGKBSEC='AVG*TRANSFER RATE*(KB/SEC)'
   AVGRESP='AVG*ELAPSED*TIME(SEC)'
   AVGPEND='AVG*PENDING*TIME(SEC)';

*---------- FSR STATISTICS ---- ----;

PROC PRINTTO PRINT=FSRSTAT; OPTIONS PAGENO=1;

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES ELAPS/NOCUM;
TITLE1"FSR RECORDS BY TIME COMPLETED (SECONDS)";
PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES FUNCTION*ELAPS/NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY FUNCTION TYPE AND COMPLETED TIME (SECONDS)";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES XP/NOCUM;
TITLE1"FSR RECORDS BY TIME WITHOUT DELAY (SECONDS)";
PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES FUNCTION*XP/NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY FUNCTION TYPE AND EFFECTIVE TIME (SECONDS)";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES REQ/NOCUM;
TITLE1"FSR RECORDS BY REQUEST";
PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES FUNCTION*REQ/NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY FUNCTION - REQUEST";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES AGE/NOCUM;
TITLE1"FSR RECORDS BY DATASET AGE";
PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES FUNCTION*AGE /NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY FUNCTION AND DATASET AGE";

PROC FREQ DATA=OK;
   TABLES DATE / NOCUM;
FORMAT DATE WEEKDATE.;
TITLE1"FSR RECORDS BY DATE";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES TIME/NOCUM;
TITLE1"FSR RECORDS BY 1 HOUR SLOT";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES EXREDU/NOCUM;
TITLE1"FSR RECORDS BY EXTENT REDUCTION";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES EXPIRE/NOCUM;
TITLE1"FSR RECORDS BY EXPIRE";

PROC PRINTTO PRINT=MGMTCLAS; OPTIONS PAGENO=1;
PROC FREQ DATA=OK;
   TABLES FSRMGTCL/MISSING NOCUM;
TITLE1"FSR RECORDS BY MGMT CLASS";

PROC FREQ DATA=OK;
   TABLES FUNCTION*FSRMGTCL/MISSING NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY MGMT - FSRTYPE";

PROC FREQ DATA=OK ORDER=FORMATTED;
   TABLES FSRMGTCL*AGE/MISSING NOFREQ NOPERCENT NOCUM;
TITLE1"FSR RECORDS BY MGMT CLASS - DATASET AGE";

*------- RECYCLING ---------;

DATA RECY; SET OK;
IF FSRTYPE =10 OR FSRTYPE =12 ;
   XFERTM =DURATION - PENDING;

34 © 2002, Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
FSRFVOL =TAPEVOL;       TOTKB =FSRTBYBK*16;
MBREAD = (TOTKB/1024);   KBSEC = TOTKB/ XFERTM;

PROC SUMMARY DATA=RECY NWAY;
   CLASS FUNCTION DATE;
VAR DURATION PENDING MBREAD KBSEC FSRCPU;
OUTPUT OUT=S
MEAN(DURATION PENDING MBREAD KBSEC FSRCPU)=
   AVGRESP AVGPEND AVGSIZE AVGKBSEC AVGCPU
N=COUNT;

*---- RECYCLE SUMMARY ----;

PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
TITLE1 "DFHSM RECYCLE ACTIVITY - SUMMARY";
VAR DATE FUNCTION COUNT AVGRESP AVGPEND AVGCPU
   AVGSIZE AVGKBSEC;
FORMAT AVGRESP AVGPEND AVGCPU 7.2;
LABEL AVGSIZE = 'AVG DS*SIZE*IN MB'
   AVCPU = 'AVG*CPU*TIME'
   COUNT = 'DATASETS*PROCESSED'
   AVGKBSEC = 'AVG*TRANSFER RATE*(KB/SEC)'
   AVGRESP = 'AVG*ELAPSED*TIME(SEC)'
   AVGPEND = 'AVG*PENDING*TIME(SEC)';

*---- RECYCLE DETAIL ----;

PROC SORT DATA=RECY; BY DATE FSRTIMR;
PROC PRINTTO PRINT=RECYCLE; OPTIONS PAGENO=1;
PROC PRINT DATA=RECY UNIFORM NOOBS SPLIT="*";
TITLE1 "DFHSM RECYCLE ACTIVITY - DETAIL";
VAR FSRTIMR FUNCTION FSRCPU
   TOTKB XFERTM TAPEVOL FSRAGE FSRSN;
   BY DATE;
LABEL DATE = 'RECYCLE*DATE'
   FSRTIMR = 'RECYCLE*TIME'
   FSRCPU = 'CPU TIME*USED'
   TAPEVOL = 'RECYCLED*VOLUME'
   TOTKB = 'KB READ'
   XFERTM = 'XFER TIME*(IN SEC)'
   KBSEC = 'XFER RATE*(KB/SEC)';

*----- BACKUP: DAILY & SPILL ------;

DATA BACKUP; SET OK;
   MIG=PUT(FSRTYPE,MIGRAT.);
IF MIG='93' ;       /* BACKUP: DAILY, SPILL */
   TP=PUT(FSRTYPE,TPT.);
IF TP='98' THEN FSRTVOL=TAPEVOL ;    /* BACKUP TAPEOUT */
IF FSRFLAGS = '...1....'B THEN KBREAD = FSRBYTR;
    ELSE KBREAD = (FSRBYTR / 1024);
IF FSRTFLGS = '1....1'B THEN KBWRITE = FSRTBYBK;
    ELSE KBWRITE = (FSRTBYBK / 1024);

MBREAD=(KBREAD/1024);   MBWRITE=(KBWRITE/1024);
TOTKB=KBREAD+KBWRITE;   COMPR=KBREAD/KBWRITE;
XFERTIME= DURATION - PENDING;
KBSEC=TOTKB/XFERTIME;   TOTMB=(KBREAD+KBWRITE)/1024;

PROC SORT DATA=BACKUP NODUP ; BY DATE FSRTIMR FUNCTION;

DATA NOTBK; SET BACKUP; IF KBWRITE GT 0 THEN DELETE;
DATA BACK; SET BACKUP; IF KBWRITE=. THEN DELETE;
IF KBWRITE GT KBREAD THEN
    KBWASTE = KBWRITE-KBREAD;
    BLOCKS=KBWRITE/16;
LABEL DATE = 'BACKUP*DATE'
    FSRTIMR='BACKUP*TIME '
    FSRCPU = 'CPU TIME*USED'
    KBREAD = 'DATASET*SIZE(KB)'
    KBWASTE='KB WASTED'
    BLOCKS = '16K OUTPUT*BLOCKS*WRITTEN'
    KBSEC = 'XFER RATE*(KB/SEC)' ;

*---- BACKUP DETAIL -----

PROC SORT DATA=BACK; BY DATE FSRTIMR;
PROC PRINTTO PRINT=BKRCVY; OPTIONS PAGENO=1;
PROC PRINT DATA=BACK UNIFORM NOOBS SPLIT="*";
    VAR FSRTIMR FUNCTION FSRCEN FSRFVOL FSRTVOL
    KBREAD KBWASTE BLOCKS KBSEC FSRDSN ;
    BY DATE;
TITLE1"BACKED UP DATASETS - DETAILED REPORT ";

*---- BACKUP SUMMARY -----

PROC SUMMARY DATA=BACK NWAY;
    CLASS FUNCTION DATE;
    VAR DURATION KBWASTE TOTMB BLOCKS COMPR KBSEC FSRCPU;
    OUTPUT OUT=S
    MEAN(DURATION TOTMB BLOCKS COMPR KBSEC FSRCPU)=
        AVGRESP AVGMB AVGBLK AVGCOMP AVGKBSEC AVGCPU
    SUM(KBWASTE)=TOTWAST
    N=COUNT ;

PROC SORT DATA=S ; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
VAR DATE FUNCTION COUNT AVGRESP AVGCPU
   AVGMB TOTWAST AVGBLK AVGCOMP AVGKBSEC;
FORMAT AVGRESP AVGCPU 7.2 ;
FORMAT AVGMB 5.1;
LABEL AVGM  ='AVG MB*XFERED*(READ + WRITE)'
COUNT    ='DATASETS*PROCESSED'
AVGRESP  ='AVG*ELAPSED*TIME(SEC)'
AVG CPU  ='AVG*CPU*TIME'
TOTWAST  ='TOTAL*KB WASTED'
AVG BLK  ='AVG NO.OF*16 KB BLOCKS*WRITTEN'
AVGC MP  ='AVG*COMPRESS*RATIO'
AVG KBSEC='AVG*TRANSFER RATE*(KB/SEC)';
TITLE1"DFHSM BACKUP ACTIVITY - SUMMARY";

PROC SORT DATA=NOTBK; BY DATE FSRTIMR;
PROC PRINTTO PRINT=BKRCVVY; OPTIONS PAGENO=1;
PROC PRINT DATA=NOTBK UNIFORM NOOBS SPLIT="*";
   VAR FSRTIMR FUNCTION FSRA GE FSRD SN ;
   BY DATE;
TITLE1"DATASETS NOT BACKED UP ";

*--- EXPIRED INCREMENTAL BACKUP DATASETS -----

DATA EXPB K; SET OK;
IF FSRTY PE =19;
IF FSRTNENT1 GT 0 THEN FSRFVOL =TAPE VOL;

*--- EXPIRED INCREMENTAL BACKUP DATASETS DATAIL -----

PROC SORT DATA=EXPB K NODUP; BY DATE FSRTIMR EXPIRE;
PROC PRINTTO PRINT=BKDELEX; OPTIONS PAGENO=1;
PROC PRINT DATA=EXPB K UNIFORM NOOBS SPLIT="*";
   VAR FSRTIMR FSRFVOL  FSRTGEN
   FSRA GE FSRMGTL C FSRD SN ;
   BY DATE;
TITLE1"EXPIRED INCREMENTAL BACKUP DATASETS - DETAIL";

PROC SUMMARY DATA=EXPB K NWAY;
   CLASS FUNCTION DATE;
   VAR FSRA GE ;
OUTPUT OUT=S MIN(FSRA GE) = MINAGE
   MEAN(FSRA GE) = AVGAGE
   MAX(FSRA GE) = MAXAGE
   N=COUNT ;

PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
   VAR DATE FUNCTION COUNT MINAGE AVGAGE MAXAGE;
TITLE1"EXPIRED INCREMENTAL BACKUP DATASETS - SUMMARY";

LABEL COUNT = 'DATASETS*PROCESSED'
MINAGE = 'MINIMAL*DS AGE'
AVGAGE = 'AVG AGE'
MAXAGE = 'MAXIMAL*DS AGE';

*---- DELETED INCREMENTAL BACKUP DATASETS ----;

DATA DELBK; SET OK;
IF FSRTYPE = 20;
IF FSRTNENT1 GT 0 THEN FSRFVOL = TAPEVOL;

PROC SORT DATA=DELBK NODUP; BY DATE FSRTIMR EXPIRE;
PROC PRINTTO PRINT=BKDELEX; OPTIONS PAGENO=1;
PROC PRINT DATA=DELBK UNIFORM NOOBS SPLIT='*';
   VAR FSRTIMR FSRTGEN FSRAGE FSRFVOL FSRMGTCI FSRDSN;
   BY DATE;
TITLE1 "DELETED INCREMENTAL BACKUP DATASETS - DETAIL";

