218

MVS update

November 2004

In this issue

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

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

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Nicole Thomas
E-mail: nicole@xephon.com

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs $505.00 in the USA and Canada; £340.00 in the UK; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 2000 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

MVS Update on-line

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

Contributions

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

This issue is dedicated to the memory of Chris Bunyan, co-founder of Xephon and creator of the *Update* journals.

Printed in England.

A REXX EXEC for the systems programmer's toolkit

The following REXX EXEC can be used from an ISPF 3.4 dataset list. It will dynamically add a dataset to the APFLIST. You will need access to the TSO console command and the MVS SETPROG command.

```
/* Rexx */
arg dsn
if dsn = '' then
   exit
if pos('(',dsn) ¬= Ø then
   do
      say 'incorrect usage'
      exit
   end
x = listdsi(dsn smsinfo)
if (sysrecfm ¬= 'U' | sysdsorg ¬= 'PO') then
   do
      say 'not a load library'
      exit
   end
if sysstorclass = '' then
   vol = 'VOL='sysvolume
else
   vol = 'SMS'
command = 'SETPROG APF,ADD,DSN='substr(dsn,2,length(dsn)-2)','vol
"consprof soldisp(no) unsoldisp(no)"
"console activate name(apfadd)"
address CONSOLE "cart APF1"
address CONSOLE command
getcode = getmsg('response.','SOL','APF1',,1Ø)
if ¬datatype(response.Ø,'N') then
   say '..... no response to 'command
else
   do i = 1 to response.Ø
      say response.i
   end
"console deactivate"
exit Ø
```

Deleting a console device from HCD

THE PROBLEM

Recently, I had to delete a 3174 definition from my HCD deck. In the 3270 devices managed by this 3174 there was an MVS console that was replaced by a 2074 device.

CONSOL00 shows that device 0845 was defined as an MVS console:

```
CONSOLE DEVNUM(Ø845) ROUTCODE(ALL)
        RBUF(1Ø) PFKTAB(PFKTAØØ)
        NAME(MASTØ845)
        AUTH(ALL)
        UNIT(327Ø-X)
        MONITOR(JOBNAMES-T)
        UD(Y)
        CON(N) SEG(16) DEL(R) RNUM(19) RTME(1) MFORM(J) AREA(NONE)
```

After deleting the 3174 and its 3270 devices from my IODF, I tried to activate the new configuration. I got the following messages:

```
IOS5ØØI ACTIVATE RESULTS 997
ACTIVATE FAILED - ERROR MESSAGE(S) ISSUED
NOTE = A886,FOLLOWING DEVICES ARE TO BE DELETED FROM
PROCESSOR
            CPCØCØ2: Ø.Ø84Ø-Ø.Ø85F
 COMPID=SC1XL
NOTE = A883,FOLLOWING CONTROL UNITS ARE TO BE DELETED FROM
PROCESSOR
            CPCØCØ2: Ø.Ø8Ø8
 COMPID=SC1XL
REASON=Ø151,CAN NOT DELETE DEVICE Ø845
 DESCTEXT=DEVICE PINNED, ASID = ØØØ1
          DEVICE IN USE AS A CONSOLE
 COMPID=SC1CK
```

The HCD activation fails because of pinned devices!

The reason is that, at IPL time, the system pins UCBs for console devices defined in CONSOL*xx*.

The UCBs can be unpinned only with the console removal

definition service IEAVG730 (information about IEAVG730 (return codes) can be found in *z/OS Planning: Operations*).

In order to be able to activate my new HCD configuration, I wrote a little utility program, using the IEAVG730 service, to dynamically 'delete' the console definition.

CONUNPSO UTILITY

```
CONUNPSO CSECT
CONUNPSO AMODE 31
CONUNPSO RMODE ANY
*
         SAVE  (14,12)
         BASR  R12,Ø
         USING *,R12               R12 = BASE REGISTER
*
         L     R2,Ø(R1)            LOAD PARAMETER ADDRESS
*
         GETMAIN R,LV=WORKL
*
         ST    R1,8(R13)
         ST    R13,4(R1)
         LR    R13,R1
         USING WORK,R13
***************************
** COPY PARMS   ***********
***************************
         SR    R3,R3
         LH    R3,Ø(R2)            LENGTH
*
         CH    R3,=H'Ø'            PARM LENGTH = Ø ?
         BE    MSG1
*
         MVI   CONSNAME,C' '       CLEAR RECEIVING AREA
         MVC   CONSNAME+1(L'CONSNAME),CONSNAME
*
         BCTR  R3,Ø                SUBTRACT 1 FROM LENGTH FOR MOVE
         EX    R3,MOVE             MOVE COMMAND FROM PARAMETER LIST
*
         AUTHON                    AUTH SVC
         MODESET KEY=ZERO,MODE=SUP
*
         LA    R2,CONSNAME
         ST    R2,POINTER
         LA    R1,POINTER
         LINK EP=IEAVG73Ø
*
```

```
         ST    R15,RC               RETURN CODE
         ST    RØ,RSN               REASON CODE
*
         MODESET KEY=NZERO,MODE=PROB
         AUTHOFF                    AUTH SVC
*
         L     R15,RC
         LTR   R15,R15
         BNZ   MSG2
*
         B     MSG3
*
RETURN   L     R13,4(R13)               RESTORE R13
         L     R1,8(R13)
         FREEMAIN R,LV=WORKL,A=(R1)
         L     R14,12(R13)
         LM    RØ,R12,2Ø(R13)
         SR    R15,R15                  SET UP RC
         BSM   Ø,R14                    RETURN TO MVS AND USE RC=R15
*
MSG1     EQU   *
         MVC   WTOA(WTOL),WTOLIST
         MVC   WTOM,=CL8Ø'MISSING CONSNAME PARAMETER'
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
         WTO   TEXT=(R2),MF=(E,WTOLIST)
*
         B RETURN
*
MSG2     EQU   *
         MVC   WTOA(WTOL),WTOLIST
         MVC   WTOM,=CL8Ø'CONSOLE 12345678 WAS NOT DELETED. IEAVG73Ø RCX
               =1234 RSN=1234'
*
RCOFF    EQU   WTOM+46
RSNOFF   EQU   WTOM+55
*
         UNPK  RCOFF(5),RC+2(3)
         TR    RCOFF(4),HEXTAB
*
         UNPK  RSNOFF(5),RSN+2(3)
         TR    RSNOFF(4),HEXTAB
*
         MVC   WTOM+Ø8(Ø8),CONSNAME
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
         WTO   TEXT=(R2),MF=(E,WTOLIST)
*
```

```
         B RETURN
*
MSG3     EQU   *
         MVC   WTOA(WTOL),WTOLIST
         MVC   WTOM,=CL8Ø'CONSOLE 12345678 HAS BEEN SUCCESSFULLY DELETEX
               D'
         MVC   WTOM+Ø8(Ø8),CONSNAME
         L     R2,=A(WTOMLEN)
         STH   R2,WTOML
         LA    R2,WTOMSG
         WTO   TEXT=(R2),MF=(E,WTOLIST)
*
         B RETURN
*
MOVE     MVC   CONSNAME(Ø),2(R2)    MOVE PARM (OFFSET 2 / LENGTH)
*
HEXTAB   EQU   *-24Ø
         DC    C'Ø123456789ABCDEF'
*
WTOLIST  WTO   TEXT=,ROUTCDE=11,MF=L
WTOL     EQU   *-WTOLIST
*
WORK     DSECT
SAVEAREA DS    18F
CONSNAME DS    CL8
POINTER  DS    F
RC       DS    F
RSN      DS    F
*
WTOA     DS    CL(WTOL)
*
WTOMSG   DS    ØF
WTOML    DS    H
WTOM     DS    CL8Ø
WTOMLEN  EQU   *-WTOM
*
WORKL    EQU   *-WORK
*
         REGISTER
*
         END
```

