146 MVS update

November 1998

In this issue

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

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

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

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

DASD performance monitor

INTRODUCTION

In all computing environments, ranging from large mainframes to the humble home PC on which I am typing this article, there is a marked discrepancy between the rate of performance increase between instruction processors and the various formats of input/output devices, but especially Direct Access Storage Devices (DASD) or disks.

In the DASD world, it seems that access speed reached a plateau some years ago, which has proven expensive and difficult for manufacturers to reach beyond. Storage density has increased dramatically, but the basic speed of access of DASD devices has remained in the neighbourhood of 10 milliseconds for a decade, an unprecedented length of time in an industry that in other areas expects to continue to see exponential increases in performance.

Until a new storage paradigm emerges from the laboratories, manufacturers have only two methods to improve DASD performance – specifically by making the disk spin at higher RPM or the access arm move faster, and these technologies seem to have been pushed to their limits. Hence the emphasis on avoiding I/O altogether, or doing as much I/O as possible to electronic storage in its various forms of central storage, expanded storage, and cache. But try as hard as we may to achieve optimal levels of I/O avoidance, those spinning platters still bear a huge load in most computing environments, so it is imperative to ensure that when I/O does go to disk, we know exactly how those disks are performing.

With these considerations in mind, I have developed a very simple, yet powerful real-time DASD performance monitor in REXX, which I find indispensable when I require a quick snapshot of what is happening in the DASD subsystem.

The heart of any analysis of I/O in an IBM mainframe environment is the Channel Measurement Block (CMB), which IBM introduced as part of the XA architecture in the 1980s. A CMB exists for each addressable device in the system, and counters are maintained there for the number of start subchannel (SSCH) operations, and the time that the device was in each of three states relating to the execution of an I/O.

This article is not a tutorial on how I/O works in the IBM environment, so I will keep this discussion to as few words as possible. The three states are connect, pending, and disconnect:

- During the connect state, the channel subsystem is connected to the I/O device and data flows between the two, either a Channel Command Word (CCW) telling the device what is required of it for this I/O operation (read/write, etc) or the data involved.
- The disconnect state is when the channel subsystem disconnects from the device having told it what to do, and the device does whatever electromechanical things it needs to do to service the request. Because the channel is purely electronic, it is much faster than any DASD, the channel can use the time saved to talk to other devices, allowing you to drive many devices with few channels.
- The pending state is usually much shorter than connect or disconnect, and represents time when other components of the I/O subsystem are delaying the I/O process.

Each of these states is interesting to monitor individually, but it is the sum of the three averaged by the number of SSCH operations that gives the overall service time for the device. This is the basis for deciding whether the performance of the device is approaching that of the manufacturer's specifications or not.

Any performance analyst will immediately, however, point out that service time should not to be confused with response time. The response time of a device, as measured by products like RMF, takes into account the service time as well as any software queueing time. RMF can do this analysis because of the started task component, which is timer driven to record how many I/Os are queued on each Unit Control Block (UCB). Queueing theory is then used to predict how long each I/O waits based on the length of the queue and the average time to process each I/O. Actually one can also use queueing theory to calculate response time from service time and arrival rate (SSCH rate), and in my experience the results mirror RMF's values quite closely for I/O profiles characterized by fairly constant SSCH rates, but because it is not possible to guarantee such a profile I have omitted this complication from my routine.

The REXX program DASDPRF reads information from the CMBs on the system, stores it in ISPF tables, and then formats it for presentation

on an ISPF panel. Note the comment on the first line of code, where the variable ‘stdev’ is initialized. This variable represents the address of the first device that will be displayed on the ISPF panel. This value should be modified to represent whatever device you would like to be the first in the display.

The REXX program should of course be saved in a dataset that forms part of your TSO session’s SYSEXEC or SYSPROC concatenations, while the two panels following should be in an ISPPLIB dataset.

DASDPRF REXX

```
/*------------------------------ REXX --------------------------------*/
/* Function   : Read H/W maintained CMB to get device util.          */
/*-------------------------------------------------------------------*/
/* First find CMB indexes of volumes required                        */
/*-------------------------------------------------------------------*/
stdev = '100'             /* <====== Modify to desired value.        */
change_stdev:
sdev = stdev
devs = 0; ref = 'Y'
address ispexec,
  "tbcreate devtab names(name voli cmbi ssch conn pend disc,
         ossch oconn opend odisc dssch rssch rconn rpend rdisc serv),
   nowrite replace"
addr = d2x(16)
cvt  = storage(addr,4)
addr = d2x(c2d(cvt) + c2d(x2c(04e4)))
ucb  = storage(addr,4)
add  = 'n'
last = 'n'
ucbz = x2c(00000000)
do until last = 'y'
  addr = d2x(c2d(ucb) - c2d(x2c(0020)))
  ucbx = storage(addr,80)
  name = substr(ucbx,46,3)
  if name = stdev then add = 'y'
  if add = 'y' then
    do
    if substr(name,1,2) ¬= substr(stdev,1,2) then
      do
      add = 'n'
      last = 'y'
      end
    end
  voli = substr(ucbx,61,6)
  cmbi = x2d(c2x(substr(ucbx,3,2)))
  if add = 'y' then
    do
```

```
    address ispexec "tbadd devtab"
    devs = devs + 1
    end
  addr = d2x(c2d(ucb) + c2d(x2c(0008)))
  ucb  = storage(addr,4)
  if ucb = ucbz then last = 'y'
end
address ispexec "tbtop devtab"
/*-------------------------------------------------------------------*/
/* Now go look for the required CMBs                                 */
/*-------------------------------------------------------------------*/
addr = d2x(16)
cvt  = storage(addr,4)
addr = d2x(c2d(cvt) + c2d(x2c(025c)))
rmct = storage(addr,4)
addr = d2x(c2d(rmct) + c2d(x2c(0118)))
cmct = storage(addr,4)
addr = d2x(c2d(cmct) + c2d(x2c(0014)))
cmb  = storage(addr,4)
/*-------------------------------------------------------------------*/
/* Initialize table - CMB is 32 bytes                                */
/*-------------------------------------------------------------------*/
elap = time('E')
addr = d2x(c2d(cmb))
do j = 1 to devs
  address ispexec "tbskip devtab"
  address ispexec "tbget devtab"
  addr = d2x(c2d(cmb) + (cmbi * 32))
  cmbx = storage(addr,32)
  ssch = c2d(substr(cmbx,1,2))
  conn = c2d(substr(cmbx,5,4))
  pend = c2d(substr(cmbx,9,4))
  disc = c2d(substr(cmbx,13,4))
  address ispexec "tbput devtab"
end
/*-------------------------------------------------------------------*/
/* Main loop                                                         */
/*-------------------------------------------------------------------*/
do forever
  address ispexec "tbtop devtab"
  if ref = 'N' then elap = time('E')
    else elap = time('R')
  addr = d2x(c2d(cmb))
  trat = 0
  do j = 1 to devs
    address ispexec "tbskip devtab"
    address ispexec "tbget devtab"
    if ref ¬= 'N' then
      do
      ossch = ssch
      oconn = conn
      odisc = disc
      opend = pend
```

```
      end
    addr = d2x(c2d(cmb) + (cmbi * 32))
    cmbx = storage(addr,32)
    ssch = c2d(substr(cmbx,1,2))
    conn = c2d(substr(cmbx,5,4))
    pend = c2d(substr(cmbx,9,4))
    disc = c2d(substr(cmbx,13,4))
    dssch = ssch - ossch
    if dssch < 0 then dssch = 65535 + dssch
    rssch = format(dssch / elap,3,2)
    trat = trat + rssch
    if dssch = 0 then
      do
      rconn = format(0,3,0)
      rdisc = format(0,3,0)
      rpend = format(0,3,0)
      end
      else
      do
      rconn = format((((conn - oconn) * 128) / 1000) / dssch,3,0)
      rdisc = format((((disc - odisc) * 128) / 1000) / dssch,3,0)
      rpend = format((((pend - opend) * 128) / 1000) / dssch,3,0)
      end
    serv = format(rconn + rdisc + rpend,3,0)
    address ispexec "tbput devtab"
  end
  address ispexec "tbtop devtab"
  address ispexec "tbdispl devtab panel(dasdprfp)"
  if rc ¬= 0 then
    do
    address ispexec "tbclose devtab"
    exit 0
    end
  address ispexec "vget (stdev ref)"
  if stdev ¬= sdev then
    do
    address ispexec "tbclose devtab"
    signal change_stdev
    end
end
exit 0
```

DASDPRFP PANEL

```
)attr
 ! type(output) color(green) just(right)
 # type(output) color(yellow) just(right)
 $ type(output) intens(high)
 " type(text) color(turq)
   type(text) skip(on) intens(low)
)body expand(@@)
%@-@  DASD Activity ex-CMB  @-@
```

```
%COMMAND ===>_ZCMD                                          %SCROLL ===>_AMT
+
%Refresh ===>_Z%Display from ===>_Z  %Total Rate ===>_trat %Interval ===>_ELAP
+
%
"  Unit Volser ]   SSCH ]   Rate ]   Conn ]   Pend ]   Disc ]   Serv
"  ------------]--------]--------]--------]--------]--------]--------
)model
   !Z  !Z     "]#Z     "]#Z     "]#Z     "]#Z     "]#Z     "]#Z     "
)init
.help = dasdprfh
.zvars = '(ref stdev name voli dssch rssch rconn rpend rdisc serv)'
&zcmd = &z
&ztdmark = ' '
if (&stdev = ' ')
  &stdev = '34Ø'
if (&ref = ' ')
  &ref = 'y'
)proc
vput (stdev ref)
)end
```

DASDPRFH PANEL

```
)attr
 $ type(output) intens(high) just(right)
)body expand(@@)
%@-@  DASD Activity ex-CMB  @-@
%COMMAND  ===>_ZCMD
+
+       This function looks at the current device activity
+       gathered from the Channel Measurement Blocks.
+
+       Elapsed seconds refers to the interval during which other
+       fields were calculated.
+
+       Conn, Pend and Disc are displayed in milliseconds.
+
+       SSCH - Start/resume subchannel operations in the interval
+       Rate - Start/resume subchannel rate (SSCH / ELAPSED)
+       Conn - Device connected to a channel path
+       Pend - Interval between acceptance of SSCH at the channel
+            - and acceptance of the first command at the device
+       Disc - Device logically disconnected from the subchannel
+       Serv - Device service time, Conn + Pend + Disc
+
)init
)proc
&zcont = dasdprfh
)end
```

Testing REXX EXECs

THE PROBLEM

Often whilst coding and testing a new TSO REXX routine, I find I need to make some small change to the code, try it out, and then revert back to the original. If I were working with a batch JCL deck, this would present no problem, because I would merely EDIT the required deck, make the changes, issue the SUB edit command, and then CANCEL out of the member without saving the changes. Unfortunately, with REXX there is no equivalent to the SUB command. I therefore decided to write my own EDIT macro to serve the same purpose. The result was the SUBE edit macro.

A SOLUTION

SUBE is invoked from the command line within edit. Its format is:

```
SUBE {TRACE} {arguments}
```

It works by using the edit macro LINE function to extract the in-storage lines from the currently edited member into variables. A temporary PDS is then created, and the variable data lines are then written into a member in this temporary PDS with the same name as the member being edited.

The TSO ALTLIB command is then invoked to activate the temporary PDS as a task-level EXEC library, and the routine is then invoked along with any arguments specified on the SUBE command line. If the keyword 'TRACE' is specified on the SUBE command line, then the TSO EXECUTIL command is invoked prior to calling the REXX routine to switch into interactive debug. This provides a quick way of tracing an existing routine without changing the code.

As a safety feature, SUBE will check if the called routine has the same name as itself. If so, a warning message will be issued requiring a specific reply before the process will continue. This is because, if you recursively invoke SUBE (ie use SUBE to run SUBE itself), you are likely to send your TSO session into an endless loop resulting in an abend either from lack of storage, or from the excessive allocations of temporary PDS caused by each invocation of SUBE.

This routine was written and tested under OS/390 1.3 TSO and ISPF (equivalent to TSO/E 2.5 and ISPF 4.4), but should work without change under earlier versions. You may need to check and modify the ALLOC statements for the temporary PDS to conform to your site's naming and/or SMS conventions.

```
/*REXX
  Member   : SUBE - EDIT MACRO
  Function : Allows currently edited member to be executed as
             a REXX EXEC without saving. (c/f SUB for JCL)
  Format   : SUBE {TRACE} parms
             where:-
                TRACE - if specified, will cause the REXX to be
                        interactively traced for debugging
                parms - any parameters to be passed to the REXX
*/
parse source . . myname .                    /* extract my routine name */
"ISREDIT MACRO (parm1,PARM2)"                /* I'm an edit macro!      */
if rc = 20 then
do                                    /* someone called me as TSO xxxx!!!*/
  say myname': Invalid invocation - command is an EDIT macro,',
      'do not prefix with ''TSO''.'
  exit 99
end
/*if 1st word not 'TRACE' then assume it's part of parms*/
trace = translate(parm1)
if trace ¬= 'TRACE' then
   parms = parm1 parm2
else
   parms = parm2
"ISREDIT (MEMB) = MEMBER"         /* get member name being edited    * /
if memb = '' then memb = '$TEMP'  /* no member? - sequential dataset
                                  so create temporary member name     */
if memb = myname then              /* oh-oh! EXEC is same name as me!
                               therefore likely recursive invocation
                               which is NOT GOOD!                    */
do
   say myname': WARNING!! EXEC has same member name as this routine!'
   say myname': Recursive invocation of 'myname' will cause an endless'
   say myname': loop and possibly ABEND your TSO session!'
   say myname':'
   say myname': If this is NOT the case, then reply ''CONTINUE'' to '
   say myname': proceed. Any other reply will terminate.'
   pull reply
   if reply ¬= 'CONTINUE' then exit
   say myname': Are you sure? (''YES'' to proceed).'
   pull reply
   if reply ¬= 'YES' then exit
end
"ISREDIT (RECL) = LRECL"          /* get record length of data        */
"ISREDIT (RECF) = RECFM"          /* get record length of data        */
```

```
'ISREDIT (lastline) = LINENUM .ZL'     /* how many lines?             */
do cnt = 1 to lastline
 'ISREDIT (out'cnt') = LINE 'cnt       /* assign each one to variable * /
end
alloc:
/* calculate space required for lines.
   bytes = record length*number of lines
   tracks = bytes/56664        56664 = # of bytes/track for 3390 DAS  D
   **NOTE!** use 47476 for 3380, or 19069 for 3350

   Will be used to allocate temporary PDS. EXEC will be written
   to this and then executed from it. This leaves the 'real' EXEC
   alone.
*/
allc_temp:
bytes = recl*cnt
traks = (bytes%56664)+1
/*
  Allocate temporary pds to hold in-storage lines
*/
tddname='SUB'time(s)                   /* generate unique ddname  */
tdsname= userid()'.TEMP.SUBE.'tddname  /* build dataset name      */
"ALLOC F("tddname") NEW CATALOG TR SP("traks",1) DIR(5) REUSE" ,
"DA("tdsname")",
"DSORG(PO) LRECL("recl") RECFM("recf")"
"FREE FI("tddname")"                   /* free up to catalog      */
/*
  now write in-storage exec lines to member in temp pds
*/
"ALLOC FI("tddname") DA('"tdsname"("memb")') OLD REUSE"
"EXECIO * DISKW "tddname" (STEM OUT FINIS"
"FREE FI("tddname")"
/*
   Now use ALTLIB to set temp pds as valid EXEC library.
   If TRACE specified,
   then invoke EXECUTIL first to start interactive trace
*/
"ALTLIB ACT APPL(EXEC) DA("tdsname")"
if trace = 'TRACE' then "EXECUTIL TS"
/*
  invoke routine - REXX will translate variable and, because the
  resultant string is not prefixed with a recognized REXX verb,
  will pass the whole string as a host command as if it were
  TSO xxxxx
*/
routine = memb parms
(routine)
exrc = rc          /* get return code from EXEC for message */
"EXECUTIL TE"      /* stop tracing                          */
"ALTLIB RESET"     /* drop temp library search              */
call msg 'OFF'     /* suppress delete message               */
"DELETE "tdsname   /* delete temp dataset                   */
```

```
/*
   set REXX result as ISPF message
*/
zedsmsg = memb" rc="exrc
zedlmsg = "EXEC "memb" Ended with return code of "exrc
"ISPEXEC SETMSG MSG(ISRZ001)"
exit
```


Building DSECT Assembler macro listings

THE PROBLEM

I have found over the years that, as an MVS systems programmer who regularly codes in Assembler language, I need to frequently reference offsets within MVS control blocks when analysing dumps as well as writing new code. The multiple volumes of the IBM MVS/ESA data areas are a problem to handle in hardcopy format. The softcopy version of these manuals, according to my understanding, are no longer being provided on CD-ROM (so if you have them, you might not want to delete them from your system). Even if IBM is providing them on tape cartridge, each installation has to refresh the books on DASD with tape copies every time a change occurs.

A SOLUTION

As an alternative to this method, I developed a REXX EXEC and CLIST to perform an interactive assembly of any required macro, including the ability to specify any optional invocation parameters, storing the resulting listing into a PDS so the listings can be referenced any time they are needed. The added benefit of this simple system is that any time IBM maintenance is applied to the operating system or components (usually via the SMP/E product) that updates any macro, you can simply run the EXEC that rebuilds the listing for the affected element. This process will work for any macro library that you would care to include the SYSLIB DD statement concatenation, which the

Assembler uses to resolve macro invocations – such as JES2, DFSMS, ISPF, or SDSF. For that matter, the process also applies to any other OEM software product that comes with a macro library for which it would be handy to have a listing of the macro expansion on-line.