PROC SUMMARY DATA=DELBK NWAY;
   CLASS FUNCTION DATE;
   VAR FSRAGE;
   OUTPUT OUT=S
      MIN(FSRAGE) = MINAGE
      MEAN(FSRAGE) = AVGAGE
      MAX(FSRAGE) = MAXAGE
N=COUNT;

PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
   VAR DATE FUNCTION COUNT MINAGE AVGAGE MAXAGE;
LABEL COUNT = 'DATASETS*PROCESSED'
MINAGE = 'MINIMAL*DS AGE'
AVGAGE = 'AVG AGE'
MAXAGE = 'MAXIMAL*DS AGE';
TITLE1 "DELETED INCREMENTAL BACKUP DATASETS - SUMMARY";

*-- RECOVERING BACKUP DATASETS ------;

DATA RECOV; SET OK;
IF FSRTYPE = 9; /* RECOVERY */
TP=PUT(FSRTYPE,TPT.);
IF TP='97' THEN FSRFVOL=TAPEVOL; /* RECOVERY TAPEIN */

IF FSRFLAGS = '....1....'B THEN KBWRITE= FSRBYTW;
   ELSE KBWRITE = (FSRBYTW / 1024);
IF FSRTFLGS = '1........'B THEN KBREAD = FSRTBYBK;
   ELSE KBREAD = (FSRTBYBK / 1024);

MBREAD=(KBREAD/1024); MBWRITE=(KBWRITE/1024);
TOTKB=KBREAD+KBWRITE; COMPR=KBREAD/KBWRITE;
XFERTIME= DURATION - PENDING;
KBSEC=TOTKB/XFERTIME; TOTMB=(KBREAD+KBWRITE)/1024;

PROC SORT DATA=RECOV NODUP; BY DATE FSRTIMR FUNCTION;
PROC PRINTTO PRINT=BKRCVY; OPTIONS PAGENO=1;
PROC PRINT DATA=RECOV UNIFORM NOOBS SPLIT="*";
VAR FSRTIMR FSRAGE FSRFVOL FSRTVOL
FSRMGTCB KBWRITE KBSEC FSRSN ;
BY DATE;
LABEL KBWRITE = 'DATASET * SIZE(KB)' KBSEC = 'XFER RATE * (KB/SEC)' ;
TITLE1 "RECOVERED DATASETS - DETAILED REPORT ";

*---- RECOVERY SUMMARY -----

PROC SUMMARY DATA=RECOV NWAY;
CLASS FUNCTION DATE;
VAR DURATION PENDING TOTMB MBWRITE COMPR KBSEC FSRCPU;
OUTPUT OUT=S
MEAN(DURATION PENDING TOTMB MBWRITE COMPR KBSEC FSRCPU)=
AVGRESP AVGPEND AVGMB AVGFSIZE AVGCMP AVGKBSEC AVGCPU
N=COUNT ;

PROC SORT DATA=S; BY DATE FUNCTION;
PROC PRINTTO PRINT=FUNSUMM; OPTIONS PAGENO=1;
PROC PRINT DATA=S UNIFORM NOOBS SPLIT='*';
TITLE1 "DFHSM RECOVERY ACTIVITY - SUMMARY";
VAR DATE FUNCTION COUNT AVGRESP AVGCPU
AVGMB AVGFSIZE AVGCMP AVGKBSEC;
FORMAT AVGRESP AVGPEND AVGCPU 8.2 ;
LABEL AVGMB = 'AVG MB * XFERED *(READ + WRITE)'
AVGFSIZE = 'AVG DS * SIZE * IN MB '
AVGCMP = 'AVG * COMPRESS * RATIO '
COUNT = 'DATASETS * PROCESSED' AVGRESP = 'AVG * ELAPSED * TIME (SEC)' AVGCPU = 'AVG * CPU * TIME (SEC)'
AVGKBSEC = 'AVG * TRANSFER RATE * (KB/SEC)' ;

*---- TAPE USAGE SUMMARY -----

DATA TAPES; SET OK;
IF FSRNENT1 LT 1 THEN DELETE;

PROC SUMMARY DATA=TAPES NWAY;
CLASS TAPEVOL DATE;
BY FSRTYPE;
VAR DURATION PENDING FSRCPU;
OUTPUT OUT=K
MEAN(DURATION PENDING FSRCPU)= AVGRESP AVGPEND AVGCPU
N=COUNT;

DATA SUMMARY; SET K;
    FUNCTION=PUT(FSRTYPE,FSRFMT.);

PROC SORT DATA=SUMMARY; BY DATE FUNCTION TAPEVOL;
PROC PRINTTO PRINT=TAPES; OPTIONS PAGENO=1;
PROC PRINT DATA=SUMMARY UNIFORM NOOB SPLIT="*";
    VAR TAPEVOL FUNCTION COUNT AVGRESP AVGPEND AVGCPU;
    BY DATE;
LABEL TAPEVOL = 'TAPE*VOLSER ';
    AVGRESP = 'AVG*ELAPSED*TIME(SEC)'
    AVGPEND = 'AVG*PENDING*TIME(SEC)'
    FUNCTION = 'DFHSM*FUNCTION'
    COUNT = 'DATASETS*PROCESSED'
    AVGCPU = 'AVG CPU*TIME(SEC)'
    TITLE1"TAPES USED REPORT";

*----- DUMP VOLUME SUMMARY ------;

DATA DUMP; SET OK;
    IF FSRTYPE =13;
PROC PRINTTO PRINT=DUMPS; OPTIONS PAGENO=1;
PROC PRINT DATA=DUMP UNIFORM NOOB SPLIT="*";
    TITLE1"DUMP VOLUME REPORT";
    VAR DATE FSRTIMR FSRFVOL TAPEVOL
    DURATION FSRDCOPR FSRDCOPF FSRDCL1;

*----- THRASHING ANALYSIS --------;

DATA TRAS; SET OK;
    MIG=PUT(FSRTYPE,MIGRAT.);
    IF MIG='95' OR MIG='94';
    IF MIG='95' THEN FUNCTION='MIGRATION';
    ELSE FUNCTION='RECALL';
PROC SORT DATA=TRAS; BY DATE FSRTIMR;
PROC PRINTTO PRINT=FSRSTAT; OPTIONS PAGENO=1;
PROC FREQ DATA=TRAS ORDER=FORMATTED;
    TABLES FUNCTION;
    TITLE1"THRASHING ANALYSIS";
PROC FREQ DATA=TRAS ORDER=FORMATTED;
    TABLES DATE*FUNCTION/NOFREQ NOPERCENT NOCUM NOCOL;
    TITLE1"THRASHING ANALYSIS BY DATE ";
PROC PRINTTO PRINT=MGMTCLAS; OPTIONS PAGENO=1;
PROC FREQ DATA=TRAS ORDER=FORMATTED;
    TABLES FSRMGCL*FUNCTION/NOFREQ NOPERCENT NOCUM NOCOL;
    TITLE1"THRASHING ANALYSIS BY MGT. CLASS ";

Mile Pekic
Systems Programmer (Yugoslavia)  © Xephon 2002
Editing a new member when in Option 3.4

Personally I very seldom use Option 3.1 (Library Utility) to access the members of a dataset, preferring to use Option 3.4 (Data Set List Utility) instead. For several years, because I didn’t know of another way and there is no menu option available under 3.4, I have used a strange method to edit an empty member.

The method I used was to go into an existing member and use the line command `c` (copy) together with the main command `CREATE newname` to create a new member. This new member was then selected with another 3.4 session and the extraneous text in the member deleted.

In a recent seminar, I learnt that it is possible to edit a new member directly from the Data Set List by typing:

```
e / (newname)
```

in the line command position for the selected dataset – see Figure 1.

---

Figure 1: Edit a new member

---

Rolf Parker  
Systems Programmer (Germany)  
© Xephon 2002
Get current operating system version

PROBLEM ADDRESSED
In situations where operating systems are being installed and tested, the problem sometimes arises of the current operating system version having to be identified in order to adapt jobs appropriately. This functionality can also be required when an application has to adapt its processing depending on the operating system currently running.

SOLUTION
The GETOSVR program obtains the current operating system version and release from the Extended Communications Vector Table (ECVT). The program displays this information (for simplicity the TPUT service is used; this means that the display is made only in TSO – the WTO service with routing code 11 could be used to output the display on the job log). The program returns a hash code formed from the operating system name, the version, and the release. The application can use this returned hash code to perform the appropriate processing.

EXAMPLES
The following list shows the output for some recent operating systems:

- Hash code for OS/390 V2.8: 1819 (X'71B'):
  
  05/390 020800

- Hash code for OS/390 V2.10: 1555 (X'613'):
  
  05/390 021000

- Hash code for z/OS V1.2: 3968 (X'F80'):
  
  z/OS 010200

The hash code is a numeric value within the range 0-4095 that the program sets as a return code.
PROGRAM CODE

TITLE 'Get OS Name and Version'
* Display operating system name and version, eg OS/390 021000
* Return hash code formed from product name (bytes 0-3), version, and
* release
GETOSVR CSECT
GETOSVR AMODE 31
GETOSVR RMODE 24 RMODE(24) required for TPUT
  BAKR 14,Ø Save registers and return address
  BASR 12,Ø Set base register
USING *12
SPACE
  L 1,16 A(CVT)
USING CVT,1
* R1: Address of Communications Vector Table
  L 2,CVTECVT A(ECVT)
USING ECVT,2
* R2: Address of Extended Communications Vector Table
  MVC PNAMEVRL,ECVTPTNAM
  LA 1,PNAMEVRL
  LA Ø,L'PNAMEVRL
  TPUT (1),Ø
* Form hash code from product name (bytes 0-3), version and release
  XC HASH,HASH Clear <hash>
  XC HASH,PRODNAME+Ø Bytes 0-1
  XC HASH,PRODNAME+2 Bytes 2-3
  XC HASH,PRODVERS Version
  XC HASH,PRODREL Release
  NC HASH,=X'FFF' Modulo(4096)
  LH 15,HASH
PR , Program return
SPACE
HASH DS H
SPACE
PNAMEVRL DS ØCL22
PRODNAME DS CL16 Product (Operating System) Name
PRODVERS DS CL2 Product Version
PRODREL DS CL2 Product Release
DS CL2 Product Modification Level
SPACE
CVT DSECT=YES
IHAECVT
END

PROGRAM INVOCATION

The program does not have any files or parameters.
OS/390 (z/OS) provides a number of directed subsystem interface (SSI) function calls that are available with the IEFSSREQ macro call. SSI function code 1 (process SYSOUT datasets) has existed for many years and was/is used by external writer interfaces to allow a program to access JES SYSOUT datasets independently of traditional JES functions such as print. Although this is a powerful interface, it has some limitations:

- Certain functionality is only available if the external writer program is running as a started task.
- For JES2 environments, the external writer program is unable to select spooled SYSOUT data that is directed to held output classes.

The more recent SSI function code 79 (SYSOUT Application Programming Interface – SAPI) is a more robust implementation that overcomes many of the restrictions inherent in SSI function code 1. Unlike SSI function code 1, the SSI 79 interface supports multiple threads and allows for SYSOUT dataset selection that goes beyond the scope of SSI 1 external writers.