The IEAVG730 service needs to receive control in supervisor state and key zero.

In my source, I use an authorization SVC (macros AUTHON and AUTHOFF) to get authorized. If you do not want to use that kind of SVC, you can remove these two macro calls and link-edit the module with AC=1 in an APF library.

USAGE

The JCL to start CONUNPSO is:

```
//STEP1    EXEC PGM=CONUNPSO,PARM='MAST0845'
```

CONUNPSO SYSOUT looks like:

```
 $HASP373 SXSP001B STARTED - WLM INIT  - SRVCLASS JOB_60   - SYS PROD
 IEF403I SXSP001B - STARTED - TIME=15.32.01
 +CONSOLE MAST0845 HAS BEEN SUCCESSFULLY DELETED
 -                                          --TIMINGS (MINW.)--
 -JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB  ELAPS   SERV
 -SXSP001B STEP1                00      6    .00    .00     .0   1005
 IEF404I SXSP001B - ENDED - TIME=15.32.02
 -SXSP001B ENDED.  NAME-SYSTEM TEAM          TOTAL CPU TIME=   .00
TOTAL
 $HASP395 SXSP001B ENDED
```

Alexandre Goupil
Systems Programmer (France)

Getting started with Linux for S/390 (zSeries)

When the Linux operating system is first appeared in 1991, it was used on IBM PC compatibles, Apple, Atari, and 68000-based Amiga computers. It also worked on Sun Spark workstations, Alpha-based personal computers, and MIPS, PowerPC, HP PA-RISC and ARM architectures.

The S/390 architecture has been widely used with IBM's VM, VSE, and z/OS (formerly MVS and OS/390) operating systems. IBM has chosen to use Linux as one of the 'native' operating systems for this architecture since 1999.

It is widely said that Linux works as an API or emulation on S/390 platforms, but it is not true – it works as a 'native' operating system so that all the hardware capability of this platform is used. The Linux kernel and common code are used without any modification, and the Linux system structure remains untouched. Only some 'adaptations' are required to match

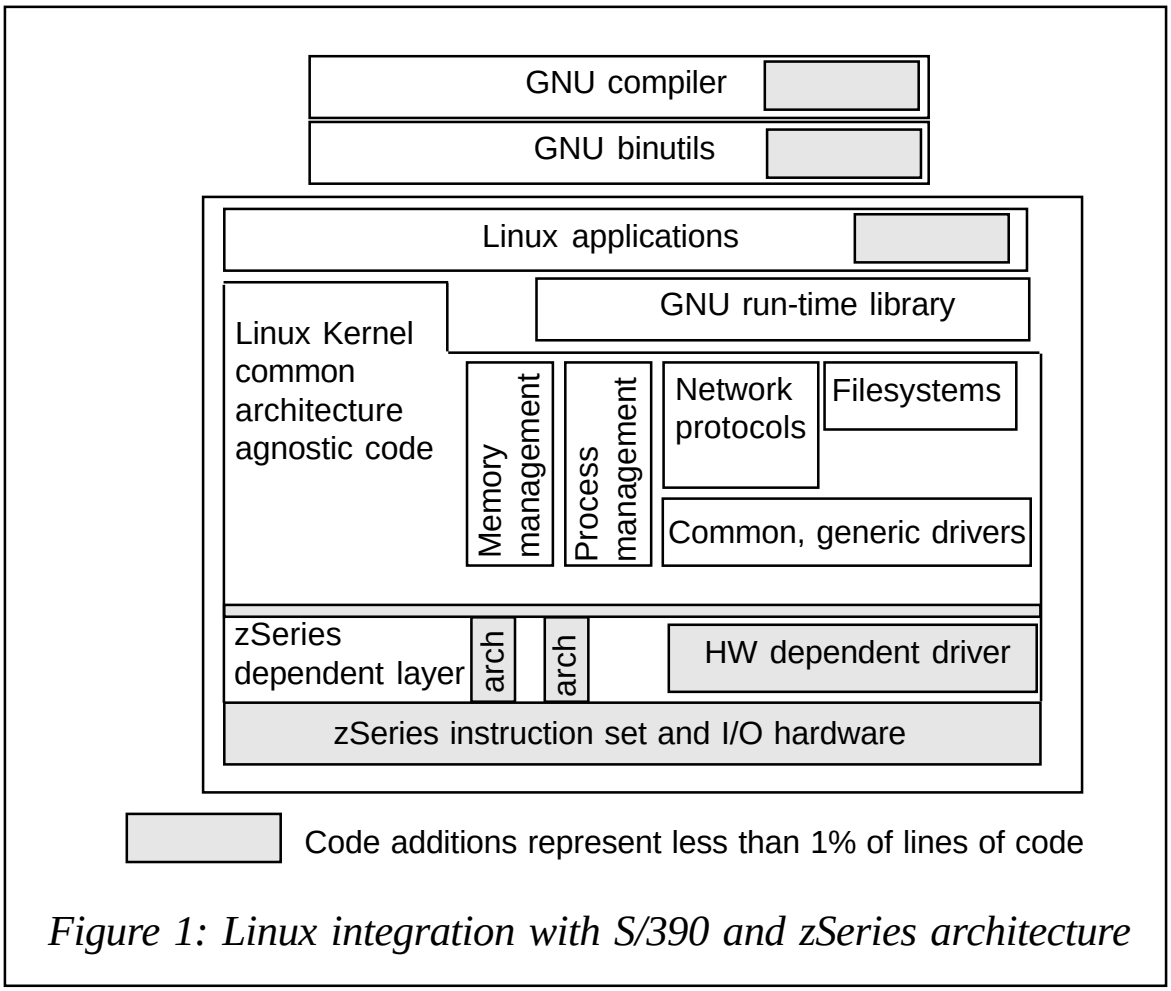

Figure 1: Linux integration with S/390 and zSeries architecture

and implement S/390 architecture. It works with ASCII character set instead of EBCIDIC. Figure 1 shows Linux integration with S/390 and zSeries architecture.