You invoke the REXX EXEC with the name of the macro you wish to assemble along with any optional macro invocation parameters. For example, in order to build a listing of the CVT macro which will be stored in the PDS as member CVT, you would type:

```
XDSECT CVT LIST=YES,DSECT=YES
```

The EXEC will assemble the macro using the IBM High-Level Assembler after wrapping a few other Assembler directives in front of and behind the macro invocation (although you could modify the code to use Assembler H). Upon assembly, the EXEC invokes the ISPF editor (after checking that it is running in an ISPF environment) on the dataset pointed to by the SYSPRINT DD statement, and uses an ISPF edit macro to clean up some of the superfluous lines in the listing and shift the lines to the left for easier on-line viewing. The edited listing is then saved back into the same member, thus providing a reference for fields and offsets within the macro. ISPF services are also used to add ISPF statistics on the member after it is saved to provide an audit trail of when and by whom the macro listing was generated.

Using this process, I have created and populated a PDS that currently contains over 150 expanded mapping macros. These include many of the mappings for common control blocks used by JES2, RACF, GRS, SMS, SRM, ASM, RSM, and MVS job and data management. Even our CA-1 (TMS) tape management record descriptions provided by the vendor have been included.

The PDS which stores the listings as members needs to be pre-allocated with RECFM=FBM, LRECL=133, and DSORG=PO (the RECFM and LRECL are Assembler requirements for the SYSPRINT DD). The amount of space and directory blocks needed depends on how many members are intended to be stored for reference. The blocksize can be allowed to default to a system-determined blocksize, or coded as any reasonable multiple of the LRECL of 133. The SYSIN input to the Assembler is assumed to be a PDS whose dataset name is userid.ASM, but further logic can be coded to check through any

number of datasets using the REXX LISTDSI function to verify the existence of the datasets and/or member.

XDSECT REXX EXEC

```
/* --------------------  REXX PROCEDURE  ---------------------- */
arg macname opt1 opt2 opt3
parse upper arg
if sysvar(SYSISPF)
¬= 'ACTIVE' then do
   say 'This procedure can only be run in an ISPF environment'
   exit 16
end
if length(macname) = Ø then return(12)
dat.Ø = 5
dat.1 = '         ' || 'PRINT GEN'
if    substr(macname,1,1) = '$' then
      dat.2 = '          ' $HASPEQU
else  dat.2 = '*'
dat.3 = '         ' || macname opt1 opt2 opt3
dat.4 = '         ' || 'IEZBITS       in case needed by macro'
dat.5 = '         ' || 'END'
"ALLOC DD(SYSIN)    DA(ASM(XDSECT)) SHR REUSE"
"EXECIO * DISKW SYSIN (FINIS STEM dat."
"ALLOC DD(SYSLIB)   DA('SYS1.MACLIB' 'SYS1.MODGEN' 'SYS2.MACLIB'" ||,
       " 'SYSTEMS.ESA.PVTMACS') SHR REUSE"
"ALLOC DD(SYSPRINT) DA('SYSTEMS.DSECT.LIST("macname")') SHR REUSE"
"ALLOC DD(SYSLIN)   NEW DEL SP(5 1) CYL UNIT(VIO) REUSE"
"ALLOC DD(SYSPUNCH) DUMMY REUSE"
"ALLOC DD(SYSUT1)   NEW DEL SP(5 1) CYL UNIT(VIO) REUSE"
"CALL 'HLA.SASMMOD1(ASMA9Ø)' 'TERM,ALIGN,NOOBJECT,DECK,XREF(SHORT)'"
if rc <= 4 then DO
   address ISPEXEC "EDIT DATASET('SYSTEMS.DSECT.LIST("macname")')" ||,
           " MACRO(ISRDSECT)"
   "ALLOC DD(SYSPRINT) DA('SYSTEMS.DSECT.LIST') SHR REUSE"
   address ISPEXEC "LMINIT   DATAID(OU) DDNAME(SYSPRINT)"
   address ISPEXEC "LMMSTATS DATAID("ou") MEMBER("macname")"
   address ISPEXEC "LMFREE   DATAID("ou")"
end
"FREE ALL"
"ALLOC DD(SYSIN)    DA(*) REUSE"
"ALLOC DD(SYSPRINT) DA(*) REUSE"
```

ISRDSECT ISPF EDIT MACRO

```
ISREDIT MACR
/* ISPF EDIT MACRO FOR MVS DSECT ASSEMBLER LISTING FORMATTING */
CONTROL MSG NOFLUSH
ISREDIT NUMBER = OFF
```

```
ISREDIT BOUNDS = 1 133
ISREDIT CHANGE P'.' ' ' ALL
ISREDIT CHANGE 'I' ' ' 1 1 ALL
ISREDIT CHANGE 'Ø' ' ' 1 1 ALL
ISREDIT FIND 'PRINT GEN' FIRST
ISREDIT DELETE ALL NX .ZFIRST .ZCSR
ISREDIT FIND ' END ' LAST
ISREDIT (LDF) = LINENUM .ZCSR
SET LDF = &LDF + 2
IF &LASTCC = Ø THEN ISREDIT DELETE ALL NX &LDF .ZLAST
ISREDIT BOUNDS = 9 133
ISREDIT (TOP) = LINENUM .ZFIRST
ISREDIT (BOT) = LINENUM .ZLAST
DO I = &TOP TO &BOT
   ISREDIT SHIFT ( &I 33
END
DO I = &TOP TO &BOT
   ISREDIT FIND 'PAGE' 8Ø 85 ALL
   IF &LASTCC > Ø THEN GOTO RESET
   ISREDIT (LDF) = LINENUM .ZCSR
   SET LDL = &LDF + 2
   ISREDIT DELETE &LDF &LDL
END
RESET: +
ISREDIT RESET
ISREDIT SAVE
ISREDIT END
```


Terminating tasks within address spaces

INTRODUCTION

We recently ran into a situation where a system task (HSM) had a problem which caused an exclusive ENQ on an HSM resource. The exclusive ENQ prevented other tasks in the HSM address space from running. We did not want to recycle the HSM address space, so we had to develop some mechanism to get HSM to free the ENQed resource. We decided to write a small program to terminate the task (represented by a TCB control block) under which the ENQ was issued by issuing a CALLRTM macro against the offending address space and TCB.

The CALLRTM macro is documented in the *MVS/ESA Authorized Assembler Services Reference*. To use the CALLRTM macro, the issuer must be running authorized (that is in an APF authorized library and linkedit with the AC(1) attribute) since CALLRTM can only be invoked by programs running in supervisor state and PSW key 0.

In order to use the TCBTERM program, two pieces of information are necessary – the address space identifier (ASID) of the address space whose TCB is to be terminated, and the specific address of the TCB to be terminated. Both can be obtained by issuing an MVS operator command to display information about specific ENQ resources; in the case of our HSM problem, the major and minor resource names are ARCGPA and ARCCAT, respectively. The syntax of the MVS command and its output follow:

```
D GRS,RES=(ARCGPA,ARCCAT)
S=STEP    ARCGPA   ARCCAT
SYSNAME   JOBNAME  ASID      TCBADDR    EXC/SHR     OWN/WAIT
SYSB      HSM      0040      0099FE88   SHARE       OWN
SYSB      HSM      0040      009B14B8   EXCLUSIVE   WAIT
SYSB      HSM      0040      0099F038   SHARE       WAIT
SYSB      HSM      0040      009E26C0   SHARE       WAIT
SYSB      HSM      0040      009AF038   SHARE       WAIT
SYSB      HSM      0040      009B9A38   SHARE       WAIT
SYSB      HSM      0040      0099A1F8   SHARE       WAIT
SYSB      HSM      0040      009B1738   SHARE       WAIT
SYSB      HSM      0040      009B03C8   SHARE       WAIT
SYSB      HSM      0040      009B0968   SHARE       WAIT
SYSB      HSM      0040      009B71D8   SHARE       WAIT
SYSB      HSM      0040      00998480   SHARE       WAIT
SYSB      HSM      0040      009982E8   SHARE       WAIT
SYSB      HSM      0040      0099AC68   SHARE       WAIT
SYSB      HSM      0040      0099AA48   SHARE       WAIT
SYSB      HSM      0040      0099A8B0   SHARE       WAIT
SYSB      HSM      0040      009B0E88   SHARE       WAIT
```

TCBTERM requests the 4-digit hexadecimal ASID number of the address space (listed in the ASID column above), and then requests the 8-digit hexadecimal TCB address (listed in the TCBADDR column above). The particular TCB that needs to be terminated in the display above is the one that says EXCLUSIVE in the EXC/SHR column, whose TCB address is 009B14B8. This exclusive ENQ is causing the other shared requests for the resource to wait.

Please note that this is a specific incident and may not be appropriate in all circumstances. Terminating a task could have other ramifications

for the address space. This method should only be used if cancelling or forcing the address space out of the system is not an option.

TCBTERM can also be used to terminate a task that only has a single exclusive ENQ which might be causing other address spaces to wait for the availability of the ENQed resource. It is unfortunate that IBM does not provide a simple facility to issue a DEQ macro against the resource being held to alleviate this kind of condition. The only way currently available is to schedule an SRB into the offending address space in order to issue a DEQ from within the address space directed at the TCB under which the ENQ was issued; this method is called directed DEQ. While it is feasible to write such a mechanism, it is considerably more complex that just terminating the TCB under which the ENQ was issued. This directed DEQ would not be without its risks also since after the directed DEQ was done, the application might still abend if were to issue its own DEQ of the resource later on.

JCL TO RUN TCBTERM PROGRAM

```
//TCBTERM  PROC
//TCBTERM  EXEC PGM=TCBTERM
//SYSABEND DD SYSOUT=*
```

TCBTERM

```
TCBTERM  CSECT
         YREGS
         SAVE  (14,12),,TCBTERM-&SYSDATE
         LR    R12,R15
         USING TCBTERM,R12
         LA    R8,SAVE
         ST    R8,8(,R13)
         ST    R13,4(,R8)
         LR    R13,R8
         LA    R13,SAVE1
GETASID  WTOR  'TCB001A ENTER FOUR BYTE HEX ASID NUMBER TO TERMINATE', X
               REPLY,4,REPECB,ROUTCDE=(1,2,11)  GET ASID
         WAIT  ECB=REPECB         WAIT FOR REPLY
         TRT   REPLY(4),TRANTAB1  IS REPLY VALID HEX DIGITS
         BZ    ASIDOK             YES, GO TO OK
         WTO   'TCB002I INVALID HEX ASID',ROUTCDE=(1,2,11)
         MVC   REPLY,BLANKS       ELSE, CLEAR REPLY FIELD
         XC    REPECB,REPECB            CLEAR ECB
         B     GETASID            REISSUE MESSAGE
ASIDOK   TR    REPLY(4),TRANTAB2  TRANSLATE TO HEX
         PACK  PACKFLD(3),REPLY(5) REMOVE ZONES
         MVC   ASIDNUM+2(2),PACKFLD SAVE ASID
```

```
         L     R7,ASIDNUM
         MVC   REPLY,BLANKS       CLEAR REPLY FIELD
         XC    REPECB,REPECB      CLEAR ECB
GETTCBA  WTOR  'TCB003A ENTER EIGHT BYTE HEX TCB ADDRESS TO TERMINATE',X
               REPLY,8,REPECB,ROUTCDE=(1,2,11)  GET TCB ADDRESS
         WAIT  ECB=REPECB         WAIT FOR REPLY
         TRT   REPLY(8),TRANTAB1  IS REPLY VALID HEX DIGITS
         BZ    TCBAOK             YES, GO TO OK
         WTO   'TCB004I INVALID HEX TCB ADDRESS',ROUTCDE=(1,2,11)
         MVC   REPLY,BLANKS       ELSE, CLEAR REPLY FIELD
         XC    REPECB,REPECB            CLEAR ECB
         B     GETTCBA            REISSUE MESSAGE
TCBAOK   TR    REPLY(8),TRANTAB2  TRANSLATE TO HEX
         PACK  PACKFLD+0(3),REPLY+0(5) REMOVE ZONES
         PACK  PACKFLD+2(3),REPLY+4(5) REMOVE ZONES
         MVC   TCBADDR,PACKFLD    SAVE TCB ADDRESS
         L     R6,TCBADDR
         MODESET KEY=ZERO,MODE=SUP
         CALLRTM TYPE=ABTERM,COMPCOD=X'FFF',TCB=(R6),ASID=(R7),DUMP=YES
         WTO   'TCB999I ABTERM ISSUED',ROUTCDE=(1,2,11)
         LA    R13,SAVE
         L     R13,SAVE+4
         RETURN (14,12),RC=0
SAVE     DS    18F
SAVE1    DS    18F
DECIMAL  DC    D'0'               AREA TO PACK DECIMAL ASID
ASIDNUM  DC    F'0'
REPECB   DC    F'0'               ECB TO WAIT ON FOR WTOR
EYE1     DC    C'TCBA'
TCBADDR  DC    F'0'
EYE2     DC    C'ASID'
PACKFLD  DS    CL9
BLANKS   DC    CL8' '             TO BLANK OUT REPLY FIELD
REPLY    DC    CL8' '             REPLY AREA FOR WTOR
TRANTAB1 DC    256X'FF'
         ORG   TRANTAB1+C'A'
         DC    X'000000000000'          C'ABCDEF'
         ORG   TRANTAB1+C'0'
         DC    X'00000000000000000000'  C'0123456789'
         ORG
TRANTAB2 DC    256X'00'
         ORG   TRANTAB2+C'A'
         DC    X'0A0B0C0D0E0F'          C'ABCDEF'
         ORG   TRANTAB2+C'0'
         DC    X'000102030405060708090'  C'0123456789'
         ORG
         DS   0F
         CVT DSECT=YES
         END
```

Processor configuration and IPL information (part 2)

This month we complete our look at a utility that displays processor information and IPL information. The following REXX EXEC will display information contained in the SCCB, PCCA, and IPA.