External writers serve many useful purposes. They are well suited for managing SYSOUT data not processed by JES directly. Many third-party vendor products use the external writer interface – particularly report management and distribution products. The SAPISSI program provided with this article functions very much like a traditional SSI function code 1 external writer except that it can run as a batch job (or started task) and still get automatic notification of new SYSOUT output that has been created that matches its selection criteria, and it can extract SYSOUT output that is directed to JES2 held output classes.

The SAPISSI program does nothing more than collect output datasets that have been directed to the spool in the output classes that the external writer has been directed to collect output for. To initialize the external writer, you need to start a batch job or started task that specifies the external writer program name, the parameter-supplied external writer name, and an optional output class list. Here are a few examples:
The first example would start external writer TESTWTR, which will collect SYSOUT data for all output classes (no class list is specified in the program parameters so the default is to collect data for all classes). The second example would start external writer TESTWTR2, which will collect SYSOUT data for only classes AGRYU49.

You can direct output to an external writer using the SYSOUT parameter on a JCL DD statement. Here is an example:

```
//OUTDD DD SYSOUT=(A,TESTWTR)
```

The above example would direct its output to an external writer with a name of TESTWTR. The TESTWTR external writer would process the output providing it was currently set up to collect class A output.

The external writer provided with this article has an operator command interface that supports two commands. The first command is a modify command that can be used to modify the class list that is currently being monitored by the external writer. The command has the following format:

```
F xwtr,CLASSLIST=classlst
```

where ‘xwtr’ is the name of your external writer started task or batch job, and ‘classlst’ is the list of classes you want the external writer to be managing. The ‘classlst’ can consist of a list of up to 36 alphanumeric characters (A-Z and 0-9) or a single ‘*’. A single asterisk indicates that every class is eligible for processing by this external writer.

The second operator command that is supported by the external writer is the stop command. It has the following format:

```
P xwtr
```

where ‘xwtr’ is the name of your external writer started task or batch job. This command will cause the termination of the external writer.
The SAPISSI external writer program should be link-edited into an authorized load library using the following link-edit control cards:

```
INCLUDE OBJECT(SAPISSI)
ENTRY SAPISSI
SETCODE AC(1)
NAME SAPISSI(R)
```

A SAPISSI external writer batch job or started task can be initiated with the following sample JCL:

```
//SAPISSI EXEC PGM=SAPISSI,PARM='TESTWTR1'
//STEP1B DD DSN=authorized.library,DISP=SHR
//OUTPUT DD DSN=xwtr.output,DISP=SHR
//SYSPRINT DD SYSOUT=* 
```

The SAPISSI external writer expects an OUTPUT DD statement to be used as a collection dataset for the target data. Since you cannot be sure what the characteristics of the source data will be, I recommend that the OUTPUT dataset used by the external writer be defined as RECFM=VB, LRECL=32765, BLKSIZE=32760, although the external writer will support RECFM=FB files and SYSOUT as well. The SYSPRINT DD dataset is used for messages that are produced from the external writer itself.

All output collected by the external writer is dumped to the OUTPUT dataset. Each independent output dataset is identified in the collection OUTPUT dataset by a separator record that has the following format:

```
JBN: jobname  JB#: jobnum  PRC: prcname  STP: stpnam  DDN: ddname  
CLS: c  DSN: dsn
```

This allows you to determine the various different output datasets that have been dumped to the collector OUTPUT dataset. If you choose, additional information could be included in the separator record such as the system name that the job ran on, the JES member the job ran on, the time of day the job ran, etc. Have some fun deciding what your separator record will look like.

The SAPISSI program is meant to provide a very simple example of an external writer and some of its capabilities. I hope this article provides some interesting insights.
SAPISSI ASM

SAPISSI CSECT
SAPISSI AMODE 31
SAPISSI RMODE 24

* THIS PROGRAM DEMONSTRATES THE USE OF THE SYSDOUT APPLICATION *
* PROGRAM INTERFACE (SAPI) SUBSYSTEM FUNCTION CALL (SSI 79) AND *
* PROVIDES A VERY BASIC EXAMPLE OF AN OS/390 EXTERNAL WRITER. *

* THE SAPI SSI 79 FUNCTION CALL CAN BE USED TO EXTRACT SYSDOUT DATA *
* FROM THE JES SPOOL. THE INTERFACE PROVIDES FOR A NUMBER OF *
* DIFFERENT SELECTION CRITERIA OPTIONS OF WHICH ONLY A FEW ARE USED *
* BY THIS PARTICULAR PROGRAM. A COMPLETE DESCRIPTION OF THE SAPI *
* SELECTION CRITERIA CAN BE FOUND IN 'OS/390 MVS USING THE *
* Subsystem Interface' OR BY EXAMINING THE IAZSSS2 CONTROL BLOCK *
* MAPPING MACRO.

* THIS PROGRAM SHOULD BE LINK-EDITED INTO AN AUTHORIZED DATASET *
* USING THE FOLLOWING LINK-EDIT CONTROL CARDS:

* INCLUDE OBJECT(SAPISSI)
* ENTRY SAPISSI
* SETCODE AC(1)
* NAME SAPISSI(R)

* THE PROGRAM CAN BE RUN AS FOLLOWS:

* //STEP1 EXEC PGM=SAPISSI,PARM='wtrnm,claslist'
* //STEP1B DD DSN=authorized.library,DISP=SHR
* //OUTPUT DD SYSOUT=* 
* //SYSPRINT DD SYSOUT=* 

* WHERE 'wtrnm' IS THE EXTERNAL WRITER NAME THAT WILL BE USED BY *
* THIS PROGRAM AND 'claslist' IS THE LIST OF JES OUTPUT CLASSES *
* THAT WILL BE PROCESSED BY THIS EXTERNAL WRITER. 'wtrnm' IS *
* REQUIRED BUT THE 'claslist' PORTION OF THE PARM IS OPTIONAL. IF *
* 'claslist' IS OMITTED, ALL OUTPUT CLASSES ARE ELIGIBLE FOR *
* SELECTION. HERE ARE SOME VALID EXAMPLES:

* //STEP1 EXEC PGM=SAPISSI,PARM='TESTWTR1'

* THIS WILL START EXTERNAL WRITER TESTWTR1. THIS EXTERNAL *
* WRITER WILL PROCESS SYSDOUT OUTPUT DIRECTED TO IT ON ANY *
* OUTPUT QUEUE AND ANY OUTPUT CLASS.

* //STEP1 EXEC PGM=SAPISSI,PARM='TESTWTR1,ADF259'

* THIS WILL START EXTERNAL WRITER TESTWTR1. THIS EXTERNAL
* WRITER WILL PROCESS SYSOUT OUTPUT DIRECTED TO IT ON ANY
  * OUTPUT QUEUE AND ONLY OUTPUT CLASSES "ADF.M259".
  *
  * IF YOU WANT TO SEND OUTPUT TO THIS EXTERNAL WRITER, RUN A JOB
  * THAT DIRECTS OUTPUT TO A SYSOUT DATASET:
  *
  * //SYSPRINT DD SYSOUT=(F,TESTWR1)
  *
  * THIS EXTERNAL WRITER WILL PROCESS THIS SYSOUT DATASET WHEN THE
  * CREATING PROGRAM TERMINATES.
  *
  * TO DYNAMICALLY MODIFY THE LIST OF CLASSES THE EXTERNAL WRITER
  * IS ELIGIBLE TO PROCESS, ENTER THE MODIFY OPERATOR COMMAND:
  *
  * F sapissi,CLASSLIST=classlst
  *
  * WHERE 'sapissi' IS THE JOBNAME OF THE EXTERNAL WRITER JOB AND
  * 'classlst' IS THE NEW LIST OF VALID CLASSES.
  *
  * TO REACTIVATE ALL 36 OUTPUT CLASSES, THE FOLLOWING MODIFY
  * OPERATOR COMMAND CAN BE USED:
  *
  * F sapissi,CLASSLIST=*
  *
  * TO STOP THIS EXTERNAL WRITER SIMPLY ENTER THE STOP OPERATOR
  * COMMAND:
  *
  * P sapissi
  *
  * WHERE 'sapissi' IS THE JOBNAME OF THE EXTERNAL WRITER JOB.

******************************************************************************

STM R14,R12,12(R13)   SAVE INCOMING REGISTER VALUES
LR  R12,R15           COPY MODULE ADDRESS
LA  R11,4095,(R12)    SET SECOND BASE...
LA  R11,(,R11)        REGISTER ADDRESS
USING SAPISSI,R12,R11  SET MODULE ADDRESSABILITY
LR  R10,R1            SAVE PARM ADDRESS
LR  R3,R13            COPY OLD SAVEAREA ADDRESS
L   R9,=A(WORKLEN)    GET STORAGE LENGTH
STORAGE OBTAIN,LENGTH=(R9) GET DYNAMIC STORAGE
LR  R13,R1            SAVE STORAGE ADDRESS
LR  R0,R1             COPY IT
LR  R14,R1            AGAIN
L   R1,=A(WORKLEN)    GET STORAGE LENGTH
XR  R15,R15           SET FILL BYTE
MVCL R0,R14           CLEAR THE DYNAMIC STORAGE
ST  R3,4,(R13)        SAVE OLD SAVEAREA ADDRESS
USING WORKAREA,R13    SET WORKAREA ADDRESSABILITY

******************************************************************************

* VALID PARM FORMATS ARE:
* 'WRTNM'
* 'WRTNM,CLASSLIST'
* WHERE 'WRTNM' IS A MAXIMUM OF EIGHT CHARACTERS
* 'CLASSLIST' IS A MAXIMUM OF 36 CHARACTERS
* IF 'CLASSLIST' IS SPECIFIED, THE 'WRTNM' AND 'CLASSLIST'
* PARAMETERS MUST BE COMMA SEPARATED.