Linux can be installed in three different ways on S/390 platforms:

- *Native mode* – this is installed directly on the system hardware. It is not usually the preferred solution because only one operating system runs on the hardware.
- *Logical Partitions (LPARs)* – hardware partitioning enables up to 15 'logical partitions', each running a separate operating system (including traditional ones such as MVS, VSE, OS/390, and Linux).
- *Virtual partitions (z/VM)* – this is called z/Series

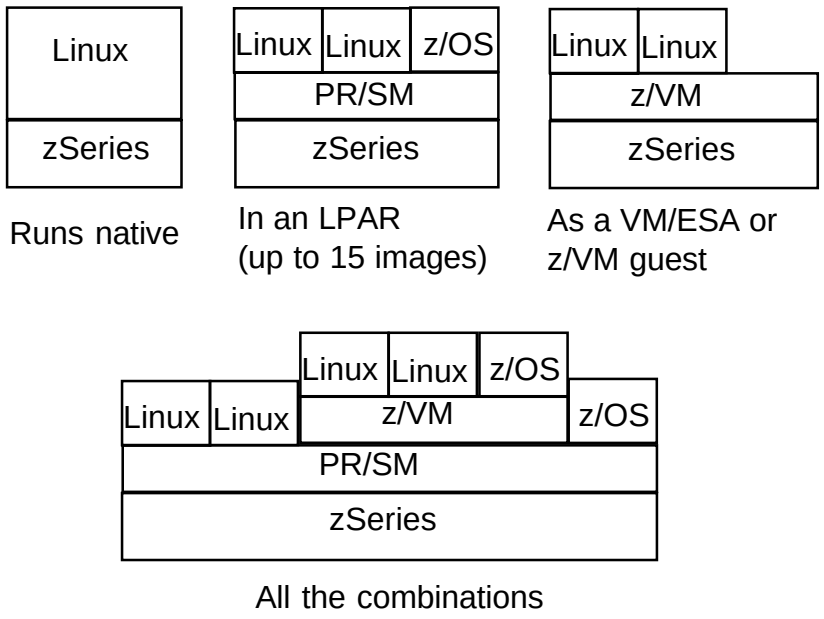

Figure 2: Three types of installation

virtualization technology. It supports large numbers of Linux images (1,000+) with rich system management capabilities in the same hardware. This installation approach is very flexible and ideal for server consolidation.

Figure 2 shows the three types of installation.

If the required number of Linux servers is 15 or below, the LPAR solution is a good choice. If you need more, eg 100 or 1,000 Linux images, z/VM is the answer.

RedHat, SuSE, and Turbolinux are major distributors for S/390 and zSeries.

You can use the links below to download them:

- RedHat:
  - *RedHat Linux 7.2 for S/390* – www.redhat.com/software/linux/s390.
  - *RedHat Linux 7.1 for z/Series* – www.redhat.com/software/linux/ibmseries/z.html.

- SuSE:
  - *SuSE Linux Enterprise Server 7 for S/390 and zSeries* – www.suse.com/us/business/products/server/sles/s390.html.
- TurboLinux:
  - *TurboLinux server 6.5 for zSeries and S/390* – www.turbolinux.com/products/s390.

There are also some distributions in binary. You can get them from the links below:

- http://www.millenux.de – Millenux Think Blue distributes Red Hat-based binary 31-bit Linux for S/390 and 64-bit Linux for z/Series.
- Linux390.marist.edu – Marist College has used Linux for S/390 since January 2000.

Distributions for S/390 and zSeries are summarized in Figure 3.

The requirements for running Linux for S/390 are:

Distribution	*Linux kernel*	*Addressing mode*
SuSE 7.0	2.2.16	31-bit
SuSE Linux Enterprise Server 7	2.4.7	31-bit
SuSE Linux Enterprise Server 7	2.4.17	64-bit
SuSE Linux Enterprise Server 8	2.4.19	31 and 64-bit
TurboLinux 6.0	2.2.16	31-bit
TurboLinux 6.5.1	2.2.19	31-bit
TurboLinux	2.4.5	on customer request
TurboLinux	2.4.7	on customer request
RedHat 7.2	2.4.9	31-bit
RedHat 7.1	2.4.9	64-bit
Caiman 1.0 (Linux Korea)	2.2.16	31-bit
Marist	2.2.16	31-bit
Debian ‘True GNU Linux distribution’	2.4.17	31-bit
Think Blue from Millenux	2.4.7	64-bit

Figure 3: Distributions for S/390 and zSeries

- 9672 G5/G6, Multirise 3000, or z/Series 800, 900, 990 IBM processors.
- 64MB+ memory (absolute minimum – distributions and applications dependent).
- 500+ cylinders of disk space (Model 3390 – for a small minimal system).
- IBM Network Device Support (one required) Ethernet, Token Ring, Fast Ethernet, ESCON, OSA, or HiperSocket. More devices are supported.
- Before Linux can use a device, the associated driver for zSeries and S/390 must be available to the kernel.
- There are kernel-resident drivers and external drivers for S/390 and zSeries devices.
- External drivers are modules loaded on request with their parameters by means of commands.
- Resident drivers receive their parameters at boot time, from a kernel parameter line, held in a file.
- Non-open source OCO (Object Code Only) drivers are drivers subject to licence conditions (ie QETH for OSA Express GbE and Hipersocket, Tape 3590). OCO drivers may not be provided with all distributions and need to be downloaded from IBM Developer Works when not in the distribution.

WHY LINUX FOR S/390?

The most important reason for using Linux on S/390 is server consolidation. Legacy applications have jumped to distributed applications and then to Web-based applications. First data then applications were distributed everywhere. The number of servers increased enormously. This increase brought some problems:

- Each and every new server means new hardware and

capacity, and an increase in cooling, cabling, and connection etc. So, every time, those 'physical' parameters have to be monitored and fixed.