DSCPINFO REXX EXEC

```
/*********************************REXX*********************************/
/*                                                                    */
/*  Program-id           DSCPINFO                                     */
/*  Remarks              This routine will display information        */
/*                       contained in the SCCB, PCCA and IPA.         */
/*trace i*/
/*....................................................................*/
/* Driver section                                                    .*/
/*....................................................................*/
/*trace i*/                          /* Turn tracing on             */
call init_vars                       /* Call scp info module        */
call scp_info_module                 /* Call scp info module        */
call Allocation_section              /* Allocation section          */
call display_option_panel            /* Display the option panel*/
call Deallocate_section              /* Deallocation section        */
exit(0)                              /* Return to caller            */
/*********************************************************************/
/* Call module scpinfo to obtain all the CPU information             */
/*********************************************************************/
scp_info_module:
scpinfo                              /* Call scpinfo module         */
return                               /* Return to caller            */
display_option_panel:
do forever
   Panel= 'scppan01'                 /* Panel name                  */
   ADDRESS "ISPEXEC" "ADDPOP ROW(2) COLUMN(2)" /* pop-up position  */
   call display_the_panel            /* Display options panel       */
   If (panel_rc = 8) Then Do         /* Return?                     */
      ADDRESS "ISPEXEC" "REMPOP"     /* Remove pop-up               */
      Leave                          /* Yes- return to caller       */
   End
   select
      when (spcfg= '/') then do      /* Display proc config pan?*/
           call display_pcfg_info    /* Yes-                        */
      end
      when (scpu= '/') then do       /* Display CPU panel?          */
           call display_cpu_info     /* Yes-                        */
      end
```

```
      when (scpuvc= '/') then do        /* Display CPU vector and  */
                                        /* Crypto panel?           */
           call display_cpu_vector_crypto_info /* Yes-             */
      end
      when (shsa = '/') then do         /* Display HSA panel?      */
           call display_hsa_info        /* Yes-                    */
      end
      when (sipl = '/') then do         /* Display IPL panel?      */
           call display_ipl_info        /* Yes-                    */
      end
      when (sysi = '/') then do         /* Display IEASYSnn panel? */
           call display_ieasys_info     /* Yes-                    */
      end
      otherwise
           ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ1)"
      end
   ADDRESS "ISPEXEC" "REMPOP"           /* Remove pop-up           */
end
Return                                  /* Return to caller        */
/*.............................................................*/
/* Display panel section                                      .*/
/*.............................................................*/
Display_the_panel:
ADDRESS "ISPEXEC" "DISPLAY PANEL("Panel")"   /* Display menu panel */
if (rc > 8) then do                      /* Error?                 */
   say 'Display error    rc = 'rc''      /* Yes- output message    */
   call deallocate_section               /* Remove allocation      */
   exit(Ø)                               /* And quit               */
end
Panel_rc= rc                            /* Save the rc             */
return
/*.............................................................*/
/* Build MVS level information                                .*/
/*.............................................................*/
MVS_Level_info:
prodo=    Substr(mvslevel,1,16)         /* Product owner           */
prodn=    Substr(mvslevel,17,16)        /* Product name            */
mvsfmid=  Substr(mvslevel,47,8)         /* MVS FMID                */
cpumodel= Substr(cpuv,9,4)              /* CPU model               */
prodv=    'V'||Substr(mvslevel,33,2)    /* MVS version             */
prodv=    prodv||'R'||Substr(mvslevel,35,2) /* MVS release         */
prodv=    prodv||'M'||Substr(mvslevel,37,2) /* MVS modification    */
msplevel= Substr(mvslevel,39,8)         /* MVS SP level            */
call      calc_ipl_date                 /* Calculate the IPL date  */
call      calc_ipl_time                 /* Calculate the IPL time  */
iplvol=   Substr(mvslevel,63,6)         /* IPL volser              */
Return                                  /* Return to caller        */
/*.............................................................*/
/* Display the configuration information                      .*/
/*.............................................................*/
display_pcfg_info:
call MVS_Level_info                     /* Build MVS level info    */
```

```
sar=      x2d(c2x(Substr(cscp,1,2)))    /* Real stor address range */
                                        /* maximum storage incr no */
                                        /* installed               */
sar=      sar||'M'                      /* Real storage in Megs    */
ncps=     x2d(c2x(Substr(cscp,7,2)))    /* Number of CPUs installed*/
sai=      x2d(c2x(Substr(cscp,3,1)))    /* Real storage address    */
sai=      sai||'M'                      /* Increment in units of 1M*/
                                        /* installed               */
nhsa=     x2d(c2x(Substr(cscp,9,2)))    /* Number of HSAs          */
lparm=    Substr(cscp,11,8)             /* Load Parm               */
sbs=      x2d(c2x(Substr(cscp,4,1)))    /* Real storage block size */
sbs=      sbs||'K'                      /* in units of 1K          */
sii=      x2d(c2x(Substr(cscp,5,2)))    /* Real storage increment  */
                                        /* block interleave        */
                                        /* interval                */
vss=      x2d(c2x(Substr(cscp,29,2)))   /* Vector section size     */
vpsm=     x2d(c2x(Substr(cscp,31,2)))   /* Vector partial sum      */
mesi=     x2d(c2x(Substr(cscp,19,4)))   /* Extended storage range  */
mesi=     mesi||'M'                     /* in Megs                 */
nxsb=     x2d(c2x(Substr(cscp,23,4)))   /* No of 4k storage blocks */
                                        /* in an extended storage  */
mese=     x2d(c2x(Substr(cscp,27,2)))   /* Maximun extended storage*/
                                        /* element number          */
                                        /* increment               */
ifm1= Substr(cscp,33,1)                 /* Installed facility map  */
                                        /* Byte 1                  */
chpi= 'N'                               /* Default                 */
If (testbit(ifm1,'80'x)) = 1 Then       /* Channel path info?      */
   chpi= 'Y'                            /* Yes-                    */
chps= 'N'                               /* Default                 */
If (testbit(ifm1,'40'x)) = 1 Then       /* Channel path subsystem  */
                                        /* Command?                */
   chps= 'Y'                            /* Yes-                    */
chpr= 'N'                               /* Default                 */
If (testbit(ifm1,'20'x)) = 1 Then       /* Channel path reconfig?  */
                                        /* Command?                */
   chpr= 'Y'                            /* Yes-                    */
cpui= 'N'                               /* Default                 */
If (testbit(ifm1,'08'x)) = 1 Then       /* CPU information?        */
                                        /* Command?                */
   cpui= 'Y'                            /* Yes-                    */
cpur= 'N'                               /* Default                 */
If (testbit(ifm1,'04'x)) = 1 Then       /* CPU reconfiguration?    */
   cpur= 'Y'                            /* Yes-                    */
ifm2= Substr(cscp,34,1)                 /* Installed facility map  */
                                        /* Byte 2                  */
sgnl= 'N'                               /* Default                 */
If (testbit(ifm2,'80'x)) = 1 Then       /* Signal alarm?           */
   sgnl= 'Y'                            /* Yes-                    */
omr=  'N'                               /* Default                 */
If (testbit(ifm2,'40'x)) = 1 Then       /* Write operator message  */
                                        /* and read operator resp? */
```

```
   omr=  'Y'                          /* Yes-                     */
stst= 'N'                             /* Default                  */
If (testbit(ifm2,'20'x)) = 1 Then     /* Store status on load?    */
   stst= 'Y'                          /* Yes-                     */
rstr= 'N'                             /* Default                  */
If (testbit(ifm2,'10'x)) = 1 Then     /* Restart reasons?         */
   rstr= 'Y'                          /* Yes-                     */
itrc= 'N'                             /* Default                  */
If (testbit(ifm2,'08'x)) = 1 Then     /* Instruction address      */
                                      /* Trace?                   */
   itrc= 'Y'                          /* Yes-                     */
lprm= 'N'                             /* Default                  */
If (testbit(ifm2,'04'x)) = 1 Then     /* Load parameter?          */
   lprm= 'Y'                          /* Yes-                     */
wdat= 'N'                             /* Default                  */
If (testbit(ifm2,'02'x)) = 1 Then     /* Read and write facility?*/
   wdat= 'Y'                          /* Yes-                     */
ifm3= Substr(cscp,35,1)               /* Installed facility map   */
                                      /* Byte 3                   */
sir=  'N'                             /* Default                  */
If (testbit(ifm3,'80'x)) = 1 Then     /* Real storage incrememt   */
                                      /* Reconfig?                */
   sir=  'Y'                          /* Yes-                     */
sei=  'N'                             /* Default                  */
If (testbit(ifm3,'40'x)) = 1 Then     /* Real storage element     */
                                      /* information?             */
   sei=  'Y'                          /* Yes-                     */
ser=  'N'                             /* Default                  */
If (testbit(ifm3,'20'x)) = 1 Then     /* Real storage element     */
                                      /* reconfiguration?         */
   ser=  'Y'                          /* Yes-                     */
cars= 'N'                             /* Default                  */
If (testbit(ifm3,'10'x)) = 1 Then     /* Copy and re-assign?      */
   cars= 'Y'                          /* Yes-                     */
carl= 'N'                             /* Default                  */
If (testbit(ifm3,'01'x)) = 1 Then     /* Copy and re-assign?      */
                                      /* Storage list?            */
   carl= 'Y'                          /* Yes-                     */
sum=  'N'                             /* Default                  */
If (testbit(ifm3,'08'x)) = 1 Then     /* Extended storage         */
                                      /* useability map?          */
   sum=  'Y'                          /* Yes-                     */
esei= 'N'                             /* Default                  */
If (testbit(ifm3,'04'x)) = 1 Then     /* Extended storage element*/
                                      /* information?             */
   esei= 'Y'                          /* Yes-                     */
eser= 'N'                             /* Default                  */
If (testbit(ifm3,'02'x)) = 1 Then     /* Extended storage element*/
                                      /* ireconfiguration?        */
   eser= 'Y'                          /* Yes-                     */
ifm4= Substr(cscp,36,1)               /* Installed facility map   */
                                      /* Byte 4                   */
```

```
vfr=  'N'                                /* Default                 */
If (testbit(ifm4,'80'x)) = 1 Then        /* Vector feature reconfig?*/
   vfr=  'Y'                             /* Yes-                    */
evnt= 'N'                                /* Default                 */
If (testbit(ifm4,'40'x)) = 1 Then        /* Read/write event?       */
   evnt= 'Y'                             /* Yes-                    */
rrgi= 'N'                                /* Default                 */
If (testbit(ifm4,'08'x)) = 1 Then        /* Read resource group?    */
   rrgi= 'Y'                             /* Yes-                    */
con1= Substr(cscp,37,1)                  /* Configuration           */
                                         /* Characteristics?        */
sopf= 'N'                                /* Default                 */
If (testbit(con1,'20'x)) = 1 Then        /* Suppression on          */
                                         /* protection?             */
   sopf= 'Y'                             /* Yes-                    */
irin= 'N'                                /* Default                 */
If (testbit(con1,'10'x)) = 1 Then        /* Initiate reset?         */
   irin= 'Y'                             /* Yes-                    */
cscf= 'N'                                /* Default                 */
If (testbit(con1,'08'x)) = 1 Then        /* Store subchannel        */
                                         /* Subsystem characterists?*/
   cscf= 'Y'                             /* Yes-                    */
display_Panel = 'y'                      /* Display the panel       */
Do While (display_Panel = 'y')           /* Do while display_panel  */
                                         /* = 'y'                   */
   panel= 'scppan02'                     /* Panel name              */
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position  */
   call display_the_panel                /* Display options panel   */
   If (panel_rc = 8) Then Do             /* Return?                 */
      ADDRESS "ISPEXEC" "REMPOP"         /* Remove pop-up           */
      display_Panel = 'n'                /* Quit the do loop        */
      Iterate                            /* Round  again            */
   End
   If (dcecf = 'n' ≥ dcecf = 'N') Then do /* Display CEC FAC?      */
      ADDRESS "ISPEXEC" "REMPOP"         /* Remove pop-up           */
      display_Panel = 'n'                /* Quit the do loop        */
      Iterate                            /* Round again             */
   End
   Panel= 'scppan03'                     /* Panel name              */
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* pop-up position  */
   call display_the_panel                /* Display options panel   */
   ADDRESS "ISPEXEC" "REMPOP"            /* Remove pop-up           */
   ADDRESS "ISPEXEC" "REMPOP"            /* Remove pop-up           */
End
return                                   /* Return to caller        */
/*.............................................................*/
/* Display the CPU information                                .*/
/*.............................................................*/
display_cpu_info:
ADDRESS "ISPEXEC" "TBCREATE CPUTAB NOWRITE REPLACE"
if (rc >  4) then do                     /* Call OK?                */
   say 'TBCREATE error    rc = 'rc''     /* No-inform the user      */
```

```
   call deallocate_section              /* Remove allocation       */
   exit(0)                              /* Lets quit               */
end
ztdmark= ''                             /* No information          */
ADDRESS  "ISPEXEC" "VPUT  (ztdmark) SHARED"
tabrows=  16                            /* Max table rows          */
comperr= ''                             /* Initialize              */
ncpu=     x2d(c2x(Substr(cscp,7,2)))    /* Number of CPUs          */
If (ncpu = 0) then do
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIM002)" /* Issue message       */
   return                               /* Return to caller        */
end
x=1                                     /* Starting position       */
Do i= 1 to ncpu                         /* Do no of table entries  */
   pcpid= substr(cpuv,x,12)             /* cpuid                   */
   x= (x + 12)                          /* Next field              */
   pcpua=  c2x(substr(cpuv,x,2))        /* CPU address             */
   x= (x + 2)                           /* Next field              */
   ppsav=  c2x(substr(cpuv,x,4))        /* Virtual PSA address     */
   x= (x + 4)                           /* Next field              */
   ppsar=  c2x(substr(cpuv,x,4))        /* Real PSA address        */
   x= (x + 4)                           /* Next field              */
   piscm=  c2x(substr(cpuv,x,1))        /* ISC                     */
   x= (x + 2)                           /* Next field              */
   stod=   c2x(substr(cpuv,x,1))        /* TOD number              */
   x= (x + 2)                           /* Next field              */
   cpf2=   Substr(cpuv,x,1)             /* CPU characteristics flag*/
   pspace= 'N'                          /* Default                 */
   If (testbit(cpf2,'80'x)) = 1 Then    /* Private space?          */
      pspace= 'Y'                       /* Yes-                    */
   per2= 'N'                            /* Default                 */
   If (testbit(cpf2,'01'x)) = 1 Then    /* PER?                    */
      per2= 'Y'                         /* Yes-                    */
   x= (x + 1)                           /* Next entry              */
   ADDRESS "ISPEXEC" "TBADD CPUTAB
            SAVE(pcpid,pcpua,ppsav,ppsar,piscm,stod,pspace,per2)
            MULT("TABROWS")"
   if (rc ¬= 0) then do                 /* Call OK?                */
      comperr= 'CPU'                    /* Component error         */
      ADDRESS "ISPEXEC" "SETMSG MSG(SCPIM003)" /* Issue message    */
      leave                             /* Lets quit               */
   end
end
If (comperr ¬= '') Then                 /* Error?                  */
   Return                               /* Yes- return to caller   */
ADDRESS "ISPEXEC" "TBTOP CPUTAB"        /* Position to top of CPU  */
if (rc ¬= 0) then do                    /* Call OK?                */
   comperr= 'CPU'                       /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIM003)" /* Issue message       */
   return                               /* Lets quit               */
end
```

```
ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position     */
ADDRESS "ISPEXEC" "TBDISPL CPUTAB PANEL(SCPPANØ4)"
if (rc > 8) then do                       /* Error?                */
   comperr= 'CPU'                         /* Component error       */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"             /* Remove pop-up         */
   return                                 /* Lets quit             */
end

ADDRESS "ISPEXEC" "TBCLOSE  CPUTAB"       /* Close the table       */
if (rc ¬= Ø) then do                      /* Call OK?              */
   comperr= 'CPU'                         /* Component error       */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"             /* Remove pop-up         */
   return                                 /* Lets quit             */
end
ADDRESS "ISPEXEC" "REMPOP"                /* Remove pop-up         */
return
/*.............................................................*/
/* Display CPU vector and crypto information                  .*/
/*.............................................................*/
display_cpu_vector_crypto_info:
ADDRESS "ISPEXEC" "TBCREATE CPUVCTAB NOWRITE REPLACE"
if (rc >  4) then do                      /* Call OK?              */
   say 'TBCREATE error    rc = 'rc''      /* No- inform the user   */
   call deallocate_section                /* Remove allocation     */
   exit(Ø)                                /* Lets quit             */
end
ztdmark= ''                               /* No information        */
ADDRESS  "ISPEXEC" "VPUT  (ztdmark) SHARED"
tabrows=  16                              /* Max table rows        */
comperr= ''                               /* Initialize            */
ncpu=     x2d(c2x(Substr(cscp,7,2)))      /* Number of CPUs        */
If (ncpu = Ø) then do
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ2)" /* Issue message       */
   return                                 /* Return to caller      */
end
x=1                                       /* Starting position     */
Do i= 1 to ncpu                           /* Do no of table entries*/
   pcpid= substr(cpuv,x,12)               /* cpuid                 */
   x= (x + 12)                            /* Next field            */
   pcpua=  c2x(substr(cpuv,x,2))          /* CPU address           */
   x= (x + 13)                            /* Next field            */
   cpf1=   Substr(cpuv,x,1)               /* CPU characteristics flag*/
   pvfi=   'N'                            /* Default               */
   If (testbit(cpf1,'8Ø'x)) = 1 Then      /* Vector feature installed*/
      pvfi= 'Y'                           /* Yes-                  */
   pvfc=   'N'                            /* Default               */
   If (testbit(cpf1,'4Ø'x)) = 1 Then      /* Vector feature connected*/
      pvfc= 'Y'                           /* Yes-                  */
   pvfss=  'N'                            /* Default               */
   If (testbit(cpf1,'2Ø'x)) = 1 Then      /* Vector feature in     */
```

```
                                    /* Standby mode?           */
      pvfss= 'Y'                    /* Yes-                    */
   pcfi=   'N'                      /* Default                 */
   If (testbit(cpf1,'1Ø'x)) = 1 Then /* Crypto feature installed*/
      pcfi= 'Y'                     /* Yes-                    */
   x= (x + 2)                       /* Next entry              */
   ADDRESS "ISPEXEC" "TBADD CPUVCTAB
            SAVE(pcpid,pcpua,pvfi,pvfc,pvfss,pcfi)
            MULT("TABROWS")"
   if (rc ¬= Ø) then do             /* Call OK?                */
      comperr= 'CPUVC'              /* Component error         */
      ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message    */
      leave                         /* Lets quit               */
   end
end
If (comperr ¬= '') Then             /* Error?                  */
   Return                           /* Yes- return to caller   */

ADDRESS "ISPEXEC" "TBTOP CPUVCTAB"  /* Position to top of CPU  */
if (rc ¬= Ø) then do                /* Call OK?                */