***********************************************
OPEN (SYSPRINT,OUTPUT),MODE=31
LR R1,R10 GET PARM ADDRESS
L R9,0(R1) GET ADDRESS OF PARM
CLC 0(R9),=H'45' MAX LENGTH OK?
BH RETURN08 NO - ISSUE MESSAGE AND END
CLC 0(R9),=H'0' A PARM?
BNH RETURN08 NO - ISSUE MESSAGE AND END
B WTRNMOK WRITER NAME SHOULD BE OK
WTRNMOK EQU *
XR R14,R14 CLEAR R14
ICM R14,'0011',0(R9) COPY THE LENGTH
XR R1,R1 CLEAR LENGTH COUNTER
MVC WTRNM(R8),=8C' ' CLEAR THE AREA
LA R2,WTRNM GET TARGET ADDRESS
LA R9,2(R9) SKIP PAST LENGTH
WTRNMLP EQU *
LTR R14,R14 PARM DONE?
BZ WTRNMCNK YES - CHECK LENGTH
C R1,=F'8' MAX LENGTH?
BH RETURN08 YES - ISSUE MESSAGE AND END
CLI 0(R9),C', END OF PARM?
BNE WTRNMMV NO - COPY THE DATA
BCTR R14,0 REDUCE LENGTH BY ONE
LA R9,1(R9) POINT TO NEXT DATA BYTE
LTR R14,R14 END OF DATA?
BZ RETURN08 YES - ISSUE MESSAGE AND END
B WTRNMCNK CHECK LENGTH
WTRNMMV EQU *
ST R14,R14SAVE SAVE R14
BAL R14,CHARCHK CHECK FOR ALPHANUMERIC
LTR R15,R15 ALPHANUMERIC?
BNZ RETURN08 NO - ALL DONE
L R14,R14SAVE LOAD R14
MVC 0(1,R2),0(R9) MOVE DATA
BCTR R14,0 REDUCE LENGTH BY ONE
LA R9,1(R9) POINT TO NEXT DATA BYTE
LA R2,1(R2) POINT TO NEXT TARGET BYTE
LA R1,1(R1) ADD ONE TO LENGTH
B WTRNMLP CHECK NEXT BYTE
WTRNMCHK EQU *
LTR R1,R1 LENGTH OK?
BZ RETURN08 NO - ISSUE MESSAGE AND END
MVC CLASLIST(36),=36C' ' CLEAR THE AREA
LA R2,CLASLIST GET CLASS LIST AREA ADDRESS
XR R1,R1 CLEAR LENGTH COUNTER

CLLSTLP EQU *
LTR R14,R14 PARM DONE?
BZ CLLSTDON YES - WE'RE DONE
C R1,=F'36' MAX LENGTH?
BNL RETURN08 YES - ISSUE MESSAGE AND END
ST R14,R14SAVE SAVE R14
BAL R14,CHARCHK CHECK FOR ALPHANUMERIC
LTR R15,R15 ALPHANUMERIC?
BNZ RETURN08 NO - ALL DONE
L R14,R14SAVE LOAD R14
MVC @1(R1),@0(R9) MOVE DATA
BCTR R14,0 REDUCE LENGTH BY ONE
LA R9,1(R9) POINT TO NEXT DATA BYTE
LA R2,1(R2) POINT TO NEXT TARGET BYTE
LA R1,1(R1) ADD ONE TO LENGTH
B CLLSTLP CHECK NEXT BYTE

CLLSTDON EQU *
**************************************************************************
* DETERMINE THE ACTIVE SUBSYSTEM NAME.                                *
**************************************************************************
L R15,16 GET CVT ADDRESS
L R15,0(R15) GET TCB/ASCB AREA ADDRESS
L R15,4(R15) GET CURRENT TCB
L R15,TCBJSCB-TCB(R15) GET JSCB ADDRESS?
L R15,JSCBSSIB-IEZJSCB(R15) GET SSIB ADDRESS?
MVC SSNMSAVE(8),=8C' ' INITIALIZE THE AREA
MVC SSNMSAVE(4),SSIBSSNM-SSIB(R15) MV SSNAME
**************************************************************************
GETCIB LA R5,COMADDRC ADDR OF RESPONSE AREA FOR QEDIT
MVC EXTWRK(EXTLN),EXTLST MOVE IN THE MODEL
EXTRACT (5),FIELDS=COMM,MF=(E,EXTWRK) GET ADDR OF COMM AREA
L R5,COMADDRC LOAD ADDR OF COMM AREA
USING COMLIST,R5
ST R5,COMADDRC SAVE COMM AREA ADDRESS
L R6,COMCIBPT GET ADDRESS OF CIB
USING CIBNEXT,R6
C R6,=F'0' CIB EXIST ?
BE SETCOUNT NO - GO SET COUNT
QEDIT ORIGIN=COMCIBPT,BLOCK=(R6) YES - FREE IT
LTR R15,R15 GO O.K.? 
BZ SETCOUNT YES - GO SET COUNT
ABEND @0 ERROR

SETCOUNT EQU *
****SET LIMIT ON MODIFY COMMANDS****

**QEDIT ORIGIN=COMCIBP,T,CIBCTR=1 ONE OPERATOR CMD AT A TIME**

WTO 'XWR0801 - COMMAND INTERFACE ACTIVATED.', XRCODE=(1),DESC=(6)

LA R7, TIOTADDR GET ADDRESS OF TARGET AREA
EXTRACT (R7), FIELDS=TIOT GET THE TIOT ADDRESS
L R7, TIOTADDR COPY THE TIOT ADDRESS
LA R7, 24(R7) POINT TO TIOT ENTRY AREA
USING TIOENTRY,R7 SET ADDRESSABILITY

DDLOOP1 EQU *

CLI TIOLERGH,X '00' END OF LIST?
BE OPENOUT YES - NOT GOOD - LET OPEN FAIL
CLC TIOEDDMN(B), = C 'OUTPUT' THE 'OUTPUT' DD?
BE DDMINCH YES - CHECK DATASET TYPE
XR R1, R1 CLEAR R1
IC R1, TIOLERGH GET TIOT ENTRY LENGTH
LA R7, 0(R1, R7) POINT TO NEXT ENTRY
B DDLOOP1 GO CHECK NEXT ENTRY

DDMINCH EQU *

LA R8, EPA GET ADDRESS OF THE EPA
ST R8, SWEPAPTR INITIALIZE EPA POINTER
USING ZB505, R8 ADDRESSABILITY TO EPA
XC EPA, EPA CLEAR THE WORK AREA
MVC SWVA, TIOEJFCB MV SVA OF JFCB INTO EPA
SWAREQ FCODE=RL,
EPA=SWEPAPTR,
UNAUTH=YES,
MF=(E, SWAPARMS)

LTR R15, R15 LOCATE WAS SUCCESSFUL?
BNZ RETURN20 NO - TERMINATE
L R8, SWBLKPTR SET POINTER TO THE JFCB
DROP R8
USING INFJFCB, R8 SET JFCB ADDRESSABILITY
TM JFCBTSMD, JFCSDS A SYSDATASET?
BNO OPENOUT NO - OPEN THE OUTPUT DATASET
MVC OUTPUT+82(2), = 'H 32760' MOVE IN THE LRECL
DROP R7, R8

OPENOUT EQU *

OPEN (OUTPUT, OUTPUT), MODE=31

******************************************************************************

MVI RECBUFF, C ' ' SET FILL CHARACTER
MVC RECBUFF+I(132), RECBUFF CLEAR OUT RECORD AREA
MVC RECBUFF(L'MSG98), MSG98 MOVE IN THE MESSAGE
MVC RECBUFF+27(8), WTRNM MOVE IN THE WRITER NAME
PUT SYSPRINT, RECBUFF WRITE THE MESSAGE

******************************************************************************

SSICALL EQU *

```assembly
LA R3, SSOBAREA       GET SSOB ADDRESS
USING SSOBEGIN,R3    SET ADDRESSABILITY
LA R4, SSOBGN        GET SSS2 ADDRESS
USING SSS2,R4        SET ADDRESSABILITY
XC SSOB(SSOBHSIZE),SSOB CLEAR THE SSOB
MVC SSOBID(4),=C'SSOB' SET SSOB ID
MVC SSOBFUNC(2),=AL2(SSOBSSOU2) SET FUNCTION ID
MVC SSOBLEN(2),=AL2(SSOBHSIZE) SET SSOB HEADER SIZE
ST R4, SSOBINDV      SET SSS2 ADDRESS

LA R14, SSS2         GET AREA ADDRESS
L R1, =A(SSS2SIZE)   GET LENGTH
LR R0, R14           COPY ADDRESS
XR R15, R15          SET FILL BYTE
MVCL R0, R14         CLEAR THE AREA
MVC SSS2EYE(4),=C'SSS2' SET EYECATCHER
MVC SSS2LEN, =AL2(SSS2SIZE) SET SSS2 SIZE
MVI SSS2VER, SSS2VER SET THE VERSION

* DETERMINE THE JOB SELECTION CRITERIA THAT WILL BE USED BY THE  *
* EXTERNAL WRITER. THE IAZSSS2 MACRO PROVIDES INSIGHT INTO WHAT  *
* CRITERIA MAY BE USED.                                       *
*                                                            *
* THIS PARTICULAR EXTERNAL WRITER SELECTS SYSOUT INDEPENDENTLY OF *
* CLASS DISPOSITION (SSS2SAWT), IT SELECTS BY CLASS LIST (SSS2SCLS *
* AND SSS2SCLSL) IF A CLASS LIST HAS BEEN SPECIFIED ON THE PROGRAM *
* PARM, AND IT SELECTS BY WRITER NAME (SSS2SPGM AND SSS2PGMN).    *
*                                                            *
* AN ECB ADDRESS HAS BEEN INCLUDED IN SSS2ECBP. THIS ALLOWS THE *
* EXTERNAL WRITER TO WAIT ON AN ECB THAT WILL GET POSTED WHEN NEW *
* SYSOUT OUTPUT BECOMES AVAILABLE ON SPOOL MATCHING THE SELECTION *
* CRITERIA SPECIFIED.                                         *

MVI SSS2SEL1, SSS2SAWT SELECT ANY DISP
O1 SSS2SEL2, SSS2SPGM SELECT BY WRITER NAME
MVC SSS2PGMN(8), WTRNM SET THE WRITER NAME
O1 SSS2TYPE, SSS2PUGE SET TYPE TO PUT/GET

SSCALL2 EQU *
O1 SSS2DPS1, SSS2RNPR DON'T RETURN THIS D/S AGAIN
NI SSS2SEL1, 255-SSS2SCLS TURN CLASS SELECTION FLAG OFF
CLC CLASSLST(36), =36C' ' ANY CLASS SELECTION CRITERIA?
BE NOCLASS NO - DON'T SET CLASS FLAGS
O1 SSS2SEL1, SSS2SCLS SET CLASS SELECTION FLAG
MVC SSS2SCLSL(36), CLASSLST MOVE IN CLASS LIST

NOCLASS EQU *
XC SELECB(4), SELECB CLEAR ECB
LA R1, SELECB GET ECB ADDRESS
ST R1, SSS2ECBP SAVE ECB ADDRESS
```
MODESET MODE=SUP
LA R1,SSOBAREA
O R1,'X'00000000'
ST R1,SSOBPTR
LA R1,SSOBPTR
IEFSSREQ , MAKE SUBSYSTEM REQUEST
LR R9,R15
MODESET MODE=PROB
LTR R9,R9
SUBSYSTEM REQUEST OK?
BNZ SSREQAB NO - LET'S END
XR R8,R8
CLEAR R8
ICM R8,'B'001',SSOBRET+3
GET RETURN CODE
C R8,'=F'44'
SSOBRET TOO HIGH?
BH ERRUNKWN YES - UNKNOWN
B BRTBL1(R8)
PROCESS ACCORDINGLY

BRTBL1 EQU *
B REQOK SSOBRETN=00
B REQEODS SSOBRETN=04
B REQINV A SSOBRETN=08
B REQUA N SSOBRETN=12
B REQUAD P SSOBRETN=16
B REQIDT SSOBRETN=20
B REQU D SSOBRETN=24
B REQTKNNM SSOBRETN=28
B REQLRR SSOBRETN=32
B REQILSS SSOBRETN=36
B REQBDIS SSOBRETN=40
B REQCLON SSOBRETN=44