- All software has to be licensed in each server, which means additional cost. For instance, your favourite database has to be licensed on each server on a CPU basis.
- Connectivity is another important issue. Cabling, gateways, switches, routers, all increase the total cost and affect network performance.
- A Disaster Recovery (DR) solution is almost impossible with individual servers. Operation and maintenance costs of DR get higher, more complicated, and unworkable with a huge number of servers.
- Database/application/system management issues. CPU, DASD and workload sharing have to be done individually, for each and every server.

Those are some of the potential problems where Linux images are run on different hardware. If all of the images are run on a single S/390 platform:

- Although all Linux images share the same hardware (CPU, DASD, I/O subsystem, memory, etc) they behave like individual, mutually exclusive, logical servers, and can be used for different applications. In this way, the increase in the number of servers does not affect the maintenance cost. It can be monitored and controlled easily and decreases in time. By sharing these resources, the throughput of the system is maximized.
- All servers share the same CPU, so that software licence fees decrease.
- All the connections between servers are internal, so the communication overhead is almost nothing and network performance is maximized.

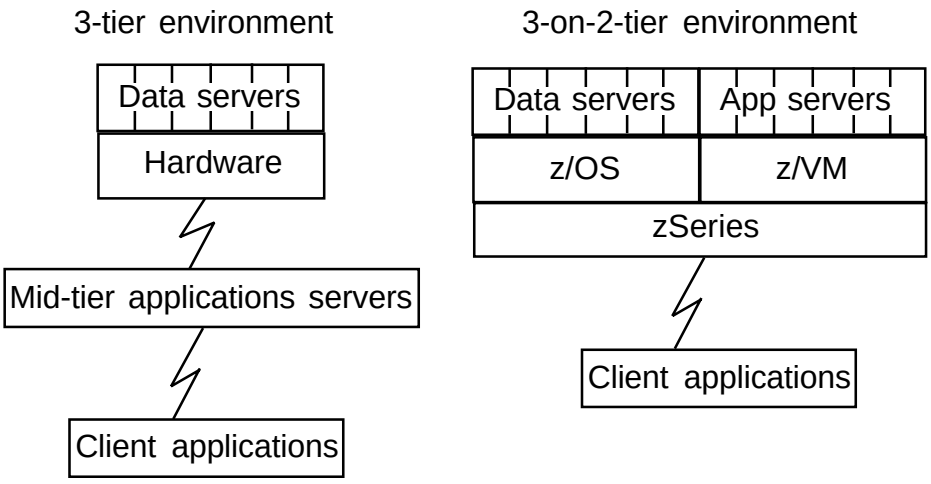

Figure 4: 3-tier architecture on 2-tier hardware

- Adding a new server is easy – simply clone a logical server.
- Disaster recovery is much easier. DASD farms and subsystems can be easily, quickly, and safely copied in minutes with FlashCopy, PPRC (Peer-To-Peer-Remote-Copy), or Snapshot features. With the advantage of 'single hardware', cloned copies can easily be loaded at a DR site.

Three-tiered application architecture can be easily realized in two-tiered hardware. Client/application server/data server can be combined in S/390 like application servers/databases – see Figure 4. Hipersocket Fibre channel features support the communication subsystem features, and connection problems disappear.

RESOURCES

- L*inux for S/390*, IBM Redbook
- *Linux for z/Series*, Atruro Calandrino, zSeries Tech Support.

Mehmet Cuneyt Goksu
DB2 Specialist (Turkey)

Displaying TSO attribute list information

One of the things that have bothered me with TSO for years concerns the TSO ATTRIB command. The ATTRIB command allows DCB parameters to be dynamically introduced and named for use with subsequent TSO ALLOCATE commands. Attribute lists are referenced in the ALLOCATE command by means of the USING keyword, and they can be freed by means of the ATTRLIST keyword of the TSO FREE command. IBM, however, does not provide any way to actually list any information about currently built attribute lists. An acquaintance of mine actually had an Assembler program that was written years ago to display the attribute lists. That program stopped working once MVS started allowing the movement of its Scheduler Work Area (SWA) control blocks to reside above the 16-megabyte line, which is where one of the control blocks used by the Assembler program (detailed later) is built. Using his program as a model, and a public domain REXX function written by a very knowledgeable MVS systems programmer named Gilbert Saint-flour, I decided to write a REXX EXEC to do the work of the defunct Assembler program.

The result is a REXX EXEC called ATTRLIST. The EXEC scans the entries in the Task I/O Table (TIOT) looking for the special attributes that differentiate attribute lists from normal allocations, and then it decodes and formats the information contained in the attribute lists.

It turns out that there are three basic fields that differentiate an attribute list from other allocations. These are:

- The Job File Control Block (JFCB) field called JFCBDSNM, which contains a value of NULLFILE (the JFCB is pointed to by the TIOT).
- The TIOT TIOELINK field, which has the X'20' bit turned on.
- The TIOT TIOEDDNM field, which contains a valid DDNAME value, as per MVS conventions.

The EXEC formats every field that is specifiable on the ATTRIB command with one exception. If a retention period (RETPD) keyword was coded on the ATTRIB command, such retention periods are translated to and stored as expiration dates within the JFCB control block. The EXEC therefore can display only the translated expiration date, if one was specified on the original ATTRIB command. Since the attribute lists built by the ATTRIB command contain only the values for the keywords actually used on the ATTRIB command, the REXX EXEC will therefore display only those values. That is to say, no default values are put into an attribute list for any keywords not coded on the original ATTRIB command – the attribute lists contain only values for keywords that were actually coded on the ATTRIB command.

The SWAREQ REXX function, written by Gilbert Saint-flour, takes an SWA token from the TIOEJFCB field and translates it to a virtual address. This virtual address points to the JFCB control block. This translation is required, in the JES2 environment, when the SWA=ABOVE parameter is coded on the JOBCLASS statements in the JES2 initialization deck. Most modern installations use the SWA=ABOVE parameter, which causes the SWA control blocks to be built above the 16-megabyte line.