   comperr= 'CPUVC'                 /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   return                           /* Lets quit               */
end
ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position     */
ADDRESS "ISPEXEC" "TBDISPL CPUVCTAB PANEL(SCPPANØ9)"
if (rc > 8) then do                 /* Error?                  */
   comperr= 'CPUVC'                 /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"       /* Remove pop-up           */
   return                           /* Lets quit               */
end
ADDRESS "ISPEXEC" "TBCLOSE  CPUVCTAB"   /* Close the table         */
if (rc ¬= Ø) then do                /* Call OK?                */
   comperr= 'CPUVC'                 /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"       /* Remove pop-up           */
   return                           /* Lets quit               */
end
ADDRESS "ISPEXEC" "REMPOP"          /* Remove pop-up           */
return
/*.............................................................*/
/* Display the HSA information                                .*/
/*.............................................................*/
display_hsa_info:
ADDRESS "ISPEXEC" "TBCREATE HSATAB NOWRITE REPLACE"
if (rc >  4) then do                /* Call OK?                */
   say 'TBCREATE error    rc = 'rc''    /* No- inform the user     */
   call deallocate_section          /* Remove Allocation       */
   exit(Ø)                          /* Lets quit               */
end
```

```
ztdmark= ''                              /* No information          */
ADDRESS  "ISPEXEC" "VPUT  (ztdmark) SHARED"
tabrows=  16                             /* Max table rows          */
comperr= ''                              /* Initialize              */
nhsa=     x2d(c2x(Substr(cscp,9,2)))     /* Number of HSAs          */
If (nhsa = Ø) then do
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ2)" /* Issue message       */
   return                                /* Return to caller        */
end
x=1                                      /* Starting position       */
Do i= 1 to nhsa                          /* Do no of table entries  */
   hssz=  x2d(c2x(substr(hsav,x,2)))     /* HSA size                */
   hssz=  (hssz * 4)||'K'                /* In Kilobytes            */
   x= (x + 2)                            /* Res-position            */
   hsaa=  c2x(substr(hsav,x,4))          /* HSA size                */
   ADDRESS "ISPEXEC" "TBADD HSATAB SAVE(hssz,hsaa) MULT("TABROWS")"
   if (rc ¬= Ø) then do                  /* Call OK?                */
      comperr= 'HSA'                     /* Component error         */
      ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message    */
      leave                              /* Lets quit               */
   end
   x= (x + 4)                            /* Next table entry        */
end
If (comperr ¬= '') Then                  /* Error?                  */
   Return                                /* Yes- return to caller   */
ADDRESS "ISPEXEC" "TBTOP HSATAB"         /* Position to top of tab  */
if (rc ¬= Ø) then do                     /* Call OK?                */
   comperr= 'HSA'                        /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   return                                /* Let's quit              */
end
ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position     */
ADDRESS "ISPEXEC" "TBDISPL HSATAB PANEL(SCPPANØ5)"
if (rc > 8) then do                      /* Error?                  */
   comperr= 'HSA'                        /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"            /* Remove pop-up           */
   return                                /* Lets quit               */
end
ADDRESS "ISPEXEC" "TBCLOSE  HSATAB"      /* Close the table         */
if (rc ¬= Ø) then do                     /* Call OK?                */
   comperr= 'HSA'                        /* Component error         */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"            /* Remove pop-up           */
   return                                /* Lets quit               */
end
ADDRESS "ISPEXEC" "REMPOP"               /* Remove pop-up           */
return
/*.............................................................*/
/* Display the IPL information                                 .*/
/*.............................................................*/
```

```
display_ipl_info:
call MVS_Level_info                      /* Build MVS level info    */
lparm=    substr(iplv,1,8)               /* Loadparm                */
iodfdsn=  substr(iplv,9,4)||'.IODF'      /* iodf hlq.IODF           */
iodfdsn=  iodfdsn||substr(iplv,17,2)     /* iodf hlq.IODF.suff      */
nucid=    substr(iplv,8,1)               /* Nucleus ID              */
iocfg=    substr(iplv,19,8)              /* I/O configuration ID    */
ioedt=    substr(iplv,27,2)              /* EDT ID                  */
nuclstid= substr(iplv,29,2)              /* Nucleus ID              */
sysparm=  substr(iplv,31,63)             /* IEASYSxx suffixes       */
ieasym=   substr(iplv,94,63)             /* IEASYMxx suffixes       */
sysplex=  substr(iplv,145,8)             /* Sysplex name            */
lparname= substr(iplv,165,8)             /* LPARNAME                */
hwname=   substr(iplv,173,8)             /* HWNAME                  */
mcatd=    substr(iplv,181,25)            /* Master catalog DSN      */
mcatv=    substr(iplv,225,6)             /* Master catalog DSN      */
mcatt= 'ICF + SYS%-SYS1'                 /* Default                 */
If (substr(iplv,231,1) =  ' ') Then      /* VSAM catalog?           */
   mcatt= 'VSAM'                         /* Yes                     */
Else
If (substr(iplv,231,1) =  '1') Then      /* ICF catalog?            */
   mcatt= 'ICF'                          /* Yes                     */
mcatal=   substr(iplv,232,1)             /* Alias level             */
ccas=     substr(iplv,233,2)             /* CAS service task lower  */
                                         /* Limit                   */
plibdsn=  substr(iplv,235,44)            /* IPL PARMLIB DSN         */
plibddv=  substr(iplv,279,4)             /* IPL PARMLIB device no   */
parmdsn=  substr(iplv,284,44)            /* PARMLIB DSN             */
parmvol=  substr(iplv,328,6)             /* PARMLIB volser          */
iodds=    substr(iplv,283,1)             /* Default                 */
If (substr(iplv,283,1) =  ' ') Then      /* Load device support mods*/
   iodds= 'Y'                            /* Yes                     */
puflag1= Substr(iplv,334,1)              /* PARMLIB usage flag      */
parmiuse= 'N'                            /* Default                 */
If (testbit(puflag1,'80'x)) = 1 Then     /* PARMLIB in use?         */
   parmiuse= 'Y'                         /* Yes-                    */
parmdef=  'N'                            /* Default                 */
If (testbit(puflag1,'40'x)) = 1 Then     /* Default parmlib?        */
   parmdef= 'Y'                          /* Yes-                    */
parmcat=  'N'                            /* Default                 */
If (testbit(puflag1,'20'x)) = 1 Then     /* PARMLIB from catalog?   */
   parmcat= 'Y'                          /* Yes-                    */
parmloc=  'N'                            /* Default                 */
If (testbit(puflag1,'08'x)) = 1 Then     /* PARMLIB locate failed?  */
   parmloc= 'Y'                          /* Yes-                    */
parmmnt=  'N'                            /* Default                 */
If (testbit(puflag1,'04'x)) = 1 Then     /* PARMLIB mount failed?   */
   parmmnt= 'Y'                          /* Yes-                    */
parmopen= 'N'                            /* Default                 */
If (testbit(puflag1,'02'x)) = 1 Then     /* PARMLIB open failed?    */
   parmopen= 'Y'                         /* Yes-                    */
display_Panel = 'y'                      /* Do while display_panel  */
Do While (display_Panel = 'y')           /* Do while display_panel  */
```

```
                                    /* = 'y'                    */
   panel= 'scppanØ6'                /* Panel name               */
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position  */
   call display_the_panel           /* Display options panel    */
   If (panel_rc = 8) Then Do        /* Return?                  */
      ADDRESS "ISPEXEC" "REMPOP"    /* Remove pop-up            */
      display_Panel = 'n'           /* Quit the do loop         */
      Iterate                       /* Round again              */
   End
   If (pusage ¬= '/' ) Then do      /* Display PARMLIB usage?   */
      ADDRESS "ISPEXEC" "REMPOP"    /* Remove pop-up            */
      display_Panel = 'n'           /* Quit the do loop         */
      Iterate                       /* Round again              */
   End
   Panel= 'scppanØ7'                /* Panel name               */
   ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position  */
   call display_the_panel           /* Display options panel    */
   ADDRESS "ISPEXEC" "REMPOP"       /* Remove pop-up            */
   ADDRESS "ISPEXEC" "REMPOP"       /* Remove pop-up            */
End
return
/*.............................................................*/
/* Display IEASYSnn IPL information                           .*/
/*.............................................................*/
display_ieasys_info:
ADDRESS "ISPEXEC" "TBCREATE IEASYSTB NOWRITE REPLACE"
if (rc >  4) then do                /* Call OK?                 */
   say 'TBCREATE error    rc = 'rc''  /* No- inform the user    */
   call deallocate_section          /* Remove allocation        */
   exit(Ø)                          /* Lets quit                */
end
ztdmark= ''                         /* No information           */
ADDRESS  "ISPEXEC" "VPUT  (ztdmark) SHARED"
tabrows=  5ØØ                       /* Max table rows           */
comperr= ''                         /* Initialize               */

If (vclk ¬= '') Then Do             /* CLOCK?                   */
   sysparm= 'CLOCK'                 /* Process CLOCKxx          */
   sysvar=  vclk                    /* Switch                   */
   call build_ieasys_table_entry    /* Build table entry        */
End
If (vcmd ¬= '') Then Do             /* IEACMDxx present?        */
   sysparm= 'CMD'                   /* Process IEACMDxx         */
   sysvar=  vcmd                    /* Switch                   */
   call build_ieasys_table_entry    /* Build table entry        */
End
If (vcon ¬= '') Then Do             /* CONSOLExx?               */
   sysparm= 'CON'                   /* Process CONSOLExx        */
   sysvar=  vcon                    /* Switch                   */
   call build_ieasys_table_entry    /* Build table entry        */
End
If (vcsa ¬= '') Then Do             /* CSA?                     */
```

```
   sysparm= 'CSA'                       /* Process CSA             */
   sysvar=  vcsa                        /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vlogr ¬= '') Then Do                /* LOGREC?                 */
   sysparm= 'LOGREC'                    /* Process LOGREC          */
   sysvar=  vlogr                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vlnk ¬= '') Then Do                 /* LNKLSTxx?               */
   sysparm= 'LNKLST'                    /* Process LNKLSTxx        */
   sysvar=  vlnk                        /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vlnka ¬= '') Then Do                /* LNKAUTH?                */
   sysparm= 'LNKAUTH'                   /* Process LNKAUTH         */
   sysvar=  vlnka                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vlpa ¬= '') Then Do                 /* LPALSTxx?               */
   sysparm= 'LPALST'                    /* Process LPALSTxx        */
   sysvar=  vlpa                        /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vmlpa ¬= '') Then Do                /* IEALPAxx?               */
   sysparm= 'IEALPA'                    /* Process IEALPAxx        */
   sysvar=  vmlpa                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vmstr ¬= '') Then Do                /* MSTJCLxx?               */
   sysparm= 'MSTJCL'                    /* Process MSTJCLxx        */
   sysvar=  vmstr                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vpagp ¬= '') Then Do                /* PAGE ieasysxx?          */
   sysparm= 'PAGE(S)'                   /* Process page            */
   sysvar=  vpagp                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vpago ¬= '') Then Do                /* PAGE operator override? */
   sysparm= 'PAGE(O)'                   /* Process page            */
   sysvar=  vpago                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vpagt ¬= '') Then Do                /* PAGTOTL?                */
   sysparm= 'PAGTOTL'                   /* Process PAGTOTL         */
   sysvar=  vpagt                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
End
If (vprog ¬= '') Then Do                /* PROG?                   */
   sysparm= 'PROG'                      /* Process PROG            */
   sysvar=  vprog                       /* Switch                  */
   call build_ieasys_table_entry        /* Build table entry       */
```

```
End
If (vsch ¬= '') Then Do                 /* SCH?                      */
   sysparm= 'SCH'                       /* Process SCH               */
   sysvar=  vsch                        /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vsmf ¬= '') Then Do                 /* SMF?                      */
   sysparm= 'SMF'                       /* Process SMF               */
   sysvar=  vsmf                        /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vssn ¬= '') Then Do                 /* SSN?                      */
   sysparm= 'SSN'                       /* Process SSN               */
   sysvar=  vssn                        /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vsqa ¬= '') Then Do                 /* SQA?                      */
   sysparm= 'SQA'                       /* Process SQA               */
   sysvar=  vsqa                        /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vsvc ¬= '') Then Do                 /* SVC?                      */
   sysparm= 'SVC'                       /* Process SVC               */
   sysvar=  vsvc                        /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vsysp ¬= '') Then Do                /* SYSP?                     */
   sysparm= 'SYSP'                      /* Process SYSP              */
   sysvar=  vsysp                       /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
If (vviod ¬= '') Then Do                /* VIODSN?                   */
   sysparm= 'VIODSN'                    /* Process VIODSN            */
   sysvar=  vviod                       /* Switch                    */
   call build_ieasys_table_entry        /* Build table entry         */
End
ADDRESS "ISPEXEC" "TBTOP IEASYSTB"      /* Position to top of tab    */
if (rc ¬= Ø) then do                    /* Call OK?                  */
   comperr= 'IEASYS'                    /* Component error           */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message        */
   return                               /* Lets quit                 */
end
ADDRESS "ISPEXEC" "ADDPOP ROW(1) COLUMN(2)" /* Pop-up position       */
ADDRESS "ISPEXEC" "TBDISPL IEASYSTB PANEL(SCPPANØ8)"
if (rc > 8) then do                     /* Error?                    */
   comperr= 'IEASYS'                    /* Component error           */
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message        */
   ADDRESS "ISPEXEC" "REMPOP"           /* Remove pop-up             */
   return                               /* Lets quit                 */
end
ADDRESS "ISPEXEC" "TBCLOSE  IEASYSTB"   /* Close the table           */
if (rc ¬= Ø) then do                    /* Call OK?                  */
   comperr= 'IEASYS'                    /* Component error           */
```

```
   ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message       */
   ADDRESS "ISPEXEC" "REMPOP"           /* Remove pop-up           */
   return                               /* Lets quit               */
end
ADDRESS "ISPEXEC" "REMPOP"              /* Remove pop-up           */
Return                                  /* Return to caller        */
/*.............................................................*/
/* Display IEASYSnn IPL information                           .*/
/*.............................................................*/
build_ieasys_table_entry:
i= 5                                    /* Starting position       */
len=   x2d(c2x(substr(sysvar,1,2)))     /* Parm string length      */
ieasysxx= Substr(sysvar,3,2)            /* Source of parm string   */
If (ieasysxx = 'ØØØØ'x) Then            /* Not available?          */
   ieasysxx= '..'                       /* Yes                     */
Do While len > Ø                        /* Do while len > Ø?       */
   If (len > pdlen) Then Do             /* Length > pdlen?         */
      parmdesp= Substr(sysvar,i,pdlen)  /* Move pdlen bytes        */
      i= (I + pdlen)                    /* Next position           */
      len= (Len - pdlen)                /* Decrease the length     */
   End
   Else Do
      parmdesp= Substr(sysvar,i,len)    /* Move whats left         */
      i= (I + Len)                      /* Next position           */
      len= (Len - Len)                  /* Decrease the length     */
   End
   ADDRESS "ISPEXEC" "TBADD IEASYSTB
            SAVE(ieasysxx,sysparm,parmdesp) MULT("TABROWS")"
   if (rc ¬= Ø) then do                 /* Call OK?                */
      comperr= 'IEASYS'                 /* Component error         */
      ADDRESS "ISPEXEC" "SETMSG MSG(SCPIMØØ3)" /* Issue message    */
      leave                             /* Lets quit               */
   end
    ieasysxx= ''                        /* Reset                   */
    sysparm=  ''                        /* Reset                   */
End
return
/*.............................................................*/
/* Panel allocation section                                   .*/
/*.............................................................*/
Allocation_section:
address "ISPEXEC" "LIBDEF ISPPLIB DATASET ID('SHTSØØ1.SHL.LIB.PANELS')"
address "ISPEXEC" "LIBDEF ISPMLIB DATASET ID('SHTSØØ1.SHL.LIB.MSGS')"
return
/*.............................................................*/
/* Panel deallocation section                                 .*/
/*.............................................................*/
Deallocate_section:
address "ISPEXEC" "LIBDEF ISPPLIB"      /* Remove allocation       */
address "ISPEXEC" "LIBDEF ISPMLIB"      /* Remove allocation       */
return
/*.............................................................*/
/* This routine will test a BIT                               .*/
/*.............................................................*/
```

```
Testbit:
result= Ø                                  /* Init                    */
If (bitand(arg(1),arg(2))) = arg(2) then
   result= 1
return result
/*.............................................................*/
/* Initiate the variables                                      .*/
/*.............................................................*/
init_vars:
vcmd=  ''                                  /* IEACMDXX                */
vpago= ''                                  /* page (operator)         */
vpagp= ''                                  /* page (IEASYSxx)         */
vcon=  ''                                  /* CONSOLxx                */
vclk=  ''                                  /* CLOCKxx                 */
vcsa=  ''                                  /* CSA                     */
vsqa=  ''                                  /* SQA                     */
vsysp= ''                                  /* SYSP=                   */
vsch=  ''                                  /* SCHEDxx                 */
vsmf=  ''                                  /* SMFPRMxx                */
vssn=  ''                                  /* IEFSSNxx                */
vsvc=  ''                                  /* IEASVCxx                */
vprog= ''                                  /* PROGxx                  */
vpagt= ''                                  /* PAGTOTL                 */
vviod= ''                                  /* VIODSN                  */
vlogr= ''                                  /* LOGREC                  */
vlnk=  ''                                  /* LNKLSTxx                */
vlpa=  ''                                  /* LPALSTxx                */
vmlpa= ''                                  /* LNKLSTxx                */
vmstr= ''                                  /* MSTJCLxx                */
vlnka= ''                                  /* LNKAUTH                 */
return
/*.............................................................*/
/* Calculate the IPL date                                      .*/
/*.............................................................*/
calc_ipl_date:
iplyear= d2x(c2d(substr(mvslevel,6Ø,1))) /* YY                      */
iplday=  d2x(c2d(substr(mvslevel,61,2))) /* DDD                     */
iplday=  strip(iplday,T,F)             /* Remove the sign           */
if length(iplday) = 1 then             /* Insert two leading        */
   iplday= ØØ||iplday                  /* Ø if length = 1           */
else if length(iplday) = 2 then        /* Insert a leading          */
   iplday= Ø||iplday                   /* Ø if length = 2           */
ipldate= iplyear'.'iplday              /* The date                  */
Return                                 /* Return to caller          */
/*.............................................................*/
/* Calculate the IPL time                                      .*/
/*.............................................................*/
calc_ipl_time:
ipltime= c2d(Substr(mvslevel,55,4))    /* Get the IPL time          */
ss=      ipltime//6ØØØ                /* The number of secs        */
ss=      ss%1ØØ                       /* 1/1ØØ of a sec            */
if (ss < 1Ø) then                      /* Leading zero needed       */
```

```
   ss=   insert('0',ss)              /* Insert it                */
mins=    ipltime%6000              /* Convert to secs          */
mm=      mins//60                  /* Convert to mins          */
if (mm < 10) then                  /* Leading zero needed      */
   mm=   insert('0',mm)              /* Insert it                */
hh=      mins%60                   /* Convert to hours         */
if (hh < 10) then                  /* Leading zero needed      */
   hh=   insert('0',hh)              /* Insert it                */
ipltime= hh':'mm':'ss              /* The time                 */
return                             /* Return to caller         */
```