ERRUNKWN EQU *
MVC DBL2(4),SSOBRETN COPY RETURN CODE
UNPK DBL1(9),DBL2(5) UNPACK IT
NC DBL1(8),='8X'0F' TURN OFF HIGH NIBBLE
TR DBL1(8),='C'0123456789abcdef' MAKE IT READABLE
MVI RECBUFF,' ' SET FILL CHARACTER
MVC RECBUFF+1(132),RECBUFF CLEAR OUT RECORD AREA
MVC RECBUFF(MSG1B:MSG18),MSG18 MOVE IN THE MESSAGE
MVC RECBUFF+33(8),DBL1 MOVE IN THE SSOBRETN
PUT SYSPRINT,RECBUFF WRITE THE MESSAGE
B RETURN12 RETURN

SSREQAB EQU *
LA R1,MSG17 GET MESSAGE ADDRESS
LA R15,L'MSG17 GET MESSAGE LENGTH
BAL R14,MSGPUT GO WRITE THE MESSAGE
B RETURN12 RETURN

REQEODS EQU *
L R8,SSS2ECBP GET SAPI ECB ADDRESS?
ST R8,ECBLIST+0 SAVE IN THE ECB LIST
L R8,COMEBCBP GET ADDR OF COMMUNICATION ECB
ST R8,ECBLIST+4          SAVE IN THE ECB LIST
OI ECBLIST+4,X'80'        SET LAST ECB INDICATOR
LA R8,ECBLIST            GET ECB LIST ADDRESS
WAIT 1,ECBLIST=(R8)       WAIT FOR AN EVENT TO COMPLETE
L R8,ECBLIST+0           GET FIRST ECB ADDRESS
TM Ø(R8),X'40'           A SAPI EVENT?
BNO CHKECB2              NO - CHECK OTHER ECB
XC SELECB(4),SELECB      CLEAR ECB FOR NEXT TIME
B SSICALL2               MAKE SSI CALL AGAIN

CHKECB2 EQU *
L R8,ECBLIST+4          GET SECOND ECB ADDRESS
TM Ø(R8),X'40'           A CONSOLE EVENT?
BNO REQEODS              NO - LET'S GO WAIT
L R5,COMADDR            LOAD ADDR OF COMMUNICATION A
L R6,COMCIBPT           GET ADDRESS OF CIB
LTR R6,R6               VALID POINTER?
B2 REQEODS               NO - RETURN
CLI CIBVERB,CIBMODFY     IS IT A MODIFY COMMAND ?
BE CMODIFY               YES - GO PROCESS
CLI CIBVERB,CIBSTOP      IS IT A STOP COMMAND ?
BNE NOSTOP               NO - JUST CONTINUE
B CSTOP                  YES - GO SHUT THINGS DOWN

CMODIFY EQU *
XR R14,R14               CLEAR R15
ICM R14,B'0011',CIBDATLN  GET COMMAND LENGTH
C R14,=F'11'            MINIMUM LENGTH OK?
BL CMDERRØ1             NO - ISSUE AN ERROR
CLC CIBDATA(1),=C'CLASSLIST=' CLASSLIST= OPERAND?
BNE CMDERRØ1             NO - ISSUE AN ERROR
S R14,=F'10'          REDUCE BUFFER LENGTH
C R14,=F'36'            TOO MUCH DATA?
BH CMDERRØ2             YES - ISSUE AN ERROR
MVC CLASSBUF(36),=36C' ' INITIALIZE BUFFER AREA
CLI CIBDATA+10,C'*'    TURN ALL CLASSES BACK ON?
BE MDLSTDON             YES - THEY'LL BE ACTIVE AGAIN
BCTR R14,Ø              REDUCE LENGTH BY ONE FOR EX
EX R14,CLASSMVC         COPY THE CLASS LIST
LA R9,CLASSBUF         GET ADDRESS OF CLASS LIST
LA R14,1(R14)        ADD ONE BACK TO LENGTH

MDLSTLP EQU *
LTR R14,R14             CLASS LIST DONE?
BZ MDLSTDON            YES - GO ON
ST R14,R14SAVE         SAVE R14
BAL R14,CHARCHK        CHECK FOR ALPHANUMERIC
LTR R15,R15           ALPHANUMERIC?
BNZ CMDERRØ2          NO - ISSUE AN ERROR
L R14,R14SAVE        LOAD R14
BCTR R14,Ø              REDUCE LENGTH BY ONE
LA R9,1(R9)              POINT TO NEXT DATA BYTE
B MDLSTLP        CHECK NEXT BYTE

MDLSTDON EQU *
MVC CLASSLST(36),CLASSBUF REPLACE THE ACTIVE CLASS LIST
QEDIT ORIGIN=COMCIBPT,BLOCK=(R6) YES - FREE IT
WTO 'XWR081I - CLASSLIST UPDATE SUCCESSFUL.', X ROUTCDE=(1),DESC=(6)
B REQEOQS        JUST GO WAIT

CMDERR01 EQU *
WTO 'XWR089I - UNRECOGNIZED MODIFY COMMAND.', X ROUTCDE=(1),DESC=(6)
B NOSTOP        GO RELEASE CIB

CMDERR02 EQU *
WTO 'XWR088I - INVALID CLASSLIST FORMAT. CLASSLIST MUST NOT EXCEED 36 ALPHANUMERIC CHARACTERS.', X ROUTCDE=(1),DESC=(6)
B NOSTOP        GO RELEASE CIB

NOSTOP EQU *
QEDIT ORIGIN=COMCIBPT,BLOCK=(R6) YES - FREE IT
B REQEOQS        JUST GO WAIT

CSTOP EQU *
QEDIT ORIGIN=COMCIBPT,BLOCK=(R6) YES - FREE IT
B SAPIDONE       WE'RE DONE

REQINVA EQU *
LA R1,MSG3       GET MESSAGE ADDRESS
LA R15,L'MSG3    GET MESSAGE LENGTH
BAL R14,MSGPUT   GO WRITE THE MESSAGE
B EXIT

REQUNAV EQU *
LA R1,MSG4       GET MESSAGE ADDRESS
LA R15,L'MSG4    GET MESSAGE LENGTH
BAL R14,MSGPUT   GO WRITE THE MESSAGE
B EXIT

REQDUPJ EQU *
LA R1,MSG5       GET MESSAGE ADDRESS
LA R15,L'MSG5    GET MESSAGE LENGTH
BAL R14,MSGPUT   GO WRITE THE MESSAGE
B EXIT

REQIDST EQU *
LA R1,MSG6       GET MESSAGE ADDRESS
LA R15,L'MSG6    GET MESSAGE LENGTH
BAL R14,MSGPUT   GO WRITE THE MESSAGE
B EXIT

REQAUTH EQU *
LA R1,MSG7       GET MESSAGE ADDRESS
LA R15,L'MSG7    GET MESSAGE LENGTH
BAL R14,MSGPUT   GO WRITE THE MESSAGE
B EXIT

REQTKNM EQU *
LA R1,MSG8       GET MESSAGE ADDRESS
LA  R15,L'MSG8  GET MESSAGE LENGTH
BAL  R14,MSGPUT  GO WRITE THE MESSAGE
B  EXIT
REQLERR EQU *
LA  R1,MSG9  GET MESSAGE ADDRESS
LA  R15,L'MSG9  GET MESSAGE LENGTH
BAL  R14,MSGPUT  GO WRITE THE MESSAGE
B  EXIT
REQCLS EQU *
LA  R1,MSG10  GET MESSAGE ADDRESS
LA  R15,L'MSG10  GET MESSAGE LENGTH
BAL  R14,MSGPUT  GO WRITE THE MESSAGE
B  EXIT
REQDIS EQU *
LA  R1,MSG11  GET MESSAGE ADDRESS
LA  R15,L'MSG11  GET MESSAGE LENGTH
BAL  R14,MSGPUT  GO WRITE THE MESSAGE
B  EXIT
REQK EQU *
******************************************************************************
LA  R1,SSS2BTK  GET BLOCK TOKEN ADDRESS
LA  R0,DYNPRMS2  GET TARGET AREA ADDRESS
L  R1,=A(S992LN)  GET THE LENGTH
LA  R14,S992  GET SOURCE AREA ADDRESS
LR  R15,R1  GET THE LENGTH
MVCL  R0,R14  MOVE IN THE MODEL
LA  R1,DYNPRMS2  GET PARM AREA ADDRESS
LA  R2,S992RB-S992(,R1)  GET RELOCATED S992RB ADDRESS
O  R2,='80000000'  SET FLAG
ST  R2,0(,R1)  SAVE RELOCATED ADDRESS IN PARMS
LA  R2,S992TUPL-S992(,R1)  GET RELOCATED S992TUPL ADDRESS
STCM  R2,B'1111',S992TXTP-S992(R1)  SV RELOCATED S992TXTP ADDRESS
LA  R2,TU2001-S992(,R1)  GET RELOCATED TU2001 ADDRESS
STCM  R2,B'1111',S992TUPL-S992(R1)  SV RELOCATED TU2001 ADDRESS
LA  R2,TU2002-S992(,R1)  GET RELOCATED TU2002 ADDRESS
STCM  R2,B'1111',TU202-S992(R1)  SV RELOCATED TU2002 ADDRESS
LA  R2,TU2003-S992(,R1)  GET RELOCATED TU2003 ADDRESS
MVC  6(4,R2),SSNMSAVE  MOVE IN THE SUBSYSTEM NAME
STCM  R2,B'1111',TU203-S992(R1)  SV RELOCATED TU2003 ADDRESS
MVC  TU204-S992(4,R1),SSS2BTOK GET DALBRTKN ADDRESS
0I  TU204-S992(R1),X'80'  SET LAST TU FLAG
MVC  DSN2-S992(44,R1),SSS2DSN MOVE IN THE DNAME
SVC  99  ALLOCATE THE DATASET
LTR  R15,R15  ALLOCATE OK?
BZ ALLOC20K  YES - KEEP GOING
LA R1,DYNPRMS2  GET PARM AREA ADDRESS
MVC DBL2(4),S992ERR-S992(R1) COPY RETURN CODE
UNPK DBL1(9),DBL2(5) UNPACK IT
NC DBL1(8),=8X'0F' TURN OFF HIGH NIBBLE
TR DBL1(8),=C'0123456789ABCDEF' MAKE IT READABLE
MVC ALLOCMG+17(8),DBL1 MOVE IN RETURN CODE
MVC ALLOCMG+41(44),SS2DSN COPY THE DSNAME
PUT SYSPRINT,ALLOCMG WRITE THE MESSAGE
ABEND  ABEND

ALLOCMG  EQU  *
LA R1,DYNPRMS2  GET PARM AREA ADDRESS
MVC TEMPDDN(8),DDN2-S992(R1) SAVE THE DDNAME
MVC INPUT+40(8),TEMPDDN MOVE IN THE DDNAME
CLC SSS2MLRL(2),='0' MAX LOGICAL RECORD LENGTH=0?
BNH INDONE NO - JUST SKIP THIS ONE
MVC INPUT+82(2),SSS2MLRL MOVE IN THE LRECL
OPEN (INPUT,INPUT),MODE=31