Here is the code for the ATTRLIST REXX EXEC:

```
/*                                rexx comment *** start standard header
ATTRLIST Show attribute list information
                                  rexx comment *** end   standard header */
numeric digits 10                              /* allow up to 7fffffff   */
tiotptr = 24 + ptr(12 + ptr(ptr(ptr(16))))     /* get DDname array       */
tioelngh = c2d(stg(tiotptr,1))                 /* length of 1st entry    */
a = 0
nullfile = left('NULLFILE',44)
ddname = ' '
full0 = '00000000'x
byte0 = '00'x
do until tioelngh = 0                          /* scan until dd found    */
   tioeddnm = stg(tiotptr + 4,8)               /* get DDname from tiot   */
   tioelink = stg(tiotptr + 3,1)               /* get flag byte          */
   tioelngh = c2d(stg(tiotptr,1))              /* length of next entry   */
   tioejfcb = stg(tiotptr + 12,3)
   jfcb = swareq(tioejfcb)                     /* cnvt sva to 31-bit addr*/
```

```
jfcbdsnm = strip(stg(jfcb,44))           /* dsname jfcbdsnm        */
jfcbdsnm = stg(jfcb,44)                 /* dsname jfcbdsnm        */
tioeddn1 = substr(tioeddnm,1,1)
if tioeddn1 <> byteØ   & tioeddn1 <> " " &,
   jfcbdsnm = nullfile & bitand(tioelink,'2Ø'x) <> '2Ø'x then nop
else do
   tiotptr = tiotptr + tioelngh         /* get next entry         */
   tioelngh = c2d(stg(tiotptr,1))       /* get entry length       */
   iterate
   end
a = a + 1
dsname.a = jfcbdsnm
ddname.a = tioeddnm
volume.a = storage(d2x(jfcb+118),6)   /*volser jfcbvols-not used*/
bfaln   = stg(jfcb + 95,1)            /* and bftek */
blksize = stg(jfcb + 1Ø2,2)
bufl    = stg(jfcb + 9Ø,2)
bufno   = stg(jfcb + 88,1)
bufoff  = stg(jfcb + 67,1)
den     = stg(jfcb + 94,1)
flag2   = stg(jfcb + 78,1)            /* diags, i/o */
dsorg   = stg(jfcb + 98,2)
dsorg1  = stg(jfcb + 98,1)
dsorg2  = stg(jfcb + 99,1)
eropt   = stg(jfcb + 92,1)
expdt   = stg(jfcb + 83,3)
keylen  = stg(jfcb + 93,1)            /* and trtch */
limct   = stg(jfcb + 96,2)
lrecl   = stg(jfcb +1Ø4,2)
ncp     = stg(jfcb +1Ø6,1)
optcd   = stg(jfcb +1Ø1,1)
recfm   = stg(jfcb +1ØØ,1)
line.a = ' '
if bfaln ¬= byteØ then do
   select
      when bitand(bfaln,'Ø2'x) = 'Ø2'x then bfa = 'D'
      when bitand(bfaln,'Ø1'x) = 'Ø1'x then bfa = 'F'
      otherwise bfa = '???'
      end
   line.a = line.a 'BFALN('bfa')'
   select
      when bitand(bfaln,'6Ø'x) = '6Ø'x then bfa = 'D'
      when bitand(bfaln,'Ø1'x) = 'Ø1'x then bfa = 'F'
      otherwise bfa = '???'
      end
   line.a = line.a 'BFALN('bfa')'
   select
      when bitand(bfaln,'6Ø'x) = '6Ø'x then bft = 'A'
      when bitand(bfaln,'4Ø'x) = '4Ø'x then bft = 'S'
      when bitand(bfaln,'2Ø'x) = '2Ø'x then bft = 'R'
      when bitand(bfaln,'1Ø'x) = '1Ø'x then bft = 'E'
```

```
      when bitand(bfaln,'Ø8'x) = 'Ø8'x then bft = 'D'
      otherwise bft = '???'
      end
   line.a = line.a 'BFTEK('bft')'
   end
if blksize ¬= byteØ||byteØ then do
   line.a = line.a 'BLKSIZE('x2d(c2x(blksize))')'
   end
if bufno ¬= byteØ then do
   line.a = line.a 'BUFNO('x2d(c2x(bufno))')'
   end
if bufl ¬= byteØ||byteØ then do
   line.a = line.a 'BUFL('x2d(c2x(bufl))')'
   end
if bufoff ¬= byteØ then do
   line.a = line.a 'BUFOFF('x2d(c2x(bufoff))')'
   end
if den ¬= byteØ then do
   select
      when bitand(den,'43'x) = '43'x then dens = '1'
      when bitand(den,'83'x) = '83'x then dens = '2'
      when bitand(den,'43'x) = 'C3'x then dens = '3'
      when bitand(den,'43'x) = 'D3'x then dens = '4'
      otherwise dens = '?'
      end
   line.a = line.a 'DEN('dens')'
   end
if flag2 ¬= byteØ then do
   if bitand(flag2,'Ø4'x) = 'Ø4'x then diagns = 'DIAGNS(TRACE)'
   else diagns = ' '
   line.a = strip(line.a||diagns)
   select
      when bitand(flag2,'8Ø'x) = '8Ø'x then inout = 'INPUT'
      when bitand(flag2,'4Ø'x) = '4Ø'x then inout = 'OUTPUT'
      otherwise inout = '???'
      end
   line.a = line.a inout
   end
if dsorg ¬= byteØ||byteØ then do
   select
      when bitand(dsorg1,'8Ø'x) = '8Ø'x then dso = 'IS'
      when bitand(dsorg1,'81'x) = '81'x then dso = 'ISU'
      when bitand(dsorg1,'4Ø'x) = '4Ø'x then dso = 'PS'
      when bitand(dsorg1,'41'x) = '41'x then dso = 'PSU'
      when bitand(dsorg1,'2Ø'x) = '2Ø'x then dso = 'DA'
      when bitand(dsorg1,'21'x) = '21'x then dso = 'DAU'
      when bitand(dsorg1,'Ø2'x) = 'Ø2'x then dso = 'PO'
      when bitand(dsorg1,'Ø3'x) = 'Ø3'x then dso = 'POU'
      otherwise dso = '??'
      end
   line.a = line.a 'DSORG('dso')'
```

```
   end
if eropt ¬= byte0 then do
   select
      when bitand(eropt,'80'x) = '80'x then ero = 'ACC'
      when bitand(eropt,'40'x) = '40'x then ero = 'SKP'
      when bitand(eropt,'20'x) = '20'x then ero = 'ABE'
      otherwise ero = '???'
      end
   line.a = line.a 'EROPT('ero')'
   end
if expdt ¬= substr(full0,1,3) then do
   yy = x2d(c2x(substr(expdt,1,1)))
   yy =  2000 + yy - 100
   ddd = x2d(c2x(substr(expdt,2,2)))
   line.a = line.a 'EXPDT('yy'.'ddd')'
   end
if keylen ¬= byte0 then do
   select
      when bitand(keylen,'23'x) = '23'x then trt = 'E'
      when bitand(keylen,'3B'x) = '3B'x then trt = 'T'
      when bitand(keylen,'13'x) = '13'x then trt = 'C'
      when bitand(keylen,'2B'x) = '2B'x then trt = 'ET'
      otherwise trt = ' '
      end
   if trt ¬= ' ' then line.a = line.a 'TRTCH('trt')'
   line.a = line.a 'KEYLEN('x2d(c2x(keylen))')'
   end
if lrecl ¬= byte0||byte0 then do
   if lrecl = '8000'x then lrec = 'X'
   else lrec = x2d(c2x(lrecl))
   line.a = line.a 'LRECL('lrec')'
   end
if limct ¬= byte0||byte0 then do
   line.a = line.a 'LIMCT('x2d(c2x(limct))')'
   end
if ncp ¬= byte0 then do
   line.a = line.a 'NCP('x2d(c2x(ncp))')'
   end
if optcd ¬= byte0 then do
   select
      when bitand(optcd,'80'x) = '80'x then opt = 'W'
      when bitand(optcd,'80'x) = '80'x then opt = 'W'
      when bitand(optcd,'40'x) = '40'x then opt = 'U'
      when bitand(optcd,'20'x) = '20'x then opt = 'C'
      when bitand(optcd,'10'x) = '10'x then opt = 'H'
      when bitand(optcd,'08'x) = '08'x then opt = 'Q'
      when bitand(optcd,'04'x) = '04'x then opt = 'Z'
      when bitand(optcd,'02'x) = '02'x then opt = 'T'
      when bitand(optcd,'01'x) = '01'x then opt = 'J'
      otherwise opt = '???'
      end
```

```
      line.a = line.a 'OPTCD('opt')'
      end
   if recfm ¬= byteØ then do
      rf2 = ''; rf3 = ''; rf4 = ''
      select
         when bitand(recfm,'CØ'x) = 'CØ'x then rf1 = 'U'
         when bitand(recfm,'8Ø'x) = '8Ø'x then rf1 = 'F'
         when bitand(recfm,'4Ø'x) = '4Ø'x then rf1 = 'V'
         when bitand(recfm,'2Ø'x) = '2Ø'x then rf1 = 'T'
         otherwise rf1 = '?'
         end
      if bitand(recfm,'1Ø'x) = '1Ø'x then rf2 = 'B'
      if bitand(recfm,'Ø8'x) = 'Ø8'x then rf3 = 'S'
      if bitand(recfm,'Ø4'x) = 'Ø4'x then rf4 = 'A'
      if bitand(recfm,'Ø2'x) = 'Ø2'x then rf4 = 'M'
      line.a = line.a 'RECFM('rf1||rf2||rf3||rf4')'
      end
   tiotptr = tiotptr + tioelngh             /* get next entry         */
   tioelngh = c2d(stg(tiotptr,1))           /* get entry length       */
end
ddname.Ø = a
do j = 1 to ddname.Ø
   say ddname.j strip(line.j)
end
if ddname.Ø = Ø then
   say 'No attribute lists were found'
exit
/*--------------------------------------------------------------------*/
ptr: return c2d(storage(d2x(arg(1)),4)) /* return a pointer           */
/*--------------------------------------------------------------------*/
stg: return storage(d2x(arg(1)),arg(2)) /* return storage             */
/*--------------------------------------------------------------------*/
swareq:
if right(c2x(arg(1)),1) ¬= 'F' then     /* swa=below ?                */
   return c2d(arg(1)) + 16              /* yes, return sva+16         */
sva = c2d(arg(1))                       /* convert to decimal         */
tcb = c2d(storage(21c,4))               /* tcb psatold                */
tcb = ptr(54Ø)                          /* tcb psatold                */
jscb = ptr(tcb + 18Ø)                   /* jscb tcbjscb               */
qmpl = ptr(jscb + 244)                  /* qmpl jscbqmpi              */
qmat = ptr(qmpl + 24)                   /* qmat qmadd                 */
do while sva > 65536
   qmat = ptr(qmat + 12)                /* next qmat qmat+12          */
   sva = sva - 65536                    /* Ø10006f -> ØØØØ6f          */
   end
return ptr(qmat + sva + 1) + 16
```