The following ISPF panels will be required:

SCPPAN01

```
)ATTR
* TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
# TYPE(TEXT)    INTENS(HIGH) COLOR(TURQ)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(WHITE)
@ TYPE(OUTPUT)  COLOR(RED)
)BODY WINDOW(50,13)
%
+COMMAND ===>_ZCMD         +  SCROLL ===>_AMT   +
%
%
#          Processor configuration           :_z%
#          CPU information                   :_z%
#          CPU vector and crypto information :_z%
#          HSA information                   :_z%
#          IPL information                   :_z%
#          IEASYSnn IPL information          :_z%
#
+ Please select an option by entering a '/'
%
)INIT
.ZVARS= '(spcfg,scpu,scpuvc,shsa,sipl,sysi)'
&ZCMD=  ' '
&pdlen= 30
VPUT (pdlen) SHARED
&spcfg=  '_'
&scpu=   '_'
&scpuvc= '_'
&shsa=   '_'
&sipl=   '_'
&sysi=   '_'
&check=  ' ,_,/'
.CURSOR= spcfg
&ZWINTTL=  'Processor configuration details'
)REINIT
```

```
&ZWINTTL=  'Processor configuration details'
&spcfg=  '_'
&scpu=   '_'
&scpuvc= '_'
&shsa=   '_'
&sipl=   '_'
&sysi=   '_'
&check=  ' ,_,/'
.CURSOR= spcfg
)PROC
ver (&spcfg,listv,&check)
ver (&scpu,listv,&check)
ver (&scpuvc,listv,&check)
ver (&shsa,listv,&check)
ver (&sipl,listv,&check)
ver (&sysi,listv,&check)
)END
```

SCPPAN02

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
@ TYPE(OUTPUT)  COLOR(RED)
* TYPE(TEXT)    INTENS(HIGH) COLOR(YELLOW)
? TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(76,22)
+COMMAND ===>_ZCMD      +               SCROLL ===>_AMT   +
%
* Product owner  :#z              * Product Name    :#z   *
* FMID           :#z              * Product Version :#z   *
* SP Level       :#z              * IPL Volser      :#z   *
* IPL Date       :#z              * IPL Time        :#z   *
%
? Real Storage Information.             Processor Information.
%
* Address Range:               :#z  *    No Of CPUs installed :#z *
* Address Incr In Units of 1M: :#z  *    No Of HSAs           :#z *
* Block Size in Units of 1K:   :#z  *    Loadparm             :#z *
* Increment Block Interleave   :#z  *    Vector section size  :#z *
* Interval                               Vector partial sum No:#z *
%
? Extended Storage Information.
%                                      *________________-
* Address Range                :#z     *|%Display CEC Installed :_z*|
* No of 4K Storage Blocks in   :#z     *|%Facilities*               |
* an Extended Storage Incr              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* Maximum Extended Storage     :#z  *
* Element Number
```

```
)INIT
.ZVARS= '(prodo,prodn,mvsfmid,prodv,msplevel,iplvol,ipldate,ipltime, +
sar,ncps,sai,nhsa,sbs,lparm,sii,vss,vpsm,mesi,dcecf,nxsb,mese)'
&dcecf=  'y'
&ZCMD=  ' '
&check= 'y,Y,n,N'
.CURSOR= dcecf
&ZWINTTL=  'Processor Configuration Details'
)REINIT
&dcecf=  'y'
&check= 'y,Y,n,N'
&ZWINTTL=  'Processor Configuration Details'
.CURSOR= dcecf
)PROC
ver (&dcecf,listv,&check)
)END
```

SCPPAN03

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
@ TYPE(OUTPUT)  COLOR(RED)
* TYPE(TEXT)    INTENS(HIGH) COLOR(YELLOW)
? TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(76,17)
+COMMAND ===>_ZCMD      +              SCROLL ===>_AMT   +
%
* Channel Path Information     :#Z*  Channel Path Subsystem Command    :#z*
* Channel Path Reconfiguration :#z*  CPU Information                   :#z*
* CPU Reconfiguration          :#z*  Signal Alarm                      :#z*
* Write Operator Message and   :#z*  Store Status On Load              :#z*
* Read Operator Response             Restart Reasons                  :#z*
* Instruction Address Trace Buf:#z*  Load Parameter                    :#z*
* Read And Write Data          :#z*  Real Storage Increment Reconfig   :#z*
* Real Storage Element Info    :#z*  Real Storage Elememt Reconfig     :#z*
* Copy and Re-assign Storage   :#z*  Copy and Re-assign Storage List   :#z*
* Extended Storage Usability   :#z*  Extended Storage Element Info     :#z*
* Map                                Extended Storage Element Reconfig :#z*
* Vector Feature Reconfig      :#z*  Read Write Event Feature          :#z*
* Read Resource Group Info     :#z*  Suppress On Protection            :#z*
* Initiate Reset               :#z*  Store Channel Subsystem           :#z*
*                                    Characteristics
)INIT
.ZVARS= '(chpi,chps,chpr,cpui,cpur,sgnl,omr,stst,rstr,itrc,lprm, +
wdat,sir,sei,ser,cars,carl,sum,esei,eser,vfr,evnt,rrgi,sopf,irin,cscf)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'CEC installed facilities'
```

```
)REINIT
&ZWINTTL=  'CEC installed facilities'
.CURSOR= ZCMD
)PROC
)END
```

SCPPAN04

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(GREEN)
@ TYPE(OUTPUT)  COLOR(RED)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
* TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(YELLOW)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(76,14)
%
+COMMAND ===>_ZCMD      +
%
% CPU            CPU   PSA       PSA        Interrupt  TOD    Private  PER2
% ID             Addr  VIRTUAL   REAL       Subclass   Number Space
%                      Address   Address    Mask
%
)MODEL
 ?z           % ?z   %?z       %?z       % ?z   %     ?z  %     ?z%    ?z%
)INIT
.ZVARS= '(pcpid,pcpua,ppsav,ppsar,piscm,stod,pspace,per2)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'CPU information'
)REINIT
&ZWINTTL=  'CPU information'
.CURSOR= ZCMD
)PROC
)END
```

SCPPAN05

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(GREEN)
@ TYPE(OUTPUT)  COLOR(RED)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
* TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(YELLOW)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(27,9)
%
+COMMAND ===>_ZCMD      +
%
%  HSA Size  HSA address
```

```
%
)MODEL
  ?z     %  ?z       %
)INIT
.ZVARS= '(hssz,hsaa)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'HSA information'
)REINIT
&ZWINTTL=  'HSA information'
.CURSOR= ZCMD
)PROC
)END
```

SCPPAN06

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
@ TYPE(OUTPUT)  COLOR(RED)
* TYPE(TEXT)    INTENS(HIGH) COLOR(YELLOW)
? TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(76,22)
+COMMAND ===>_ZCMD      +               SCROLL ===>_AMT   +
%
* Product Owner  :#z*             Product Name    :#z                    *
* FMID           :#z*             Product Version :#z                    *
* SP Level       :#z*             IPL Volser      :#z                    *
* IPL Date       :#z*             IPL Time        :#z                    *
%
? IPL Information                 Master Catalog Information
? ———————                  ———————————————
* Load Parm      :#z*             MCAT Dsn        :#z                    *
* IODF DSN       :#z*             MCAT Vol        :#z                    *
* Config Id      :#z*             MCAT Type       :#z                    *
* Load Dev Supp  :#z*             Alias Level     :#z                    *
* EDT Id         :#z*             CAS Srv Task    :#z                    *
* Nucleus Suffix :#z*
* Nuclst  Suffix :#z*             ?IPL Load PARMLIB Information
* SYSPLEX Name   :#z*             ?———————————————
* LPAR Name      :#z*             PARMLIB Dsn     :#z                    *
* HWNAME         :#z*             PARMLIB Devno   :#z                    *
* IEASYS  Suffix :#z                                                     *
* IEASYSM Suffix :#z                                                     *
% Type a '/' to Display Parmlib Usage* :_z*
)INIT
.ZVARS= '(prodo,prodn,mvsfmid,prodv,msplevel,iplvol,ipldate,ipltime,       +
lparm,mcatd,iodfdsn,mcatv,iocfg,mcatt,iodds,mcatal,ioedt,ccas,nucid,       +
nuclstid,sysplex,lparname,plibdsn,hwname,plibddv,sysparm,ieasym,           +
```

```
pusage)'
&ZCMD=  ' '
.CURSOR= pusage
&pusage= '_'
&ZWINTTL=  'IPL information'
)REINIT
&ZWINTTL=  'IPL information'
.CURSOR= pusage
&pusage= '_'
)PROC
)END
```

SCPPAN07

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
@ TYPE(OUTPUT)  COLOR(RED)
* TYPE(TEXT)    INTENS(HIGH) COLOR(YELLOW)
? TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(RED)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(51,1Ø)
+COMMAND ===>_ZCMD      +   SCROLL ===>_AMT   +
%
* Parmlib Dsn              :#z*
* Parmlib Volser           :#z*
* Default Parmlib          :#z*
* Parmlib In Use           :#z*
* Parmlib Found In Catalog :#z*
* Parmlib Locate Failed    :#z*
* Parmlib Mount  Failed    :#z*
* Parmlib Open   Failed    :#z*
)INIT
.ZVARS= '(parmdsn,parmvol,parmdef,parmiuse,parmcat,parmloc,parmmnt,parmopen)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'PARMLIB usage information'
)REINIT
&ZWINTTL=  'PARMLIB usage information'
.CURSOR= ZCMD
)PROC
)END
```

SCPPAN08

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(GREEN)
@ TYPE(OUTPUT)  COLOR(RED)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
* TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(YELLOW)
```

```
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(60,20)
%
+COMMAND ===>_ZCMD      +
%
% IEASYSxx  IEASYSxx    Parameter
%  Suffix   Parameter   Description
%
%
)MODEL
    ?z %   ?z       %  ?z                              %
)INIT
.ZVARS= '(ieasysxx,sysparm,parmdesp)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'IPL IEASYSxx parameters'
)REINIT
&ZWINTTL=  'IPL IEASYSxx parameters'
.CURSOR= ZCMD
)PROC
)END
```

SCPPAN09

```
)ATTR
_ TYPE(INPUT)   INTENS(HIGH) COLOR(YELLOW)
# TYPE(OUTPUT)  INTENS(HIGH) COLOR(GREEN)
@ TYPE(OUTPUT)  COLOR(RED)
? TYPE(OUTPUT)  INTENS(HIGH) COLOR(TURQ)
* TYPE(TEXT)    INTENS(HIGH) COLOR(GREEN)
% TYPE(TEXT)    COLOR(YELLOW)
+ TYPE(TEXT)    COLOR(WHITE)
)BODY WINDOW(70,14)
%
+COMMAND ===>_ZCMD      +
%
% CPU            CPU   Vector     Vector     Vector         Crypto
% ID             Addr  Feature    Feature    Feature In     Feature
%                      Installed  Connected  Standby State  Installed
%
)MODEL
 ?z           % ?z   %  ?z%         ?z%        ?z%            ?z%
)INIT
.ZVARS= '(pcpid,pcpua,pvfi,pvfc,pvfss,pcfi)'
&ZCMD=  ' '
.CURSOR= ZCMD
&ZWINTTL=  'CPU vector and crypto information'
)REINIT
&ZWINTTL=  'CPU vector and crypto information'
.CURSOR= ZCMD
)PROC
)END
```

ISPF MESSAGE SCPIM00

```
SCPIM001 'INVALID OPTION' .WINDOW=RESP .ALARM=YES .TYPE=ACTION
'PLEASE USE A ''/'' TO SELECT AN OPTION'
SCPIM002 'NO HSAS' .WINDOW=RESP .ALARM=YES .TYPE=ACTION
'NO HSA INFORMATION IS AVAILABLE'
SCPIM003 'TABLE SERVICES ERROR' .WINDOW=RESP .ALARM=YES .TYPE=ACTION
'AN ERROR HAS OCCURRED DURING TABLE SERVICES FOR COMPONENT &COMPERR'
```

The LIBDEF statements contained in the REXX EXEC must be changed to include the required panel and message library dataset names. To initiate the dialog, enter the command, %DSCPINFO, directly on the command line, or include the command on a relevant panel selection.

THE MVSVAR TSO/E EXTERNAL FUNCTION

IBM has provided a TSO/E external function, in OS/390, named MVSVAR. It can be used in REXX EXECs, and is invoked as follows:

```
MVSVAR(arg_name)
```

The information returned depends on the arg_value specified on the function call. Some of the items which can be specified include:

- SYSMVS – returns the level of the base program (BCP) component of OS/390.
- SYSOPSYS – returns the OS/390 name, version, release, modification level, and FMID.
- SYSNAME – returns the name of the system the REXX EXEC is running on. The SYSNAME is specified in the SYS1.PARMLIB member IEASYSxx.
- SYSCLONE – returns the MVS system symbol representing its system name.
- SYSMDEF – returns symbolic variables defined within the MVS system.

Rem Perretta
Senior Systems Programmer
Millenium Computer Consultancy

ISPF command tool

INTRODUCTION

Version 4.2 of ISPF introduced a significant change in the ISPF command tables, as is explained below. However, if sites are going to make the best use of the command table improvements, they needed something better than the ISPF 3.9 Command Table Utility, so I wrote an ISPF commands tool. It is designed for users to manage their command tables, making it easy to display and modify the active ISPF commands. This article describes the use and customization ISPF command tables, followed by the benefits using this tool.

PREVIOUS ISPF COMMAND CUSTOMIZATION

ISPF 3.9 can open INACTIVE command tables for update from your ISPTLIB ddname and write the updated table to your ISPTABL ddname. However, it opens ACTIVE tables for display only. Therefore, it is normal to copy the dataset member of active tables to a new membername and use ISPF 3.9 to update the copy, then manually copy the updated member back over the original member. This process is often confusing for users. Incidentally, do not believe IBM manuals saying that command tables can be in your ISPPROF ddname; neither ISPF nor the Command Table Utility read them from there.)

There has always been the ISPCMDS table, which contained the standard ISPF commands. In addition, each ISPF application could have its own command table (called: xxxxCMDS, where xxxx is the ISPF application-id), concatenated ahead of the ISPCMDS table. If the site wanted to add some extra local commands they could add them to ISPCMDS (eg, command SDSF might be added so that all users can invoke SDSF from almost any ISPF panel). Then some users wanted their own special commands as well, so they would copy ISPCMDS to their own ISPF table datasets and modify it there. Typically these users were using commands to invoke some standard application without needing a selection panel or invoking a self-written EXEC via a command.

This was all OK as long as the base ISPCMDS table did not change. However, if a new ISPF release was installed, there was a new ISPCMDS table, so the systems programmers had to add their special site commands to it, and any users with their own copies also had to recreate their command tables too. If any user did not recreate their own ISPCMDS tables, they would not have the correct set of basic ISPF commands.

WHAT IS NEW WITH 4.2?

To make this much easier, IBM added two new types of command table with Version 4.2 of ISPF:

- Site table – for extra system-wide commands
- User table – for individual users to maintain.

ISPF option 3.9 was changed too, but only insofar as to show the names of the active command tables. The names of the site and user tables are defined in the ISRCONFG module, which is usually in SYS1.SISPLPA, but could also be in a STEPLIB or ISPLLIB dataset for testing. Member ISRCNFIG of ISP.SISPSAMP is used to customize it, as is described in ISPF planning and customization manual section on *Tailoring PDF Defaults*. By default, the names for these tables are blank, which means that they are not used. Here is a sample of a part of this member customized to add site and user tables called SITECMDS and MYCMDS respectively:

```
USERCMDS DC  CL4'MY  '          APPLID FOR USER COMMAND TABLE      @SM2
SCTSRCH  DC  CL1'A'             SITE COMMAND TABLE SEARCH ORDER    @SM2
*                               SPECIFY B FOR BEFORE ISP           @SM2
*                               SPECIFY A FOR AFTER ISP            @SM2
SITECMDS DC  CL4'SITE'          APPLID FOR SITE COMMAND TABLE      @SM2
```

Now the site can maintain its own special commands in the SITECMDS table and leave the ISPCMDS table standard. Any user wanting their own commands can now use ISPF 3.9 to create a MYCMDS table in one of their own datasets, and it will not need to be changed for new ISPF releases. (Note that the ISRCONFG module is normally customized via an SMP/E USERMOD and that must be re-customized for each new ISPF release, using the new ISRCNFIG member from the new ISPF sample library.)

You can also see the parameter SCTSRCH is set to A so that the SITECMDS table is AFTER the ISPCMDS table. This ensures that the standard ISPF commands cannot be accidentally overridden by the site commands. Since there should be no matching commands in the site table, it should not matter whether it is before or after the ISPCMDS table, but I think it is better to play safe by using this order. Therefore the order ISPF uses to search for commands would be:

1 Application command table (if the application has one)

2 User command table (if the user has one)

3 ISPCMDS command table

4 Site command table.

The tables must be in datasets allocated in ddname ISPTLIB before ISPF is started, except for the application command table, which could also be dynamically allocated via the ISPF LIBDEF service.

The command tables are normal, unsorted, non-keyed ISPF tables with one command definition in each row. ISPF searches the command tables in the row order until it finds the first matching command definition, so it is easy for a user to define their own command to override one from the ISPCMDS or site table. It is also possible for the same command to be defined more than once in one table; then only the first definition is active. That is normally only done by mistake, but if the command table has a large number of commands it can occur (eg, SDSFs command table is ISFCMDS and it has the END command defined twice).

There are, however, valid uses of repeating a command in a table: that is when the first has an action field which is blank, or a variable name (eg, &VAR) which has a value of null or blanks in that ISPF application. Then ISPF will go past the first command (with the null action) and use the second command with the same name. Later the application may put a valid action into the &VAR variable, making the first command active. Or perhaps you want to replace a command temporarily, so you repeat the command and change the action of the first to the new desired action. When you have finished using your modified command, you simply delete it from the table and the original command will be active again.

WHY DO YOU NEED THIS TOOL?

As described above, renaming members and using ISPF 3.9 is very clumsy for modifying command tables even if you are only changing a copy of ISPCMDS, which already has 72 commands. However, now that users can have up to 4 four active tables, it is also desirable to have more functionality than is provided by the ISPF 3.9 utility. Therefore this dialog was written, providing that extra functionality and effectively replacing the old utility for most users. The tool can do the following:

1 While in any ISPF application, it can display the full list of available commands from all the command tables active in that application, or you can select just particular tables.

2 The list of commands is in the standard ISPF order, or you can sort the list into different orders (eg, command name order).

3 You can list only the commands with names matching a particular mask (eg listing only commands beginning with REF).

4 It always indicates commands which overide others and also which commands are overridden (ie inactive), regardless of which commands you have listed and their sort order.

5 It can display which libraries the active command tables are in, plus some table statistics.

6 You can add/update/delete the commands in any active command table, and the changes are immediately active, without restarting ISPF. Thus you can temporarily change the commands within any ISPF application while you are using it.

7 You can permanently save any command changes to disk for user (and site) tables while they are active; but it will not save changes to application or ISPCMDS tables, since that should not be necessary. It searches your ISPTLIB files for the appropriate library, then dynamically allocates it and replaces the member.

8 It can create a new user command table (with one dummy command) for any user who wants to start using their own commands, and the new table can be immediately active.

9 You can execute a command by selecting it directly from a displayed command list; this can be done recursively too (eg,

invoke ISPFCMDS then select a command which invokes another application, then start ISPFCMDS again in that application, then select another command to start yet another application, then start another ISPFCMDS, etc).

USING THE TOOL

If your site does not have a customized ISRCONFG module with site and user tables, I recommend creating one for yourself in a library allocated to ddname ISPLLIB for testing. (But put it into LPA for production use.)

When this tool is being used in production I recommend creating a site table plus an empty user table in a library with universal read-only access, and allocated at the end of every ISPTLIB concatenation. If you do not create that empty table, you can specify the user table prefix name at the beginning of the ISPFCMDS EXEC (USERPREF = xxxx). Then this tool will know the correct name for the user table when a user wants to create a new one in their own ISPF table library. Allocate the dialog components shown in Figure 1:

	Member	Description	Allocation
1	ISPCMDS	REXX EXEC	SYSEXEC or SYSPROC
2	ISPCMDn	CMD display panels (n = 0,1,2,3)	ISPPLIB
3	ISPCMDn	copy of IBM panels (n = 4,5)	ISPPLIB
4	ISPCMHn	Help panels (n = 1,2,3,4,5,6)	ISPPLIB

Figure 1: Table dialog components

- Check that ISPFCMDS EXEC sets USERPREF as you want it. Compiling this EXEC does not make a great difference to the speed since most of the work is done by ISPEXEC (ISPLINK); but it does reduce the size and it runs a bit more efficiently, so I recommend that you compile it.
- IBM panels ISPUCMX and ISPUCMXR should be copied to ISPFCMD4 and ISPFCMD5 to make sure the tool will continue to work if IBM changes those panels in a future version of ISPF.

- You may want to customize the text in panels ISPFCMD1 and ISPFCMH5 for your site, to give more detailed advice for creating/allocating a user table library.

It is invoked by 'TSO %ISPFCMDS'; or better yet - create a command!

```
  Verb  . . . : CMDS
  Trunc . . . : Ø
  Action  . . : SELECT CMD(%ISPFCMDS)
  Description : 'ISPF Commands' tool
```

The HELP panels give all the information you should need to use it.

PANEL ISPFCMH1

```
)ATTR
/*-------------------------------------------------------------------*/
/*  HELP panel for "ISPF Commands" tool (ISPFCMDS)                   */
/*  - from panel ISPFCMDØ. ISPFCMH2,ISPFCMH3,ISPFCMH4,ISPFCMH5 follow.*/
/*-------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT) INTENS(HIGH) COLOR(WHITE)
% TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-#ISPF Commands+-\-\-HELP
+
%This dialog displays the contents of the ISPF command tables that are active
%in the current ISPF application (from which this was started).
%
%It shows the command name, table and description, and can optionally also
%show the command action.
%
%You can change the list by restricting which tables to display, by changing
%the order, or by restricting the commands by name (matching a mask).
%
%You can (temporarily) update any of the tables via line commands, and you
%can permanently save changes to User commands.
%
%
   Press#ENTER%for%DISPLAY FIELDS,%then%PRIMARY COMMANDS,%then%LINE COMMANDS
              %then%CREATE A USER COMMAND TABLE
)INIT
)PROC
 &ZCONT = ISPFCMH2
)END
```

PANEL ISPFCMH2

```
)ATTR
/*-------------------------------------------------------------------*/
/*  HELP panel for ISPF Commands tool (ISPFCMDS)                     */
```

```
/*  -
/*-------------------------------------------------------------------*/
¬ TYPE(INPUT)  INTENS(HIGH) CAPS(ON) PADC(_)
! TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
¢ TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ) CAPS(OFF) SKIP(ON)
~ TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
# TYPE(TEXT)   INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)
% TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-#ISPF Commands+-\-\-HELP
+                                                  ¨Z$Application Id :!Z
%               DISPLAY FIELDS:                   +¨Z$User table . . :!Z
+                                                  ¨Z¢DESC3         +:!TB3
+  Command  +Table +Description                    ¨Z¢DESC4         +:!TB4
+ ¨CMDSCAN ++----- ---------------------------------- Sorted by¨CMDSORT
+
%The active command tables are listed in order at the top right.  When there is
#/%next to the table name - its commands are listed below.  You can change any
%of the#/%symbols to a blank to NOT list commands from that table.
%
$*%next to a command indicates it overrides another command with the same name
$-%next to a command indicates it is an overridden command (ie not active)
%
%When the command$Actions%are shown (in red), the$Command%field is underlined if
%the number of required characters is less than the whole command name.
%
%The commands are sorted in#STANDARD ORDER%by default; that is the order which
%ISPF searches them to find a command. You can also sort them by#COMMAND, TABLE,
#ACTION%or#DESCRIPTION%by entering the first character after+Sorted by.
%
%To list only commands starting with a particular string, enter it under the
+Command%heading (eg. enter#KEY%to list only commands beginning with 'KEY')
)INIT
  .ZVARS = '(T1,ZAPPLID,T2,ZUCTPREF,T3,T4)'
)PROC
  &ZCONT = ISPFCMH3
)END
```

PANEL ISPFCMH3

```
)ATTR
/*-------------------------------------------------------------------*/
/*  HELP panel for "ISPF Commands" tool (ISPFCMDS)                   */
/*  - from panel ISPFCMH2.  Panels ISPFCMH4,ISPFCMH5 follow.         */
/*-------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT) INTENS(HIGH) COLOR(WHITE)
% TYPE(TEXT) INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-#ISPF Commands+-\-\-HELP
+
```

```
%The following%PRIMARY COMMANDS%can be used:
%
$      1 %- Switch between: showing the command Actions, and NOT showing them
%
$      2 %- Display command table statistics and the datasets they are stored in
%
$      3 %- SAVE your User command table to disk (same as the SAVE command)
%
$  L xxx %- LOCATE the first command beginning with$'xxx'%in the displayed list
%
$NEWTABLE%- Create a new User command table
%
$   SAVE %- SAVE your User command table to disk (same as option 3)
%
$SAVESITE%- Save the Site command table (only for use by System Programmers)
%
$ SORT a %- SORT the list by$'a',%where$'a'%is S, C, T, A or D (see below)
%
%The SORT orders are$'Standard order', 'Command name', 'Table name', 'Action'%or
$'Description'.% The$'*'%and$'- '%characters indicate which commands override
%others and they remain correct regardless of which order you choose.
)INIT
  &ZCONT = ISPFCMH4
)PROC
)END
$    REF %- REFRESH the list of commands (this should not be necessary)
```

PANEL ISPFCMH4

```
)ATTR
/*--------------------------------------------------------------------*/
/*  HELP panel for ISPF Commands tool (ISPFCMDS)                      */
/*  - from panel ISPFCMH3.  Panel ISPFCMH5 follows.                   */
/*--------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
* TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
$ TYPE(TEXT) INTENS(HIGH) COLOR(WHITE)
% TYPE(TEXT) INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-#ISPF Commands+-\-\-HELP
+
%The following%LINE COMMANDS%can be entered beside any listed command:
%
$ S%or$X %- EXECUTE a command  (can use*Option%field to supply a &&ZPARM value)
%
$ B%or$V %- VIEW a command
%
$ E%or$U %- UPDATE a command
%
$      I %- INSERT a new command (AFTER the current command, in the same table)
%
```

```
$      R %- REPEAT a command (BEFORE the current command, in the same table)
%
$      D %- DELETE a command
%
%
%Note that any command table changes made via this panel will be effective
#immediately%and available for use.  However, the changes are only#temporary
%and remain active until the command table is closed, but they are not written
%to disk.  If you want to make#permanent%changes to your User commands you can
%use the*SAVE%command (see HELP for Primary Commands).
)INIT
&ZCONT = ISPFCMH5
)PROC
)END
```

PANEL ISPFCMH5

```
)ATTR
/*--------------------------------------------------------------------*/
/*  HELP panel for ISPF Commands tool (ISPFCMDS)                      */
/*  - from panel ISPFCMD1 or ISPFCMH4.  Panel ISPFCMH1 follows.       */
/*--------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
$ TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT) INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-$ISPF Commands+-\-\-HELP
+
%      You can%CREATE A USER COMMAND TABLE%as follows:
%
% 1% Allocate a PDS (RECFM=FB,LRECL=8Ø) for your own ISPF table library and
%    include it in your concatenation for ddname ISPTLIB.
%
% 2% Use command#NEWTABLE%in this$"ISPF Commands"%tool to create a new User
%      commands table, with one dummy command for you to modify.
%
% 3% &H3
%
% 4% &H4
%    use the$"ISPF Commands"%tool to create your own commands, for example:
%      Verb  . . . :#LA
%      Trunc . . . :#Ø
%      Action  . . :#SELECT PGM(ISRDDN) SCRNAME(LA)
%      Description :#Show allocated datasets using standard IBM program
%
% 5% Use command#SAVE%to write your updated table to disk
)INIT
&ZCONT = ISPFCMH1
IF (&ZUCTPREF = &Z)           /* no current active User command table */
  &H3='Exit from "ISPF Commands" by repeated PF3, then exit from ISPF.'
  &H4='Start ISPF again and your new command table will be active.  Now you can'
```

```
ELSE             /* existing active User command table to be replaced */
  &H3='Select your ISPF table library, then press ENTER'
  &H4='Your new command table will be immediately active.  Then you can'
)PROC
)END
```

PANEL ISPFCMH6

```
/*--------------------------------------------------------------------*/
/*  HELP panel for ISPF Commands tool (ISPFCMDS)                      */
/*  - from panel ISPFCMD3.  Panel ISPFCMH1 follows.                   */
/*--------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
@ TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
% TYPE(TEXT) INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-#ISPF Commands+-\-\-HELP
%
 This panel lists the currently active ISPF command tables.  They are normal
 ISPF tables with one command per row.  The System table is called%ISPCMDS%and
 it is always required by ISPF, but the other tables are not always present.
 User and Site tables can only be used when their names are specified in the
 ISRCONFG module, which is in the LINKLIST, or in a STEPLIB or ISPLLIB library.
%
 The@User, Site%and@System%tables are stored in your ISPTLIB concatenation, and
 they are opened when ISPF is started.  If the User or Site table names are not
 defined in ISRCONFG, or if they are not found - ISPF works without them.
%
 The@Application%command table is also stored in your ISPTLIB concatenation
 or in a temporary allocation (LIBDEF) to ISPTLIB.  This panel will show the
@Dataset Name%preceeded by%***%if the table is found in a LIBDEF'd dataset.
 If the table exists, it is opened when the ISPF application is started.
)INIT
&ZCONT = ISPFCMH1
)PROC
)END
```

PANEL ISPFCMD0

```
)ATTR
/*--------------------------------------------------------------------*/
/*  Panel to display a table of the active ISPF commands              */
/*  - displayed by EXEC ISPFCMDS ("ISPF Commands" tool)               */
/*--------------------------------------------------------------------*/
¨ TYPE(INPUT)  INTENS(HIGH) COLOR(RED)  CAPS(ON)  PADC(_)
! TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
¢ TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ) CAPS(OFF) SKIP(ON)
# TYPE(OUTPUT) INTENS(LOW)  COLOR(GREEN)
~ TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
* TYPE(TEXT)   INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)
% TYPE(TEXT)   INTENS(LOW)  COLOR(GREEN)
```

```
)BODY EXPAND(\\)
+-\-\-*ISPF Commands+-\-\-
+Option%==>_ZCMD
+ %1 `&OPT1                                          + ¨Z$Application Id :!Z
+ %2 `Display Command Table information              + ¨Z$User table . . :!Z
+ %3 `Save your User Commands on disk                + ¨Z¢DESC3
+:!TB3
+                                                    + ¨Z¢DESC4
+:!TB4
+  Command  +Table +Description
+ ¨CMDSCAN ++----- ----------------------------------  Sorted by¨CMDSORT
)MODEL ROWS(SCAN)
¨Z¢Z!ZCTVERB ¢Z   #ZCTDESC
+
&MODELIN2
)INIT
/*  ~CMDTRUNC~ZCTACT
+
&ZCMD     = '&SELCMD &ZCMD'         /* used for executing a command  */
&SELCMD   = &Z
If (&CMDSACT = &Z)                  /* setting text for first option */
    &OPT1 = 'Also show Command Actions'
Else                                /* &CMDSACT = '/'                */
    &OPT1 = 'Show only Command Descriptions'
&SEL      = &Z
&CUR#TABS = '&T1,&T2,&T3,&T4'       /* the tables currently selected */
&CUR#SCAN = &CMDSCAN                /* mask for scan of command rows */
&CUR#SACT = &CMDSACT                /* the value of 'Show Actions'   */
&CUR#SORT = &CMDSORT                /* the current sort order        */
.ZVARS = '(T1,ZAPPLID,T2,ZUCTPREF,T3,T4,SEL,CMDOVER,CMDTAB)'
.HELP = ISPFCMH1
If (&ZSCTSRCH = B)                  /* variable from ISRCONFG module */
    &DESC3 = 'Site table . .'
    &TB3   = &ZSCTPREF              /* name of table prefix, if open */
    &DESC4 = 'System table .'
    &TB4   = 'ISP'
Else                                /* &ZSCTSRCH = A                 */
    &DESC3 = 'System table .'
    &TB3   = 'ISP'
    &DESC4 = 'Site table . .'
    &TB4   = &ZSCTPREF
&ZSCROLLD = CSR                     /* always scroll by cursor pos'n */
)REINIT
&ZCMD     = '&SELCMD &ZCMD'         /* used for executing a command  */
&SELCMD   = &Z
REFRESH(*)                          /* refresh all variables         */
)PROC
If (&ZCMD = 1)                      /* switching display MODEL line  */
    If (&MODELIN2 = 'OMIT')
        &MODELIN2 = '    ~CMDTRUNC~ZCTACT' /* define 2nd )MODEL line */
    Else
        &MODELIN2 = 'OMIT'          /* do not display 2nd )MODEL line*/
```

```
If (&ZTDSELS ¬= 0000)              /* at least one row selected     */
    VER(&SEL,LIST,I,R,D,U,E,V,B,S,X)
If (&T1 ¬= &Z) &T1 = '/'
    If (&ZAPPLID = 'ISP') &T1 = ' ' /* ISP is only the System table  */
        If (&TB3 = ISP) &T3 = '/'
        If (&TB4 = ISP) &T4 = '/'
If (&T2 ¬= &Z)
    If (&ZUCTPREF = &Z) &T2 = ' '   /* No User table open            */
    Else &T2 = '/'                  /* User table exists             */
If (&T3 ¬= &Z) &T3 = '/'
If (&T4 ¬= &Z) &T4 = '/'
)END
```

PANEL ISPFCMD1

```
)ATTR
/*-------------------------------------------------------------------*/
/*  Panel for user to create a new table of ISPF User commands       */
/*  - displayed by EXEC ISPFCMDS (ISPF Commands tool)                */
/*-------------------------------------------------------------------*/
¬ TYPE(INPUT)  INTENS(HIGH) COLOR(RED) CAPS(ON) PADC(_)
! TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
* TYPE(TEXT)   INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)
` TYPE(TEXT)   INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+-\-\-*ISPF Commands+-\-\-
+Command%==>_ZCMD
+
`     Your new &USRTABL command table will be stored in your personal
`     ISPF table library, which must be allocated in your ISPTLIB ddname
`     concatenation. Select it from the following list by entering$'S'.
`
`     If you do not have your own ISPF table library yet:
`       1) exit from*"ISPF Commands"`by repeated$PF3`(END)
`       2) create a PDS library (RECFM=FB,LRECL=80) for your tables
`       3) exit from ISPF, then add the library to your ISPTLIB ddname
`       4) start ISPF, invoke*"ISPF Commands"`, then try this again.
+
+     Libraries allocated to ISPTLIB
+    -------------------------------
)MODEL
  ¬Z+!CMDSN1                                          +
)INIT
&ZCMD     = &Z
.ZVARS = '(SEL)'
.CURSOR = SEL
.CSRROW = 1                /* cursor on 1st line of dataset list */
.HELP = ISPFCMH5
&ZTDMARK = ' '             /* blank 'END OF DATA' line           */
)REINIT
REFRESH(*)                 /* refresh all variables              */
```