******************************************************************************
* BUILD AND WRITE OUT A SEPARATOR RECORD SO THAT THE OUTPUT FROM          *
* DIFFERENT DD'S CAN BE IDENTIFIED IN THE EXTERNAL WRITER OUTPUT          *
* FILE.  THE SEPARATOR RECORD (HDRREC1) USED BY THIS PROGRAM              *
* USES JOBNAME, JOB NUMBER, PROC NAME, STEP NAME, DD NAME,               *
* OUTPUT CLASS, AND DS NAME INFORMATION.                                  *
*                                                                       *
* THE SAPI SSI CALL RETURNS MUCH USEFUL INFORMATION IN THE SSS2.          *
* OTHER POTENTIALLY USEFUL FIELDS THAT COULD BE USED FOR                 *
* SEPARATOR INFORMATION MIGHT INCLUDE THE SYSTEM THE JOB RAN               *
* ON (SSS2SYS), THE JES MEMBER THE JOB RAN ON (SSS2MBR), AND THE          *
* TIME OF DAY THE OUTPUT DATASET WAS MADE AVAILABLE TO SPOOL              *
* (SSS2TOD).                                                              *
******************************************************************************

LA  R0,RECBUFF  GET BUFFER ADDRESS
LA  R14,RECBUFF  GET BUFFER ADDRESS
L  R1,='32760'  SET LENGTH
XR  R15,R15  SET FILL BYTE
MVCL  R0,R14  CLEAR THE BUFFER
MVC RECBUFF(133),HDRREC1 MOVE IN THE SEPARATOR RECORD
MVC RECBUFF+JBNMOFF(8),SSS2JOB Employees Name
MVC RECBUFF+JB#OFF(8),SSS2JOB# Employees Name
MVC RECBUFF+PRCOFF(8),SSS2PROCD Move in the PROCNAME
MVC RECBUFF+STPOFF(8),SSS2STPD Move in the STEPNAME
MVC RECBUFF+DDNDOFF(8),SSS2DDND Move in the DDNAME
MVC RECBUFF+CLSOff(1),SSS2CLAR Move in the CLASS
MVC RECBUFF+DSNMOFF(44),SSS2DSN Move in the DSNAME
XC  LRECL(4),LRECL  CLEAR LRECL AREA
MVC  LRECL(2),='137'  SET LENGTH
TM  OUTPUT+36,='80'  FIXED LENGTH RECORDS?
BO  FIXED1  YES - SET TO FIXED OUTPUT
TM OUTPUT+36,X'40' VARIABLE LENGTH RECORDS?
BO VARIABL1 YES - SET TO VARIABLE OUTPUT
B RETURN16 SOMETHING'S NOT RIGHT
FIXED1 EQU *
LA R8,RECBUFF SET BUFFER ADDRESS
B OUTPUT1 GO WRITE THE OUTPUT
VARIABL1 EQU *
LA R8,LRECL SET BUFFER ADDRESS
OUTPUT1 EQU *
PUT OUTPUT,(R8)

********************************************************************
* THIS SAMPLE PROGRAM HAS TAKEN A VERY SIMPLE APPROACH TO HANDLING *
* THE SYSOUT OUTPUT DIRECTED TO ITS EXTERNAL WRITER. IT USES A *
* SEQUENTIAL OUTPUT DATASET (EITHER F(B) OR V(B)) THAT CAN BE *
* EITHER A REAL DATASET OR A SYSOUT DATASET ON SPOOL. IF YOU USE *
* A REAL DATASET, DEFINE THE DATASET WITH AN LRECL AT LEAST AS *
* LARGE AS THE MAXIMUM RECORD SIZE THAT WILL BE PROCESSED BY THE *
* EXTERNAL WRITER. IF YOU DIRECT THE OUTPUT DD TO SYSOUT, THE *
* SYSOUT DATASET WILL GET CREATED WITH AN LRECL OF 32760. *
* *
* THERE ARE MANY POSSIBILITIES AVAILABLE. AT THIS POINT IN THE *
* PROGRAM, THE INPUT DD IS OPENED TO A SYSOUT SPOOL DATASET THAT *
* HAS BEEN CREATED WITH OUTPUT DIRECTED TO THIS EXTERNAL WRITER. *
* HOW YOU CHOOSE TO PROCESS THIS INPUT DATASET IS LEFT TO YOUR *
* IMAGINATION.
********************************************************************
GETLP1 EQU *
LA R0,RECBUFF GET BUFFER ADDRESS
LA R14,RECBUFF GET BUFFER ADDRESS
L R1,=F'32760' SET LENGTH
XR R15,R15 SET FILL BYTE
MVCL R0,R14 CLEAR THE BUFFER
GET INPUT,RECBUFF READ INPUT RECORD
XCl LRECL(4),LRECL CLEAR LRECL AREA
XR R1,R1 CLEAR R1
ICM R1,B'0011',SSS2MLRL GET RECORD LENGTH
LA R1,4,(R1) ADD 4 FOR RDW
STCM R1,B'0011',LRECL SAVE THE LENGTH
CLC LRECL(2),=H'32756' LENGTH OK?
BNH LRECLK YES - DON'T ADJUST
MVC LRECL(2),=H'32760' SET TO MAX
LRECLK EQU *
TM OUTPUT+36,X'80' FIXED LENGTH RECORDS?
BO FIXED2 YES - SET TO FIXED OUTPUT
TM OUTPUT+36,X'40' VARIABLE LENGTH RECORDS?
BO VARIABL2 YES - SET TO VARIABLE OUTPUT
B RETURN16 SOMETHING'S NOT RIGHT
FIXED2 EQU *
LA R8,RECBUFF SET BUFFER ADDRESS
B  OUTPUT2   GO WRITE THE OUTPUT

VARIABL2 EQU  *

LA  R8,LRECL   SET BUFFER ADDRESS

OUTPUT2 EQU  *

PUT OUTPUT,(R8)
B  GETLP1   GET NEXT INPUT RECORD

INDONE EQU  *

CLOSE (INPUT),MODE=31

*******************************************************************************

LA  R0,DYNPRMS4   GET TARGET AREA ADDRESS
L  R1,=A(S994LN)   GET THE LENGTH
LA  R14,S994   GET SOURCE AREA ADDRESS
LR  R15,R1   GET THE LENGTH
MVCL R0,R14   MOVE IN THE MODEL
LA  R1,DYNPRMS4   GET PARM AREA ADDRESS
LA  R2,S994RB-S994(,R1)   GET RELOCATED S994RB ADDRESS
O  R2,='00000000'   SET FLAG
ST  R2,0(,R1)   SAVE RELOCATED ADDRESS IN PARMS
LA  R2,S994TUPL-S994(,R1)   GET RELOCATED S994TUPL ADDRESS
STCM R2,B'1111',S994TXTP-S994(1)   SW RELOCATED S994TUPL ADDR
LA  R2,TU4001-S994(,R1)   GET RELOCATED TU4001 ADDRESS
O  R2,='80000000'   SET LAST TF FLAG
STCM R2,B'1111',S994TUPL-S994(1)   SW RELOCATED TU4001 ADDR
MVC  DDN4-S994(8,R1),TEMPDDN   COPY THE DDNAME
SVC  99   DEALLOCATE THE DATASET
LTR  R15,R15   DEALLOCATE OK?
BZ  DALLOC20K   YES - KEEP GOING
LA  R1,DYNPRMS4   GET PARM AREA ADDRESS
MVC  DBL2(4),S994ERR-S994(1)   COPY RETURN CODE
UNPK  DBL1(9),DBL2(5)   UNPACK IT
NC  DBL1(8),='0F'   TURN OFF HIGH NIBBLE
TR  DBL1(8),='0F'   TURN OFF HIGH NIBBLE
MVC  DALOCMSG+17(8),DBL1   MOVE IN RETURN CODE
MVC  DALOCMSG+43(44),SSS2DSN   COPY THE DSNAME
PUT  SYSPRINT,DALOCMSG   WRITE THE MESSAGE
ABEND  11   ABEND

DALLOC20K EQU  *

B  SSICALL2   GO TRY AGAIN

SAPIDONE EQU  *

LA  R14,SSS2INPT   GET AREA ADDRESS
L  R1,=A(SSS2DISP-SSS2INPT)   GET LENGTH
LR  R0,R14   COPY ADDRESS
XR  R15,R15   SET FILL BYTE
MVCL  R0,R14   CLEAR THE AREA
LA  R14,SSS2DISP   GET AREA ADDRESS
L  R1,=A(SSS2OUTP-SSS2DISP)   GET LENGTH
LR  R0,R14   COPY ADDRESS
XR  R15,R15   SET FILL BYTE
MVCL  R0,R14   CLEAR THE AREA
OI  SSS2MSC1,SSS2CTRL   SET TERMINATION FLAG

MODESET MODE=SUP
LA R1,SSOBPTR           POINT TO SSOB PTR
IEFSSREQ               MAKE SUBSYSTEM REQUEST
LR R9,R15              SAVE THE RETURN CODE
MODESET MODE=PROB
LTR R9,R9              IEFSSREQ WAS OK?
BZ RETNCHK             YES - CHECK SSOBRETN
ST R9,DBL2             COPY THE RETURN CODE
UNPK DBL1(9),DBL2(5)    UNPACK IT
NC DBL1(8),=8'0F'       TURN OFF HIGH NIBBLE
TR DBL1(8),=C'0123456789ABCDEF' MAKE IT READABLE
MVC TERMMGS1+38(8),DBL1 MOVE IN THE RETURN CODE
PUT SYSPRINT,TERMMGS1  ISSUE MSG
B EXIT                 WE'RE DONE

RETNCHK EQU *
CLC SSOBRETN(4),=F'0'  IEFSSREQ WENT AS EXPECTED?
BE EXIT                YES - WE'RE DONE
MVC DBL2(4),SSOBRETN   SAVE RETURN CODE
UNPK DBL1(9),DBL2(5)    UNPACK IT
NC DBL1(8),=8'0F'       TURN OFF HIGH NIBBLE
TR DBL1(8),=C'0123456789ABCDEF' MAKE IT READABLE
CLI SSOBRETN+3,X'00'
BE EXIT                MOVE IN THE RETURN CODE
MVC TERMMGS2+44(8),DBL1 MOVE IN THE RETURN CODE
PUT SYSPRINT,TERMMGS1  ISSUE MSG
B EXIT                 WE'RE DONE