Brett Berger
Systems Programmer (USA)

High resource users – accumulated statistics suite based on SMF records: update and ISPF interface extension – part 2

This month we conclude the code for a tool/suite to help users to select batch jobs for performance monitoring. The code incorporates the generated statistics within an ISPF suite to allow user-friendly selection of job steps to monitor.

```
fcomm16 = ''
fcomm17 = ''
fcomm18 = ''
fcomm19 = ''
fcomm2Ø = ''
t1 = ''
t2 = ''
t3 = ''
t4 = ''
t5 = ''
t6 = ''
t7 = ''
t8 = ''
t9 = ''
ta = ''
tb = ''
tc = ''
td = ''
te = ''
tf = ''
tg = ''
th = ''
ti = ''
tj = ''
tk = ''
btchØ3trc = ''
btchØ3tpf = ''
'ISPEXEC DISPLAY PANEL(BTCHØ3T)'
btchØ3trc = rc
btchØ3tpf = pfkey
filter.1.1  = filterØ1
filter.1.2  = t1
filter.1.3  = fcommØ1
filter.2.1  = filterØ2
filter.2.2  = t2
filter.2.3  = fcommØ2
```

```
filter.3.1  = filterØ3
filter.3.2  = t3
filter.3.3  = fcommØ3
filter.4.1  = filterØ4
filter.4.2  = t4
filter.4.3  = fcommØ4
filter.5.1  = filterØ5
filter.5.2  = t5
filter.5.3  = fcommØ5
filter.6.1  = filterØ6
filter.6.2  = t6
filter.6.3  = fcommØ6
filter.7.1  = filterØ7
filter.7.2  = t7
filter.7.3  = fcommØ7
filter.8.1  = filterØ8
filter.8.2  = t8
filter.8.3  = fcommØ8
filter.9.1  = filterØ9
filter.9.2  = t9
filter.9.3  = fcommØ9
filter.1Ø.1 = filter1Ø
filter.1Ø.2 = ta
filter.1Ø.3 = fcomm1Ø
filter.11.1 = filter11
filter.11.2 = tb
filter.11.3 = fcomm11
filter.12.1 = filter12
filter.12.2 = tc
filter.12.3 = fcomm12
filter.13.1 = filter13
filter.13.2 = td
filter.13.3 = fcomm13
filter.14.1 = filter14
filter.14.2 = te
filter.14.3 = fcomm14
filter.15.1 = filter15
filter.15.2 = tf
filter.15.3 = fcomm15
filter.16.1 = filter16
filter.16.2 = tg
filter.16.3 = fcomm16
filter.17.1 = filter17
filter.17.2 = th
filter.17.3 = fcomm17
filter.18.1 = filter18
filter.18.2 = ti
filter.18.3 = fcomm18
filter.19.1 = filter19
filter.19.2 = tj
```

```
  filter.19.3 = fcomm19
  filter.20.1 = filter20
  filter.20.2 = tk
  filter.20.3 = fcomm20
  do fil = 1 to 20
    fcomm = left(filter.fil.3,46)
    ucomm = '(' || userid() || ')'
    fcomm = overlay(ucomm,fcomm,48,10,' ')
    select
      when filter.fil.1 = '' then
        freecnt = freecnt + 1
      when filter.fil.2 = 'P' then
        do
          fpgmname = filter.fil.1
          fjobname = ''
          "ISPEXEC TBADD filter ORDER "
        end
      otherwise
        do
          fjobname = filter.fil.1
          fpgmname = ''
          "ISPEXEC TBADD filter ORDER "
        end
    end
  end
return
/* ******************************************************************** */
/* compare selection against the Filters                                */
/* ******************************************************************** */
match: procedure
arg compvar,filter,lng,wildch,wildco
wildchar = 1
if wildch = '' then wildchar = 0
matchrc = 1
if wildco <> '' then
  do
    newlng = pos(wildco,filter,1)
    if newlng > 0 then lng = newlng - 1
  end
do fm = 1 to lng
    if wildchar then
      if substr(filter,fm,1) = wildch then
        iterate
    if substr(filter,fm,1) = substr(compvar,fm,1) then
      iterate
    do
      matchrc = 0
      leave
    end
end
```

```
matchrc = matchrc
return matchrc
/* ***************************************************************** */
/* block select of top/high number of entries                        */
/* ***************************************************************** */
high_select:
"ISPEXEC TBTOP btchst "
do high
  "ISPEXEC TBSKIP btchst NUMBER(1)"
  sb = Ø
  sbpgm = pgmname
  sbjob = jobname
  call select_block
  if sb then
    sel = '_'
  else
    sel = 'S'
  "ISPEXEC TBMOD btchst ORDER "
end
"ISPEXEC TBTOP btchst "
return
/* ***************************************************************** */
/* check whether entry is selectable                                 */
/* ***************************************************************** */
select_block:
  sb = Ø
  "ISPEXEC TBTOP filter"
  "ISPEXEC TBSKIP filter NUMBER(1)"
  tbskiprc = rc
  do while tbskiprc < 8
    if fjobname = '' then
      do
        filter = fpgmname
        compvar = pgmname
        sb = match(compvar,filter,8,wildch,wildco)
      end
    else
      do
        filter = fjobname
        compvar = jobname
        sb = match(compvar,filter,8,wildch,wildco)
      end
    if sb then leave
    "ISPEXEC TBSKIP filter NUMBER(1)"
    tbskiprc = rc
  end
return sb
/* ***************************************************************** */
/* load FILTER table                                                 */
/* ***************************************************************** */
```

```
open_filter_read:
  "ISPEXEC TBOPEN filter WRITE "
  select
    when rc = 0 then
      do
        "ISPEXEC TBTOP filter"
        do forever
          "ISPEXEC TBSKIP filter"
          if rc > 0 then leave
          "ISPEXEC TBDELETE filter"
          if rc > 0 then leave
        end
        call read_filter
      end
    when rc = 8 then
      do
        "ISPEXEC TBCREATE filter KEYS() NAMES(fjobname fpgmname fcomm)"
        if rc > 0 then
          do
            errorfunc = 'TBCREATE'
            errortab  = 'filter'
            call errormethod
          end
        else
          call read_filter
        "ISPEXEC TBSORT filter FIELDS(fjobname,C,A fpgmname,C,A)"
        if rc > 0 then
          do
            errorfunc = 'TBSORT'
            errortab  = 'filter'
            call errormethod
          end
        fjobname = ''
        fpgmname = ''
        "ISPEXEC TBADD filter ORDER"
      end
    otherwise
      do
        errorfunc = 'TBOPEN'
        errortab  = 'filter'
        call errormethod
      end
  end
return
/* ********************************************************************* */
/* read FILTER dataset                                                   */