```
)PROC
)END
```

PANEL ISPFCMD2

```
)ATTR
/*--------------------------------------------------------------------*/
/*  Panel for user to confirm update of an ISPF table library         */
/*  - displayed by EXEC ISPFCMDS (ISPF Commands tool)                 */
/*--------------------------------------------------------------------*/
 + TYPE(TEXT)   INTENS(LOW)  COLOR(GREEN)
 ¢ TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ)
)BODY WINDOW(45,8)
+_DUMMY                                  +
+  Press%ENTER+to CONFIRM that you want  +
+  to save a new table member:¢TABNAME    +
+  in library:¢LIBNAME                   +
+  &TXTC                                 +
+                                        +
+  .. or press%PF3+(END) to CANCEL       +
+                                        +
)INIT
 &ZWINTTL = ''                      /* no heading for Pop-Up window */
 IF (&TSTAT = OK)
     &TXTC = 'to REPLACE the existing member'
 ELSE
     &TXTC = &Z
)PROC
)END
```

PANEL ISPFCMD3

```
)ATTR
/*--------------------------------------------------------------------*/
/*  Panel to display statistics of the active ISPF Command tables     */
/*  - displayed by EXEC ISPFCMDS (ISPF Commands tool)                 */
/*--------------------------------------------------------------------*/
¨ TYPE(INPUT)  INTENS(HIGH) COLOR(RED) CAPS(ON) PADC(_)
! TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
¢ TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ)   JUST(RIGHT) CAPS(OFF)
# TYPE(OUTPUT) INTENS(LOW)  COLOR(GREEN)
@ TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW) JUST(RIGHT)
~ TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
* TYPE(TEXT)   INTENS(HIGH) COLOR(YELLOW)
$ TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)
` TYPE(TEXT)   INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+-\-\-*ISPF Commands+-\-\-
+Command%==>_ZCMD                                    `Command search order
`     These are the command tables active in the     + ¨Z$Application Id :!Z
`     current ISPF application.  ISPF searches       + ¨Z$User table . . :!Z
```

```
`     them in the order they are shown here.          + ¨Z¢DESC3
+:!TB3
+                                                     + ¨Z¢DESC4
+:!TB4
+ Table   No. of      Last Updated
+ Name     Cmds   Date      Time     User              Dataset Name
+-------- +----  --------------------------  -------------------------------
)MODEL ROWS(ALL)
@CMDTABL ¢Z   + #CMDDATE  #CMDTIME  #CMDUSER #CMDDSN
)INIT
&ZCMD     = &Z
.ZVARS = '(T1,ZAPPLID,T2,ZUCTPREF,T3,T4,CMDROWS)'
&CUR#TABS = '&T1,&T2,&T3,&T4'    /* the tables currently selected */
.HELP = ISPFCMH6
If (&ZSCTSRCH = B)               /* variable from ISRCONFG module */
    &DESC3 = 'Site table . .'
    &TB3   = &ZSCTPREF           /* name of table prefix, if open */
    &DESC4 = 'System table .'
    &TB4   = 'ISP'
Else                             /* &ZSCTSRCH = A                 */
    &DESC3 = 'System table .'
    &TB3   = 'ISP'
    &DESC4 = 'Site table . .'
    &TB4   = &ZSCTPREF
&ZTDMARK = ' '
)REINIT
REFRESH(*)                       /* refresh all variables         */
)PROC
If (&T1 ¨= &Z)
    &T1 = '/'
    If (&ZAPPLID = 'ISP')        /* ISP is only the System table  */
        &T1 = ' '
        If (&TB3 = ISP) &T3 = '/'
        If (&TB4 = ISP) &T4 = '/'
If (&T2 ¨= &Z)
    If (&ZUCTPREF = &Z) &T2 = ' '/* No User table open           */
    Else &T2 = '/'               /* User table exists             */
If (&T3 ¨= &Z)
    &T3 = '/'
If (&T4 ¨= &Z)
    &T4 = '/'
)END
```

PANEL ISPFCMD4

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*-------------------------------------------------------------------*/
/*  - displayed by EXEC ISPFCMDS (ISPF Commands tool)                */
/*-------------------------------------------------------------------*/
/*  This is a copy of the ISPUCMX panel supplied with ISPF 4.x       */
/*  - it is used to UPDATE an existing command or to specify a NEW   */
/*    command.  This copy has been made just in case IBM make some   */
/*    major change to their panel in a later release of ISPF.        */
```

```
/*--------------------------------------------------------------------*/
)ATTR DEFAULT(   ) FORMAT(MIX)
    ..
    ..
    ..
)END
```

PANEL ISPFCMD5

```
)PANEL KEYLIST(ISRSNAB,ISR)
/*--------------------------------------------------------------------*/
/*  - displayed by EXEC ISPFCMDS (ISPF Commands tool)                 */
/*--------------------------------------------------------------------*/
/*  This is a copy of the ISPUCMXR panel supplied with ISPF 4.x       */
/*  - it is used to BROWSE an existing command definition.            */
/*    This copy has been made just in case IBM make some major        */
/*    change to their panel in a later release of ISPF.               */
/*--------------------------------------------------------------------*/
)ATTR DEFAULT(   ) FORMAT(MIX)
    ..
    ..
    ..
)END
```

ISPFCMDS EXEC

```
/*============================>> REXX <<================================*/
/*  ISPFCMDS   : This EXEC displays the contents of the currently     */
/*               active command tables. It also enables the user to   */
/*               (temporarily) update all the command tables and to   */
/*               (permanently) save the user or site table to disk.   */
/*                                                                    */
/*  Externals  : panels ISPFCMDn (n=Ø,1,3)           (table display) */
/*                      ISPFCMD2                                      */
/*                      ISPFCMD4, ISPFCMD5    (copies of IBM panels ) */
/*                     (ISPUCMX,  ISPUCMXR)   (<---- the IBM panels ) */
/*                      ISPFCMHn (n=1 to 6)          (HELP panels  ) */
/*               msgs   ISRZØØ2                (standard IBM message) */
/*                                                                    */
/*               Incorporating the (possible) 4 levels of CMD tables  */
/*               that could be active with ISPF 4.2 & later versions. */
/*                                                                    */
/*  Version    : 3.8          Last Updated:  August '98              */
/*====================================================================*/
  USRPREF = 'MY'  /* Default User command table prefix, should be the */
                  /* ......... same as the value in ISRCONFG module.  */
                  /*--------------------------------------------------*/
  Address ISPEXEC                        /* commands -> ISPEXEC      */
  "CONTROL ERRORS RETURN"                /* handle return codes here */
  "VGET (ZAPPLID ZUCTPREF ZSCTPREF ZSCTSRCH ZSCREEN) SHARED"

  Do n = Ø to 9  Until rc > Ø       /* create unique table names      */
```

```
     tsuf = ZSCREEN%%n              /* up to 1Ø suffixes per ZSCREEN  */
     cmds_table = 'CMDSTB'%%tsuf    /* table for list of commands     */
     "TBQUERY" cmds_table           /* check if table already exists  */
     End
  cmdtbs_table = 'CMDTBS'%%tsuPanel: ISPFCMH5
================================================================================
)ATTR
/*-------------------------------------------------------------------------*/
/*  HELP panel for ISPF Commands tool (ISPFCMDS)                           */
/*  - from panel ISPFCMD1 or ISPFCMH4.  Panel ISPFCMH1 follows.            */
/*-------------------------------------------------------------------------*/
# TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
$ TYPE(TEXT) INTENS(HIGH) COLOR(YELLOW)
% TYPE(TEXT) INTENS(LOW)  COLOR(GREEN)
)BODY EXPAND(\\)
+HELP-\-\-$ISPF Commands+-\-\-HELP
+
%      You can%CREATE A USER COMMAND TABLE%as follows:
%
% 1% Allocate a PDS (RECFM=FB,LRECL=8Ø) for your own ISPF table library and
%    include it in your concatenation for ddname ISPTLIB.
%
% 2% Use command#NEWTABLE%in this$"ISPF Commands"%tool to create a new User
%      commands table, with one dummy command for you to modify.
%
% 3% &H3
%
% 4% &H4
%    use the$"ISPF Commands"%tool to create your own commands, for example:
%      Verb  . . . :#LA
%      Trunc . . . :#Ø
%      Action  . . :#SELECT PGM(ISRDDN) SCRNAME(LA)
%      Description :#Show allocated datasets using standard IBM program
%
% 5% Use command#SAVE%to write your updated table to disk
)INIT
&ZCONT = ISPFCMH1
IF (&ZUCTPREF = &Z)           /* no current active User command table */
  &H3='Exit from "ISPF Commands" by repeated PF3, then exit from ISPF.'
  &H4='Start ISPF again and your new command table will be active. Now you
can'
ELSE             /* existing active User command table to be replaced */
  &H3='Select your ISPF table library, then press ENTER'
  &H4='Your new command table will be immediately active. Then you can'
)PROC
)END
```

Ron Brown
Systems Programmer (Germany)

Running TSO and ISPF in production batch

INTRODUCTION

Just because a particular facility can only be accessed from the TSO or ISPF environment does not eliminate it from being run in a production batch environment. TSO can be run in a batch job, and ISPF initiated within the TSO environment. A minimum amount of documentation is provided in the relevant IBM manuals, and it is even possible to generate the kinds of job step level return or condition codes you would expect in a batch job running in production.

Clearly, there are some things that were never intended for use in a production environment and some services that just will not work in batch. For example, anything that relies on the full-screen environment of the 3270 terminal or response from an intelligent human being that typifies the interactive nature of TSO's normal on-line operation. There is also considerable discussion at my local telco customer site as to whether the batch interface for products like FileAid (CompuWare) was never intended, and, therefore should not be used, for production batch jobs that process large volumes of records on a daily basis.

DFSMShsm

Even some of the major IBM systems software products do not provide direct access from batch. The hsm component of DFSMS is a good example. Without writing a program, the only way to access even the rudimentary functions that all users need is with TSO commands like HLIST, HMIGRATE, HRECALL, HBACKUP and HRESTORE.

Why is this functionality important in a batch job? Any production datasets that are accessed less frequently than the time specified in their management class definition for migration to Level 2, typically tape, can become a real bottleneck. A batch job that references quite a few migrated datasets can take a very long time to run. The reason: each recall occurs *serially* during open processing. In other words, migrated datasets are recalled one at a time; a recall for the second

dataset is not issued until the first dataset has been fully recalled. Even JES3, which premounts all required tape datasets before beginning execution of a batch job, will not recall migrated datasets prior to starting the job.

Obviously, how big a problem this is depends on how long it takes to mount a tape, how big the datasets are and whether most of the datasets were migrated at the same time and, therefore, reside on the same physical tape. Quarterly and annual processing is where the impact is most likely to be felt.

USING HRECALL

One solution is to issue HRECALL commands for the migrated datasets at the beginning of the job.

```
HRECALL ('PAYROLL.ANNUAL.EMPLOYEE.MASTER.DATA')
HRECALL ('PAYROLL.ANNUAL.DEDUCT.MASTER.DATA')
HRECALL ('PAYROLL.ANNUAL.PENSION.MASTER.DATA')
```

Since NOWAIT is the default, all of the recall requests are issued virtually simultaneously, and the recall process is suddenly parallel instead of serial, up to the limit of available tape drives for hsm, of course.

To issue TSO commands in batch, you actually use a slightly modified subset of the JCL that is executed whenever anyone logs on to TSO. Anyone who has viewed what appears to be a batch job in the JES*x* queues with a job name of their TSO user-id may have seen this JCL, since this batch job is actually their on-line TSO session. TSO has a program name of IKJEFT01 and is executed like any other program:

```
//TSOCMDS  EXEC PGM=IKJEFT01
//SYSTSIN  DD *
HRECALL ('PAYROLL.ANNUAL.EMPLOYEE.MASTER.DATA')
//SYSTSPRT DD SYSOUT=*
```

SYSTSIN is where TSO commands are read. The output normally sent to a terminal in response goes to SYSTSPRT. Unfortunately, hsm does things somewhat differently and its output is rarely seen in the output of a batch job. A logged-on TSO user will see the hsm messages on his screen for any batch jobs that are currently running under the same user-id.

Additional DD statements can be added based on the needs of the programs being run in TSO. This is generally a better approach than using TSO ALLOCATE commands as input to SYSTSIN. The few examples in the IBM manuals show an additional DD SYSOUT=* statement for SYSPRINT, but this is only required when running a program, utility, or command that actually uses it as its default output.

DFSMSdss

The dss component of DFSMS is an even bigger problem. Migrated datasets do not slow down dss, they speed it up, because they are ignored! The closest thing to a solution is to issue the HRECALL command twice for each dataset before running dss, first with NOWAIT (the default) for each dataset, then again for each dataset with WAIT.

```
//TSOCMDS  EXEC PGM=IKJEFT01
//SYSTSIN  DD *
HRECALL ('PAYROLL.ANNUAL.EMPLOYEE.MASTER.DATA')
HRECALL ('PAYROLL.ANNUAL.DEDUCT.MASTER.DATA')
HRECALL ('PAYROLL.ANNUAL.PENSION.MASTER.DATA')
HRECALL ('PAYROLL.ANNUAL.EMPLOYEE.MASTER.DATA') WAIT
HRECALL ('PAYROLL.ANNUAL.DEDUCT.MASTER.DATA') WAIT
HRECALL ('PAYROLL.ANNUAL.PENSION.MASTER.DATA') WAIT
//SYSTSPRT DD SYSOUT=*
```

There are two problems with this approach. Recall failures will not generate an abend, non-zero return code in the job step or even an error message printed anywhere in the job output. And generating more hsm requests than the installation-specified maximum will result in *all* the requests being flushed.

REXX

There are three ways to run a REXX EXEC in TSO in batch. REXX can also be run directly in batch, without TSO, interpretively from source code, or as a load module that has been compiled with the REXX compiler. Since REXX is frequently used to automate operations-type functions, the dataset and other systems functions that are only available within the TSO environment usually make it necessary to run REXX in TSO.

The first and most obvious method of running a REXX EXEC in TSO in batch is to use the same EXEC TSO command that you would use if you were on-line. The JCL would be similar to what was used with HRECALL above:

```
//TSOCMDS  EXEC PGM=IKJEFTØ1
//SYSTSIN  DD * EXEC 'USERID.APPL.EXEC(PROGNAME)'
//SYSTSPRT DD SYSOUT=*
```

Alternatively, the USERID.APPL.EXEC PDS, where the source code for the REXX EXEC is stored, could be specified as a library:

```
//TSOCMDS  EXEC PGM=IKJEFTØ1
//SYSEXEC  DD   DSN=USERID.APPL.EXEC,DISP=SHR
//SYSTSIN  DD *             %PROGNAME
//SYSTSPRT DD SYSOUT=*
```

If the TSO step is only being used to run the REXX EXEC, it can be simplified to:

```
//TSOCMDS  EXEC PGM=IKJEFTØ1,PARM=PROGNAME
//SYSEXEC  DD   DSN=USERID.APPL.EXEC,DISP=SHR
```

Single quotes or parenthesis are required in the 'PARM=' field if parameters are to be passed to the REXX EXEC.

ISPF

Most of what you can do on-line from a terminal running ISPF can also be done in REXX by calling ISPF services. The services that do not require a full-screen 3270 terminal to function can also be done in batch by running TSO as a job step. But all required ISPF libraries must be assigned just as they are when you log-on to TSO at your 3270 terminal. This can be done with JCL DD statements or TSO ALLOCATE commands.

To determine all the library and other assignments that are in effect when you run ISPF in your installation, logon to TSO, select ISPF Option 6 and type LISTALC STATUS. A more practical approach is to write a REXX EXEC to capture the information and put it in a file.

```
/*  REXX   */
/*  EXEC this REXX EXEC from ISPF Option 6 to capture all the
    datasets assigned when ISPF is running.  For example, type:
    ISPF 6
    EXEC 'id.REXX.EXEC(LISTALC)'
*/
"free file(ddlistdd)"
/*  The following statement will generate error messages the first
    time and any other time when id.DDLIST.LIST does not exist.  */
"delete (ddlist.list)"
"alloc file(ddlistdd) dataset(ddlist.list) new catalog" ,
"   space(1,1)"
junk = outtrap("listalc.", "*", "noconcat")
"listalc status"
junk = outtrap("off")
"execio * diskw ddlistdd (open finis stem listalc."
"free file(ddlistdd)"
The output from LISTALC STATUS can be hard to read:
—DDNAME—DISP—
SYS1.SEDGHLP1
  SYSHELP  KEEP
SYS2.HELP
           KEEP
SYS1.HELP
           KEEP
TERMFILE  SYSIN
TERMFILE  SYSPRINT
SYS1.BRODCAST
  SYSØØØØ1 KEEP,KEEP
SYS1.SISPTENU
  ISPTLIB  KEEP
SYS2.SEJETENU
           KEEP
ISP.NONIBM.TABLES
           KEEP
ISP.PPROD.TABLES
           KEEP
LPSDFEDM.PROD.TABLES
           KEEP
LPSDFEDM.PRODSYS.TABLES
           KEEP
SYS1.SISPEXEC
  SYSEXEC  KEEP
```

This would translate into the following JCL:

```
//SYSHELP  DD DSN=SYS1.SEDGHLP1,DISP=SHR
//         DD DSN=SYS2.HELP,DISP=SHR
//         DD DSN=SYS1.HELP,DISP=SHR
//SYSIN    DD *
```

```
//SYSPRINT DD SYSOUT=*
//SYS00001 DD DSN=SYS1.BRODCAST,DISP=SHR
//ISPTLIB  DD DSN=SYS1.SISPTENU,DISP=SHR
//         DD DSN=SYS2.SEJETENU,DISP=SHR
//         DD DSN=ISP.NONIBM.TABLES,DISP=SHR
//         DD DSN=ISP.PPROD.TABLES,DISP=SHR
//         DD DSN=LPSDFEDM.PROD.TABLES,DISP=SHR
//         DD DSN=LPSDFEDM.PRODSYS.TABLES,DISP=SHR
//SYSEXEC  DD DSN=SYS1.SISPEXEC,DISP=SHR
```

However, you should not code the SYS00001 DD because any DD name in the format SYSnnnnn has been dynamically allocated by ISPF. There are usually a lot of DD assignments that can be eliminated, such as application-specific libraries for other applications and temporary datasets used to capture utility control cards or JCL for specific ISPF functions, like move/copy option 3.3.