******************************************************************************
EXIT EQU *
******************************************************************************
MVI RECBUFF,C' '      SET FILL CHARACTER
MVC RECBUFF+1(132),RECBUFF CLEAR OUT RECORD AREA
MVC RECBUFF(L'MSG99),MSG99 MOVE IN THE MESSAGE
MVC RECBUFF+27(8),WTRNM MOVE IN THE WRITER NAME
PUT SYSPRINT,RECBUFF   WRITE THE MESSAGE
******************************************************************************
CLOSE (OUTPUT),MODE=31
CLOSE (SYSPRINT),MODE=31
LR R1,R13           COPY WORKING STORAGE ADDRESS
L R10,4,(R13)       SAVE OLD SAVEAREA ADDRESS
L R9,=A(WORKLEN)    GET STORAGE LENGTH
STORAGE RELEASE,LENGTH=(R9),ADDR=(R1) RELEASE IT
LR R13,R10          RESTORE SAVEAREA ADDRESS
LM R14,R12,12(R13)  RESTORE REGISTERS
XR R15,R15          SET RETURN CODE
BR R14              RETURN
******************************************************************************
RETURNØ8 EQU *
PUT SYSPRINT,PARMSGØ  ISSUE BAD PARM MSG
PUT SYSPRINT,PARMSG1  ISSUE BAD PARM MSG
PUT SYSPRINT,PARMSG2  ISSUE BAD PARM MSG
CLOSE (SYSPRINT), MODE=31
LR R1, R13            COPY WORKING STORAGE ADDRESS
L R10, 4(R13)         SAVE OLD SAVEAREA ADDRESS
L R9, =A(WORKLEN)     GET STORAGE LENGTH
STORAGE RELEASE, LENGTH=(R9), ADDR=(R1) RELEASE IT
LR R13, R10           RESTORE SAVEAREA ADDRESS
LM R14, R12, 12(R13)  RESTORE REGISTERS
LA R15, 8             SET RETURN CODE
BR R14                RETURN
*****************************************************************************
RETURN12 EQU *
CLOSE (SYSPRINT), MODE=31
CLOSE (OUTPUT), MODE=31
LR R1, R13            COPY WORKING STORAGE ADDRESS
L R10, 4(R13)         SAVE OLD SAVEAREA ADDRESS
L R9, =A(WORKLEN)     GET STORAGE LENGTH
STORAGE RELEASE, LENGTH=(R9), ADDR=(R1) RELEASE IT
LR R13, R10           RESTORE SAVEAREA ADDRESS
LM R14, R12, 12(R13)  RESTORE REGISTERS
LA R15, 12            SET RETURN CODE
BR R14                RETURN
*****************************************************************************
RETURN16 EQU *
PUT SYSPRINT, RECFMSG1 ISSUE RECORD FORMAT MESSAGE
CLOSE (OUTPUT), MODE=31
CLOSE (SYSPRINT), MODE=31
LR R1, R13            COPY WORKING STORAGE ADDRESS
L R10, 4(R13)         SAVE OLD SAVEAREA ADDRESS
L R9, =A(WORKLEN)     GET STORAGE LENGTH
STORAGE RELEASE, LENGTH=(R9), ADDR=(R1) RELEASE IT
LR R13, R10           RESTORE SAVEAREA ADDRESS
LM R14, R12, 12(R13)  RESTORE REGISTERS
LA R15, 8             SET RETURN CODE
BR R14                RETURN
*****************************************************************************
RETURN20 EQU *
PUT SYSPRINT, JFCBMSG1 ISSUE JFCB LOCATE FAIL MESSAGE
CLOSE (SYSPRINT), MODE=31
LR R1, R13            COPY WORKING STORAGE ADDRESS
L R10, 4(R13)         SAVE OLD SAVEAREA ADDRESS
L R9, =A(WORKLEN)     GET STORAGE LENGTH
STORAGE RELEASE, LENGTH=(R9), ADDR=(R1) RELEASE IT
LR R13, R10           RESTORE SAVEAREA ADDRESS
LM R14, R12, 12(R13)  RESTORE REGISTERS
LA R15, 8             SET RETURN CODE
BR R14                RETURN
*****************************************************************************
MSGPUT EQU *
ST R14, R14SAVE       SAVE THE RETURN ADDRESS
MVI RECBUF, C' '      SET FILL CHARACTER
MVC  RECBUFF+1(132),RECBUFF  INITIALIZE OUTPUT RECORD AREA
BCTR  R15,0  REDUCE LENGTH BY ONE FOR EX
EX  R15,MSGMVC  MOVE THE OUTPUT DATA
PUT  SYSPRINT,RECBUFF  WRITE THE OUTPUT RECORD
L  R14,R14SAVE  LOAD RETURN ADDRESS
BR  R14  RETURN
MSGMVC  MVC  RECBUFF(*-*) ,Ø(R1)  MOVE OUTPUT DATA

*******************************************************************************

CHARCHK  EQU  *
CLI  Ø(R9),C'A'  ALPHANUMERIC?
BL  CHARBAD  NO - ISSUE MESSAGE AND END
CLI  Ø(R9),C'I'  ALPHANUMERIC?
BNH  CHAROK  YES - SAVE IT
CLI  Ø(R9),C'J'  ALPHANUMERIC?
BL  CHARBAD  NO - ISSUE MESSAGE AND END
CLI  Ø(R9),C'R'  ALPHANUMERIC?
BNH  CHAROK  YES - SAVE IT
CLI  Ø(R9),C'S'  ALPHANUMERIC?
BL  CHARBAD  NO - ISSUE MESSAGE AND END
CLI  Ø(R9),C'Z'  ALPHANUMERIC?
BNH  CHAROK  YES - SAVE IT
CLI  Ø(R9),C'Ø'  ALPHANUMERIC?
BL  CHARBAD  NO - ISSUE MESSAGE AND END
CLI  Ø(R9),C'9'  ALPHANUMERIC?
BH  CHARBAD  NO - ISSUE MESSAGE AND END
CHAROK  EQU  *
XR  R15,R15  SET RETURN CODE
BR  R14  RETURN TO MAINLINE
CHARBAD  EQU  *
LA  R15,8  SET RETURN CODE
BR  R14  RETURN TO MAINLINE

*******************************************************************************

CLASSMVC  MVC  CLASSBUF(*-*) ,CIBDATA+1Ø COPY CLASS LIST
*******************************************************************************

* REGISTER EQUATES
R0  EQU  Ø
R1  EQU  1
R2  EQU  2
R3  EQU  3
R4  EQU  4
R5  EQU  5
R6  EQU  6
R7  EQU  7
R8  EQU  8
R9  EQU  9
R10  EQU  10
R11  EQU  11
R12  EQU  12
R13  EQU  13
R14  EQU  14
R15    EQU 15

*****************************************************************************

JBNMGOFF EQU 5
JB#OFF EQU 20
PRCFOFF EQU 35
STP0FF EQU 50
DDNDOFF EQU 65
CLSOFF EQU 80
DSNM0FF EQU 88

HDRREC1 DC C'JBN: XXXXXXXX JB#: XXXXXXXX PRC: XXXXXXXX
  DC CL(133-(L'HDRREC1+L'HDRREC2))'DSN: ' 

HDRREC2 DC C'STP: XXXXXXXX DDN: XXXXXXXX CLS: X
  DC CL(133-L'PARMMSG1)'WRTNAME,CLASSLIST'

PARMMSG1 DC CL133'XWTR001' - INVALID INPUT PARAMETER FORMAT'
  DC CL(133-L'PARMMSG1)'WRTNAME,CLASSLIST'

PARMMSG2 DC C'XWTR002 - A VALID WRTNAME IS 1-8 ALPHANUMERIC
  DC CL(133-(L'PARMMSG2+L'PARMM2B))'

PARMM2B DC C'CHARACTERS. CLASSLIST IS 1-36 CHARACTERS A-Z, 0-9.'
  DC CL(133-(L'PARMMSG2+L'PARMM2B))'

*****************************************************************************

ALLOCMSG DC CL133'XWTR015I - ERROR XXXXXXXX ALLOCATING DSN '
DALOCMSG DC CL133'XWTR016I - ERROR XXXXXXXX DEALLOCATING DSN '

*****************************************************************************

TERMMSG1 DC CL133'XWTR013I - SAPI TERMINATION FAILED RC=XXXXXXX'
TERMMSG2 DC CL133'XWTR014I - SAPI TERMINATION FAILED SSBRETN=XXXXXXX'

*****************************************************************************

RECFMSG1 DC CL133'XWTR019I - OUTPUT DATASET WAS NOT F(B) OR V(B)
  DC CL133'XWTR019I - OUTPUT DATASET WAS NOT F(B) OR V(B)

*****************************************************************************

JFCBMSG1 DC CL133'XWTR02OI - NO OUTPUT DATASET JFCB LOCATED'

*****************************************************************************

MSG3 DC C'XWTR031 - SAPI INVALID SEARCH ARGUMENTS'
MSG4 DC C'XWTR041 - SAPI UNABLE TO PROCESS NOW'
MSG5 DC C'XWTR051 - SAPI DUPLICATE JOBNAME'S
MSG6 DC C'XWTR061 - SAPI INVALID DESTINATION SPECIFIED'
MSG7 DC C'XWTR071 - SAPI AUTHORIZATION FAILED'
MSG8 DC C'XWTR081 - SAPI TOKEN MAP FAILED'
MSG9 DC C'XWTR091 - SAPI LOGIC ERROR'
MSG10 DC C'XWTR101 - SAPI INVALID CLASS'
MSG11 DC C'XWTR111 - SAPI BAD DISPOSITION SETTINGS'
MSG12 DC C'XWTR121 - SAPI DATASET DISPOSITION NOT UNIFORM'
MSG17 DC C'XWTR171 - SAPI REQUEST ABENDED. XWTR TERMINATING.'
MSG18 DC C'XWTR181 - SAPI UNKNOWN SSBRETN XXXXXXXX.'
  DC C'XWTR TERMINATING.'

MSG18L EQU *-MSG18

MSG98 DC C'XWTR098I - EXTERNAL WRITER XXXXXXXX INITIALIZED,'
MSG99 DC C'XWTR099I - EXTERNAL WRITER XXXXXXXX TERMINATED.'

*****************************************************************************

INPUT DCB MACRF=(GM),DDNAME=TEMP,LRECL=133,DSORG=PS,EODAD=INDONE
OUTPUT DCB MACRF=(PM),DDNAME=OUTPUT,DSORG=PS
SYSPRINT DCB MACRF=(PM),DDNAME=SYSPRINT,LRECL=133,DSORG=PS
*** THE STATIC COPY OF THE DYNAMIC ALLOCATION PARAMETER LIST***
* FOR ALLOCATING A JES DATASET THAT RESIDES ON SPOOL AND WHOSE OUTPUT HAS BEEN DIRECTED THEREBY AN EXTERNAL WRITER
S992 DC A('80000000'+S992RB)
S992RB DC 'X'14'
S992VERB DC 'X'01'
S992FLG1 DC 'X'0000'
S992ERR DC 'X'0000'
S992INFO DC 'X'0000'
S992TXTP DC AL4(S992TUPL)
   DC XL4'00'
S992FLG2 DC XL4'00'
S992TUPL DC AL4(TU2001)
TU202 DC AL4(TU2002)
TU203 DC AL4(TU2003)
TU204 DC AL4(0)
TU2001 DC 'X'0055','X'0001','X'0008' //DDN2 DD
DDN2 DC CL8' '
TU2002 DC 'X'0002','X'0001','X'002C' DSN=SSS2DSN
DSN2 DC CL44' '
TU2003 DC 'X'005C','X'0001','X'004,C' INDICATE WHICH SUBSYS
S992LN EQU *-S992

*** THE STATIC COPY OF THE DYNAMIC UNALLOCATION PARAMETER LIST***
* FOR UNALLOCATING THE JES EXTERNAL WRITER DATASET
S994 DC A('80000000'+S994RB)
S994RB DC 'X'14'
S994VERB DC 'X'02'
S994FLG1 DC 'X'0000'
S994ERR DC 'X'0000'
S994INFO DC 'X'0000'
S994TXTP DC AL4(S994TUPL)
   DC XL4'00'
S994FLG2 DC XL4'00'
S994TUPL DC AL4(TU4001)
TU4001 DC 'X'0001','X'0001','X'0008' //DDN4 DD
DDN4 DC CL8' '
S994LN EQU *-S994