/* ********************************************************************* */
read_filter:
"ALLOC FI("dd3") DA("ds3") SHR REUSE "
filt. = ''
```

```
"EXECIO * DISKR "dd3" (STEM filt. FINIS"
do fcnt = 1 to filt.Ø
  parse var filt.fcnt ftyp fname fcomm
  if ftyp = 'P' then
    do
      fjobname = ''
      fpgmname = fname
    end
  else
    do
      fpgmname = ''
      fjobname = fname
    end
  "ISPEXEC TBADD filter ORDER"
end
"ISPEXEC TBCLOSE filter "
"ISPEXEC TBOPEN filter NOWRITE "
"FREE DSNAME("ds3")"
return
/* ******************************************************************** */
/* write FILTER dataset                                                 */
/* ******************************************************************** */
write_filter:
filt. = ''
rc = MSG('OFF')
ADDRESS TSO
"FREE DSNAME("ds3")"
"DELETE "ds3
"ALLOC DD("dd3") NEW CATALOG REUSE ",
"DSN("ds3") ",
"LRECL(8Ø) BLKSIZE(3272Ø) RECFM(F,B) ",
"DSORG(PS) ",
"STORCLAS("stor3") MGMTCLAS("mgmt3")",
"VOLUME("vol3") UNIT("unit3")",
"SPACE("spc3") TRACKS"
rc = MSG('ON')
"ISPEXEC TBTOP filter"
fcnt = Ø
do forever
  "ISPEXEC TBSKIP filter "
  if rc = Ø then
    do
      if fjobname = '' then ftyp = 'P'
      else ftyp = 'J'
      fcnt = fcnt + 1
      ffilter = left((fjobname || fpgmname),8)
      fcomm = left(strip(fcomm),58)
      filt.fcnt = ftyp || ' ' || ffilter || ' ' || fcomm
    end
    else leave
```

```
end
"EXECIO "fcnt" DISKW "dd3" (STEM filt. FINIS"
"ISPEXEC TBEND filter "
"FREE DSNAME("ds3")"
return
/* ************************************************************** */
/* remove SELECT duplicates and display selections                */
/* ************************************************************** */
rem_dups_disp_sel:
  rc = MSG('OFF')
  "ALLOC DD("seldd") DSN("seldsn") SHR REUSE "
  "EXECIO * DISKR "seldd" (STEM selin. FINIS"
  "FREE DSNAME("seldsn")"
  "DELETE "seldsn
  "ALLOC DD("seldd") NEW CATALOG REUSE ",
  "DSN("seldsn") ",
  "LRECL(8Ø) BLKSIZE(32720) RECFM(F,B) ",
  "DSORG(PS) ",
  "STORCLAS("stor2") MGMTCLAS("mgmt2")",
  "VOLUME("vol2") UNIT("unit2")",
  "SPACE("spc2") TRACKS"
  "FREE DSNAME("seldsn")"
  rc = MSG('ON')
  "ISPEXEC TBCREATE select",
  "KEYS(jobname stepnum)",
  "NAMES(stepname pgmname gomin uid ds ts)"
  do sel = 1 to selin.Ø
    jobname  = word(selin.sel,1)
    stepnum  = word(selin.sel,2)
    stepname = word(selin.sel,3)
    pgmname  = word(selin.sel,4)
    gomin    = word(selin.sel,5)
    uid      = word(selin.sel,6)
    ds       = word(selin.sel,7)
    ts       = word(selin.sel,8)
    "ISPEXEC TBADD select"
  end
  "ISPEXEC TBSORT select FIELDS(jobname,C,A,stepnum,C,A)"
  do forever
    selectrc = ''
    selectpf = ''
    "ISPEXEC TBDISPL select PANEL(BTCHØ4T)"
    if selectrc >= 8 | zcmd = 'CANCEL' then
      do
        leave
      end
    do forever
      selectrc = rc
      selectpf = pfkey
      if fun = 'D'  | fun = 'd'   then
```

```
      "ISPEXEC TBDELETE select"
      if ZTDSELS > 1 then
        "ISPEXEC TBDISPL select "
      else leave
    end
    if selectpf = "PFØ3" then
      do
        leave
      end
  end
  "ISPEXEC TBTOP select"
  outpos = Ø
  do forever
    "ISPEXEC TBSKIP select NUMBER(1) "
    if rc = Ø then
      do
        outpos  = outpos + 1
        selout.outpos = jobname stepnum stepname pgmname gomin uid ds ts
      end
      else leave
  end
  selout.Ø = outpos
  "ISPEXEC TBEND select"
  "ALLOC DD("seldd") DSN("seldsn") SHR MOD "
  "EXECIO "outpos" DISKW "seldd" (stem selout. FINIS"
  "FREE DSNAME("seldsn")"
return
/* ***************************************************************** */
/* find string within jobname stepname pgmname totsrvyy totsrvav     */
/* and position on row with the next occurrence                      */
/* ***************************************************************** */
find_it:
   zcmd_ok = "ok"
   parse value zcmd with o1 argument
   find_argument = argument
   upper argument
   argument = strip(argument)
   zsel = null
   crp = ztdtop
   rowid = crp
   start_crp = crp
   find_loop = null
   search = null
   if substr(o1,1,2) = "RF" then do
/* if o1 = "RFIND" then do  */
      argument = last_string
      find_argument = argument
      if prev_crp <> start_crp then last_find = start_crp
      last_find = last_find + 1
      "ISPEXEC TBTOP  btchst"
```

```
      "ISPEXEC TBSKIP btchst   POSITION(ROWID) NUMBER("last_find")"
      end
      else do
           "ISPEXEC TBSKIP btchst   Position(ROWID) Number("start_crp")"
           end
   if rc = 8 then do
                 s_smsg = find_argument "Found Wrapped"
                 "ISPEXEC TBTOP  btchst"
                 "ISPEXEC TBSKIP btchst     POSITION(ROWID)"
                 end
            else s_smsg = find_argument "Found"
   /* perform search */
   last_string = argument
   do forever
      line_contents = jobname stepname pgmname totsrvyy totsrvav
      search = translate(line_contents)
      if pos(argument,search) > Ø then do
         crp = rowid + Ø
         rowcrp = crp
         last_find = crp
             smsg = s_smsg  /* "Found" */
             lmsg = find_argument "found during search in row:" crp
             call do_msg
         prev_crp = start_crp
         return
         end
      "ISPEXEC TBSKIP btchst POSITION(Rowid)"
      if rc = 8 then do
            "ISPEXEC TBTOP btchst"
             s_smsg = find_argument "Found Wrapped"
         if find_loop = "on" then do
             smsg = find_argument "Not Found"
             lmsg = find_argument "Not found during search"
             rowid = crp
             call do_msg
             prev_crp = start_crp
             return
             end
             else find_loop = "on"
         end
      zsel = null
      end
return
do_msg:
   "ISPEXEC SETMSG MSG(BTCHØØØ)"
return
```