The JCL to run ISPF in batch is really just an execution of ISPF within TSO. Begin with:

```
//TSOCMDS  EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
ISPSTART CMD(%PROGNAME)
//SYSEXEC  DD   DSN=USERID.APPL.EXEC,DISP=SHR
```

Then add all of the ISPF library assignment DD statements. Alternatively, they could be included as ALLOCATE statements before the ISPSTART statement. The plus (+) and minus (-) signs can be used, just as they are in TSO, as continuation signs when required for concatenating many libraries.

This technique has only been tested in a JES3 environment where the batch job will wait in Allocation state until the TSO user logged on to the ID under which the job is being run has logged off. This is likely the result of the developers of ISPF relying, intentionally or unintentionally, on TSO's prohibition of multiple simultaneous on-line sessions under a single user-id.

SETTING RETURN CODES

A big part of data centre management involves determining whether or not each production batch job completed successfully or not. The return or condition code of each job step is the most obvious way to

determine the success or failure of a step and, taken together, they determine the success or failure of the job.

Even when abends, rather than return codes, are used to indicate a job failure, a careful look at the jobs will demonstrate that most of the abends were forced, either by programs that called an Assembler routine that purposely abends or by conditional job steps that execute an Assembler program whose purpose is to cause an abend. A look at the conditional (COND= or IF) job step often proves it to be triggered by a non-zero return code from a previous step.

So, from TSO or ISPF, how does REXX set the condition code for the job step? Without ISPF, it is as simple as specifying a value in the exit statement. Exit 12 would immediately end the job step with a condition code of 12. This will not work, however, with ISPF. The exit statement does terminate the job step, but as control passes through ISPF, it ignores the return code from REXX and sets its own zero return code.

Actually, ISPF does look for a return code to pass as a job step condition code, but it looks in a variable in the shared variable pool called ZISPFRC with a default value of zero. If the desired return code is in the REXX variable rc, the following code will do the equivalent of the *exit rc* that would be used if ISPF were not being used.

```
ZISPFRC = rc"ISPEXEC VPUT ZISPFRC SHARED"exit ZISPFRC
```

CONCLUSIONS

Running TSO and ISPF in batch offers a lot of functionality not otherwise available. Even some IBM products do not offer a direct command interface in batch, but rely instead on using commands they have provided for TSO.

The downside of this approach is that these on-line functions were not intended for the kind of high-volume processing often required in production batch jobs. Conflicts between simultaneous on-line and batch usage of ISPF under the same user-id also need to be considered. But the available functionality can save significant development and maintenance costs for batch applications, especially if the alternative requires the use of Assembler.

A customized SMP/E PTF status report

THE PROBLEM

In common with most large MVS shops, we have a sizeable portfolio of software that is installed using IBM's SMP/E program. Whilst SMP/E has many advantages as a method of installing and maintaining large and complex collections of system software, it does lack the inclusion of a flexible reporting language. One problem we frequently encounter is the question: "Has all applied maintenance been accepted?" This usually occurs just prior to applying a new batch of cumulative maintenance!

Although it is easy enough to submit a batch job to run the SMP/E LIST ALLZONES PTFS command against the global CSI for the software, it can produce a sizeable report, especially for large software such as DB2. Also the report is extremely verbose, and analysing it to identify the status of each PTF can be time consuming.

A SOLUTION

I therefore wrote the following REXX routine to process the output from a LIST ALLZONES PTFS report and reformat it into a more readable format. The main highlights of the routine are:

- It produces a simple table format report, one line per PTF, allowing easy identification of its status, RECEIVED, APPLIED, or ACCEPTED. Superseding PTFS are also identified, and the superseded PTFS are marked accordingly.
- It supports multiple TLIB zones, should your site utilize multiple SYSRES volumes and indirect cataloguing.
- The main REXX variables can be easily overridden in the batch job from a SYSIN DD statement (SMPECNTL DD *).
- Non-accepted PTFS are highlighted in the report, and the correct SMPE control statements are generated to an SMPEPUN DD, ready to ACCEPT the PTFS if you wish.

Whilst executing, messages are issued to the SYSTSPRT DD statement in the IRXJCL step to indicate the progress of the report generation.

This routine was written and tested under OS/390 1.3.0 and assumes the LIST report will be in that format. However, it should be relatively easy to modify the REXX for differing output format from other SMP/E releases as the REXX is fully commented.

```
SMPEANAL BATCH JCL
//jobname JOB your-jobcard-here
//SMPELIST EXEC PGM=GIMSMP,
//         PARM='PROCESS=WAIT',REGION=4M,
//         DYNAMNBR=12Ø
//SMPCSI   DD DISP=SHR,DSN=your.global.csi
//SMPHOLD  DD DUMMY,DCB=BLKSIZE=8Ø
//SMPRPT   DD DUMMY
//SMPOUT   DD DUMMY
//SMPLOG   DD DUMMY
//SMPLIST  DD DSN=&&SMPLIST,
//         DISP=(,PASS),
//         UNIT=SYSDA,
//         RECFM=FBA,
//         LRECL=121,
//         BLKSIZE=27951,
//         SPACE=(CYL,(1ØØ,1Ø))
//SMPCNTL  DD *
  SET    BOUNDARY (GLOBAL) .
  LIST ALLZONES PTFS .
/*
//ANALYZE  EXEC PGM=IRXJCL,PARM='SMPEANAL'
//SMPEIN   DD DSN=&&SMPLIST,DISP=(OLD,DELETE,DELETE)
//SMPEOUT  DD SYSOUT=*,DCB=RECFM=FBA
//SMPECNTL DD *
TARGET='TGTA TGTB TGTP' /* list of target zones */
DLIB='DLIB'             /* distribution zone */
/*
//* SMPEPUN IS THE OUTPUT DD FOR GENERATED ACCEPT SELECT STATEMENTS
//* YOU CAN USE ANY STANDARD RECFM=FB, LRECL=8Ø PDS LIBRARY
//*
//SMPEPUN  DD DSN=output.pds(SMPEPUN),DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSEXEC  DD DSN=your.REXX.library,DISP=SHR
//*
```

SMPEANAL REXX ROUTINE

```
/*REXX
  Member : SMPEANAL
  Function : Analyses SMPE LIST output and produces formatted
             report of PTF status (Received/applied/Accepted)
  Called by : Batch IRXJCL (all files should be allocated in JCL)
*/
say 'Analysis started at 'date(e) time()
call init
call read_input
call analyse_file
call write_report
fin:
say 'Analysis ended at 'date(e) time()
exit
/**/
read_input:
Say 'Reading Input File ...'
"EXECIO * DISKR SMPEIN (STEM SMPE. FINIS"
/* SMP/E LIST ALLZONES OUTPUT REPORT */
return
/**/
analyse_file:
Say 'Analysing File ...'
Do cnt = 1 to smpe.Ø                /* process stored report lines */
  if cnt//interval = Ø then         /* message interval reached?   */
    Say cnt 'of' smpe.Ø 'entries processed...'/* output status message*/
  if left(smpe.cnt,1) = '1' then iterate /* skip page titles        */
  if strip(smpe.cnt) = '' then iterate   /* skip empty lines        */
  if word(smpe.cnt,1) = 'NAME' then iterate /* skip header lines    */
  if word(smpe.cnt,1) = 'LIST' then iterate /*  "    "      "       */
  if word(smpe.cnt,1) = 'ENTRY-TYPE' then iterate /* "      "       */
  if word(smpe.cnt,1) = 'SYSMOD' then iterate     /* "       "      */
  if word(smpe.cnt,3) = 'ENTRIES' then  /* zone name line */
  do
    currzone = word(smpe.cnt,1) /* store current zone name */
    iterate
  end
  parse var smpe.cnt 2 ptfid 9 . 12 kwd 3Ø val  /* split up line    */
  if strip(ptfid) ¬= '' then /* SYSMOD (ptf) name specified?        */
  do
    ptf = ptfid                                 /* save PTF name    */
 /* check if I have already seen this PTF before. If not, add it to
    the list of processed PTFS */
    if wordpos(ptf,ptflist) = Ø then ptflist = ptflist ptf
  end
 /* now set indicator variable, depending on which zone list if am
currently processing */
  select
```

```
    when currzone = global then
      rec.ptf = 'YES' /* RECEIVED */
    when wordpos(currzone,target) ¬= Ø then /* valid target zone name?*/
      app.ptf = app.ptf currzone /* add to APPLIED list */
    when currzone = dlib then
      acc.ptf = 'YES' /* ACCEPTED */
    otherwise nop
  end
  if word(kwd,1) = 'SUPING' then /* does this PTF SUPERCEDE any
others?*/
  do sup = 1 to words(val)
    supptf = word(val,sup) /* get superceded name */
    if wordpos(supptf,ptflist) = Ø then iterate sup /* one of mine? */
    sup.supptf = ptf /* YES - mark supreceded PTF as such */
  end
end
return
/**/
write_report:
say 'Generating Report ...'
call header  /* output Page title headers */
do num = 1 to words(ptflist) /* go through stored PTF list */
  if counter = 3Ø then /* end of page (6Ø lines - 2 per PTF */
    call header /* start next page */
  newline = outline /* initialise output line */
  ptf = word(ptflist,num) /* extract ptf name */
/* the following if test if to catch PTFS that are not in any of the
defined zone names. If so, then all variables will be null  */
  if rec.ptf = '' ,
   & app.ptf = '' ,
   & acc.ptf = '' then iterate
  newline = overlay(ptf,newline,posptf) /* output PTF name */
  newline = overlay(rec.ptf,newline,posrec) /* RECEIVED - YES */
  do appcnt = 1 to words(app.ptf) /* for each TLIB where APPLIED*/
    appzone = word(app.ptf,appcnt) /* get TLIB name */
    newline = overlay('YES',newline,pos.appzone)
/* and mark APPLIED-YES*/
  end
  newline = overlay(acc.ptf,newline,posdlib) /* ACCEPTED - YES */
  if sup.ptf ¬= '' then /* was it superceded ? */
/* if superceded, mark all positions where 'YES' with 'SUP' and place
superceding PTF name in output line */
  do
    if rec.ptf = 'YES' then
      newline = overlay('SUP',newline,posrec)

    if app.ptf ¬= '' then
    do appcnt = 1 to words(app.ptf)
      appzone = word(app.ptf,appcnt)
      newline = overlay('SUP',newline,pos.appzone)
```

```
    end
    if acc.ptf = 'YES' then
      newline = overlay('SUP',newline,posdlib)
  newline = overlay(sup.ptf,newline,possup)
  end
/*
  If PTF marked as applied, but not accepted, overlay warning message on
line, and add PTF name to non-accepted list
*/
  if app.ptf ¬= '' & acc.ptf = '' then
  do
    newline = overlay(warning,newline,possup+10)
    nonacc = nonacc ptf
  end
  queue newline /* output generated line */
  queue sep /* ... and separator line */
  "EXECIO "queued()" DISKW SMPEOUT" /* and write to SYSOUT */
  counter = counter + 1 /* increment counter for pagination */
end
"EXECIO 0 DISKW SMPEOUT (FINIS" /* close SYSOUT file */
/*
   If any PTFS not accepted, generate SMPE ACCEPT cards and write them
   to SMPEPUN DD statement
*/
if nonacc ¬= '' then
do
  say ' Generating SMPE control cards for non-accepted ptfs ...'
  queue ' SET BDY('dlib').'
  queue ' ACCEPT CHECK SELECT('
  do cnt = 1 to words(nonacc)
    queue '                     '||word(nonacc,cnt)
  end
  queue ' ).'
end
"EXECIO "queued()" DISKW SMPEPUN (FINIS"
return
/*
  initialization section - all of the first group of variables can be
overridden with statements in the SMPECNTL DD statement in the batch
JCL. Statement must be valid REXX syntax
*/
init:
ptflist = ''
rec. = ''
app. = ''
acc. = ''
sup. = ''
nonacc = ''
interval = 10**(length(smpe.0)-1)
/* default status message update interval
```

```
 (# of lines). 1-9 lines = 1, 10-99 lines = 10, 100-999 lines = 100 etc
*/
global = 'GLOBAL'
target = 'MVST100'
dlib = 'MVSD100'
warning = '** WARNING! UNACCEPTED SERVICE! **'
/*
  Now read in SMPECNTL cards and interpret each one as a REXX statement
  to allow values such as target and distribution zone names to be
easily overridden from the JCL
*/
"EXECIO * DISKR SMPECNTL (STEM CNTL. FINIS"
do cnt = 1 to cntl.0
  interpret cntl.cnt
end
/*
  no calculate total output report line length as follows:-
  - for 'PTF NAME':
        9 columns for name
  - for 'RECEIVED':
        1 column for separator
        8 columns for name
  - for each target zone:
        1 column for separator
        8 columns for name
  - for distribution zone:
        1 column for separator
        8 columns for name
  - for superceded:
        1 column for separator
        8 columns for name
*/
line_length = 18+(9*words(target))+19
psep = '1+'||copies('-',line_length)||'+'
sep  = ' +----+----'||,
     copies('+----',(words(target)+2))||'+'
outline = ' !         !        '||,
     copies('!        ',(words(target)+2))||'!'
title = ' +'||center('PTF REPORT' date(e) time(),line_length,'-')||'+'
/*
  set relative line positions for each column value
*/
posptf = 3                              /* start position - PTF name */
posrec = posptf + 10                      /* RECEIVED status         */
cpos = posrec + 9                         /* first APPLIED status    */
do cnt = 1 to words(target)             /* repeat for each target zone */
  tgtnam = word(target,cnt)               /* get name                */
  pos.tgtnam = cpos                       /* set column position     */
  cpos = cpos + 9                         /* advance for next column */
end
```

```
posdlib = cpos                              /* ACCEPTED status         */
possup = cpos+9                     /* superceding PTF name position */
/*
  build column titles
*/
colhead = overlay('PTF NAME',outline,posptf)
colhead = overlay('RECEIVED',outline,posrec)
do cnt = 1 to words(target)
  tgtnam = word(target,cnt)
  colhead = overlay(tgtnam,colhead,pos.tgtnam)
end
colhead = overlay(dlib,colhead,posdlib)
colhead = overlay('SUPER',colhead,possup)
return
/*
  output header lines at start of each page
*/
header:
queue psep                                  /* page separator line   */
queue title                                 /* report title          */
queue sep                                   /* row separator         */
queue colhead                               /* column titles         */
queue sep                                   /* row separator         */
"EXECIO "queued()" DISKW SMPEOUT"           /* and write them out    */
counter = 0                         /* initialize pagination counter */
return
```

MVS news

IBM has made a number of significant changes to its System/390 software pricing policy, focusing on usage-based pricing, and phasing out some of the linear pricing models that have deterred users from implementing new applications on the mainframe platform.

The most significant pricing introduction is System/390 Usage Pricing (ULC) for DB2, IMS, CICS Transaction Server, MQSeries, and related products. It is consistent with PSLC pricing, and is calculated on MSUs, but it is aimed specifically at users whose peak utilization of a product is lower than 25% of total system usage.

Parallel Sysplex Level C Pricing, announced earlier this year as a way of offering a reduction in price for PSLC users at the other end of the scale (those with aggregated capacity of over 175MSUs), now covers over 80 products.

IBM has also brought in a System/390 New Application Growth Environment, which offers price reductions on OS/390 and DB2 for dedicated systems running a 'new' application, such as SAP R/3, Lotus Domino, Business Intelligence, or Net.Commerce. This is a further attempt to make the System/390 software more attractive as a low-end server, for SME users or for larger users with new applications. IBM plans to withdraw a number of older schemes, including GMLC, DSLO, and OTC, replacing them all with a combination of ULC and PSLC.

Contact your local IBM representative for further information.

* * *

BMC has announced the first products in its Enterprise Data Propagation management strategy for integrating applications and enabling near real-time data warehousing.

First, ChangeDataMove provides a transaction-based change capture for DB2 for OS/390, IMS, CICS, VSAM, and VSAM batch applications with near real-time or scheduled propagation of the captured changes to DB2 for OS/390 and to Oracle on Unix.

The product is designed to support operational applications executing hundreds of transactions per second. For IMS and VSAM, the product doesn't use logs or require any changes to an existing application's logging strategy to capture changes. It can also capture changes for batch applications with no logging.

DataReach Version 2.1, meanwhile, is a joint development with EMC, and is for moving DB2 for OS/390 data to Oracle, Sybase, and Informix databases on Unix systems. While executing on a Unix host, it uses EMC Symmetrix storage to access MVS DB2 data directly.

BMC Software, Inc, 2101 Citywest Blvd, Houston, TX 77042-2827, USA.
Tel: (713) 918 8800
Fax: (713) 918 8000 or
BMC Software Ltd, Compass House, 207-215 London Road, Camberley, Surrey, GU15 3E, UK.
Tel: (01276) 24622
Fax: (01276) 61201