******************************
LTORG
* SAVEAREA AND OTHER PROGRAM STORAGE
WORKAREA DSECT
SAVEAREA DS 18F
R14SAVE DS F
DYNPRMS2 DS 0D,CL(S992LN)
DYNPRMS4 DS 0D,CL(S994LN)

64 © 2002. Xerphon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
TIOADDR DS F
SSOBPTR DS F
REPLYECB DS F
SELECB DS F
DBL1 DS 2D
DBL2 DS 2D
SSNMSAVE DS CL8
WTRNM DS CL8
CLASSLST DS CL36
REPLY DS CL1
TEMPDDN DS CL8
SSOBAREA DS @D,CL(SSOBHSIZ+SSS2SIZE)
CLASSBUF DS CL36
******************************************************************************
COMMADDR DS F
ECBLIST DS 2F
EXTRWRK DS @D,CL(EXTLN)
******************************************************************************
SWEPAPTR DS F
EPA DS CL28
SWAPARMS SWAREQ MF=L
******************************************************************************
LRECL DS CL4
RECBUFF DS CL32760
WORKLEN EQU *-WORKAREA
******************************************************************************
PRINT NOGEN
IEFSSOBH
SSOBGN EQU *
PRINT GEN
IAZSSS2 DSECT=YES
PRINT NOGEN
IEFJESCT TYPE=DSECT
CVT DSECT=YES
IKJTCB
IEZJSCB
IEFJSSIB
MAPCOM DSECT
IEZCOM
MAPCIB DSECT
CIB IEFZCIB
MAPJFCB DSECT
IEFJFCBN LIST=YES
IEFZB505
TIOT DSECT
IEFTIOT1
END

Systems Programmer
(Canada) © Xephon 2002
Matching a filename against a pattern

The following REXX program was created as a subroutine for several other procedures. It matches a filename against a generic pattern, according to the ISPF conventions. It needs two arguments, separated by spaces – one is the filename, the other is the pattern. The order in which they are specified is not important. The EXEC returns 0 if the filename fits the pattern, -1 otherwise. You can also execute the EXEC directly (not as a subroutine), in which case the return code is displayed on the screen.

A pattern is a string containing literal characters, single asterisks (*), double asterisks (**), and percentage signs (%). It can also contain dots (.) separating the dataset qualifiers. Their meaning is as follows:

- % – only one character.
- * – any number of characters, including none. If it stands alone on a qualifier (.*.), then it means a qualifier must exist in that position.
- ** – any number of qualifiers, including none, if it stands alone on a qualifier (.**.).

For example, the following are valid patterns. In front of each pattern I also put a string to check the filename and indicate whether they match (0) or not (-1):

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Filename</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH*.IS.*.T%</td>
<td>THIS.IS.A.TEST</td>
<td>0</td>
</tr>
<tr>
<td>TH%%%%.<em>.AL</em>.A*</td>
<td>THIS.IS.ALSO.A.TEST</td>
<td>0</td>
</tr>
<tr>
<td>THIS.**.GOOD</td>
<td>THIS.IS.ALSO.GOOD</td>
<td>0</td>
</tr>
<tr>
<td>A%%.HERE%%.*</td>
<td>AND.HERE.IS.A.FAILURE</td>
<td>-1</td>
</tr>
<tr>
<td>AND. <em>.</em>.*</td>
<td>AND.HERE</td>
<td>-1</td>
</tr>
<tr>
<td>AND<em>NOT</em>ENAME</td>
<td>ANTHISISNOTAFILENAME</td>
<td>0</td>
</tr>
</tbody>
</table>

The program does not check for valid filenames. The strings passed can be almost anything, as long as only one of them is a pattern, which means it contains ‘*’ or ‘%’. Also, the strings need not be separated into qualifiers – a long string of characters will be accepted as well. The only restrictions that apply are that the string cannot contain spaces, ‘?’, or ‘;’. This is because the EXEC uses them internally. When the string is dot-separated, as filenames are, each qualifier is matched independently.
The algorithm used in the EXEC is probably not very conventional, but works very well. Basically, for each qualifier, it transforms the pattern of literals and wildcards into a string of literals, variables, and positions. This transformed pattern is used as a template for REXX’s PARSE VAR instruction. After the execution, by means of an INTERPRET, each variable receives a value from the string being checked. If there is a match, replacing these variables by their values, in conjunction with any literals that exist, will equal the checked name.

This is just an outline of what I do; the real process is a little more complicated, because there are minimum lengths to take care of (%%%% must match exactly four characters, while %%%%* must match four or more), there are relative positions to be considered, etc. The only drawback with this approach is that the algorithm produced is not directly translatable to other programming languages that lack REXX facilities. But this is a REXX EXEC, created as support for other EXECs, and so I decided to exploit its potential.

```rexx
/* REXX MVS */
/* PATTERN - This program matches a filename to a generic pattern. */
/* It should be used as a subroutine. The entry parameter is a */
/* string containing 2 words, the filename to check and the pattern */
/* separated by one or more spaces. Their order is irrelevant. */
/* */
/* Filename is a fully-qualified dataset name */
/* Pattern follows the ISPF style convention: */
/* * - any number of characters */
/* % - one character */
/* .* - one qualifier */
/* ** - any number of qualifiers (including none) */
/* */
/* RC: 0 if there is a match, -1 otherwise. */
/* If this program is called as a subroutine, then RC is returned, */
/* otherwise RC is "said". */
/* */
/*======================================================================*/
arg arg1 arg2.
if pos("*",arg1) > 0 | pos("%",arg1) > 0 then do
  file = arg2
  pattern = arg1
end
else do
  file = arg1
```

pattern = arg2
end
any = Ø
returncode = Ø
do alpha = Ø
  parse var pattern pl "." pattern
  if pl = "" then pl = ";"
  if pl = "**" then do
    if pattern = "" then do
      leave alpha
    end
    else do
      any = 1
      iterate alpha
    end
  end
  if right(pl,1) = "*" & pattern = "" & any = Ø then do
    any = 2
  end
do beta = Ø
  parse var file f1 "." file
  if f1 = "" then f1 = ";"
  if f1 = ";" & pl = ";" then leave alpha
  call check_qualifier pl f1
  returncode = result
  if any = Ø then do
    if returncode = Ø then do
      iterate alpha
    end
    else do
      leave alpha
    end
  end
  if any = 1 then do
    if returncode = Ø then do
      if right(pl,1) = "*" & pattern = "" then do
        leave alpha
      end
      else do
        any = Ø
      end
      iterate alpha
    end
    else do
      if f1 = ";" then do
        leave alpha
      end
      else do
        iterate beta
    end
  end
end
end
end
if any = 2 then do
 leave alpha
end
end

parse source . calltype .
if calltype = "COMMAND" then say returncode
else return returncode
exit

/============================================================================*/

check_qualifier: procedure

arg str1 str2

if str2 = ";" then return -1
if str1 = ";" then return -1
if str1 = "*" then return Ø

v = Ø
str3 = ""
prv = ""
no_pos = Ø

do forever
 p = pos("%",str1)
 if p = Ø then leave
 str1 = overlay("%",str1,p)
end

do forever
 p = pos("%%",str1)
 if p = Ø then leave
 str1 = overlay("%%",str1,p)
end

do forever
 p = pos("**",str1)
 if p = Ø then leave
 str1 = delstr(str1,p,1)
end

do k = 1 to length(str1)
 select
when substr(str1,k,1) = '%' then do
  do k1 = k + 1 to length(str1)
    if substr(str1,k1,1) <> '%' then leave k1
  end
  v = v + 1
  if no_pos = 1 then do
    str3 = str3"?var."v
  end
  else do
    str3 = str3"?="k"?var."v
  end
  if substr(str1,k1,1) <> '*' then do
    if no_pos = 1 then do
      str3 = str3"?"
    end
    else do
      str3 = str3"?="k1"?"
    end
    len_eq.v = k1 - k
    len_ge.v = 0
    k = k1 - 1
  end
  else do
    len_ge.v = k1 - k
    len_eq.v = 0
    k = k1
  end
end
when substr(str1,k,1) = '*' then do
  no_pos = 1
  v = v + 1
  str3 = str3"?var."v"?"
  len_eq.v = 0
  len_ge.v = 0
end
otherwise do
  str3 = str3"?"
  do k1 = k to length(str1)
    if substr(str1,k1,1) = '%',
      substr(str1,k1,1) = '*' then leave k1
    str3 = str3||substr(str1,k1,1)
  end
  str3 = str3"
  k = k1 - 1
end
end
str3 = space(str3,0)
str3 = translate(str3," ","?")
Why not share your expertise and earn money at the same time? MVS Update is looking for technical articles and hints and tips that experienced MVS users have written to make their life, or the lives of their users, easier. We would also be interested in articles about performance and tuning.

We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2. Why not call or write now for a free copy of our Notes for contributors? Alternatively, you can now download a copy from our Web site. Point your browser at www.xephon.com/nfc.
Data 21 has announced JES2Mail and JES2FTP. JES2Mail is a mainframe-based tool that transforms print files into formats such as Adobe PDF, HTML, RTF, and comma-delimited, and ‘pushes’ them directly to end-users via e-mail. JES2FTP automatically publishes JES reports on a Web site as HTML or Adobe PDF documents. It also sends e-mail notifications (with links to the reports!) to end users.

For further information contact:
Data 21, 3510 Torrance Blvd, Suite 300, Torrance, CA 90503, USA.
Tel: (310) 792 1771.

* * *

MacKinney Systems has announced VTAM Virtual Printer (VVP) Release 1.3, which is used to route print from CICS TS to the JES queue for OS/390 and z/OS or to the POWER Queue for VSE.

VVP is said to require no programming changes and it runs as a VTAM application. All printer maintenance may be done through on-line panels.

For further information contact:
MacKinney Systems, 2740 South Glenstone, Suite 103, Springfield, MI 65804, USA.
Tel: (417) 882 8012.

* * *

Compuware has announced general availability of its STROBE 2.5 and introduced iSTROBE, a new browser-based tool for analysing application performance information.

STROBE 2.5 introduces new functions that measure abnormal program behaviour. iSTROBE enables interactive analysis of enterprise applications based on WebSphere MQ, CICS/TS, Visual Age for Java, DB2, IMS, and Unix System Services.

STROBE enables sites to determine where and how applications use resources in OS/390 and z/OS environments. Incorporating STROBE measurement into key phases of the application life cycle such as development, test, and production is designed to help ensure that applications run efficiently and responsively and that no performance problems are unintentionally introduced.

Utilizing STROBE’s performance profiles, iSTROBE is said to identify the performance characteristics of complex application problems. It augments STROBE by increasing and simplifying the identification of performance opportunities within the STROBE Performance Profile.

For further information contact:
Compuware, 31440 Northwestern Highway, Farmington Hills, MI 48334-2564, USA.
Tel: (248) 737 7300.

* * *

For a limited time, the purchase of selected Tivoli NetView for z/OS Version 5.1 licences will qualify the buyer for a two-day on-site Tivoli NetView Migration and Planning Assistance Package that has an approximate value of $6,000.

For further information contact your local IBM representative.