BTCH01T

```
)ATTR
```

```
       %   TYPE(TEXT) INTENS(HIGH)
       +   TYPE(TEXT) INTENS(LOW)
       _   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
  £   TYPE(TEXT) INTENS(LOW) COLOR(GREEN)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW) JUST(LEFT)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(PINK) JUST(RIGHT)
  ]   TYPE(OUTPUT) INTENS(LOW)  COLOR(TURQ) JUST(RIGHT)
  ^   TYPE(OUTPUT) INTENS(LOW)  COLOR(RED) JUST(RIGHT)
  #   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY  EXPAND(//)
%/-/ Batch Job Statistics /-/
+Command==>_zcmd                                  / /+Scroll=>_AMT +
$zuser    +/ /+SORTS%SJ+Jobname%SY+Yearly%SA+Average / /$timestmp
+
+/ /[info                    / /+
%Go+or%PF3+To Complete Selection %PF4+CANCEL   %HIGH nnn+to autoselect
jobsteps
%FILter+ show Filters %Find+string %RF+(repeat find)    +with the
highest usage
£-------------------------------------------------------------------
|------|
%S+Select  £ |        |        |  |                        |PERCENTAGES|
|
%U+Unselect £|        |        |  |  TOTAL SERVICE UNITS   CPU|  |IO|  |
|
£?  |JOBNAME |STEPNAME|PGMNAME |NM|   YEARLY    | AVERAGE  |  SRB|  |MSO
RANK |
£-|-|--------|--------|--------|--|------------|----------|--|--|--|--|-
-----|
)MODEL CLEAR(fun) ROWS(ALL)
_z[z$jobname $stepname$pgmname $z ^totsrvyy     [totsrvav  õz [z õz [z
$row
)INIT
.HELP = BTCHØ1H
.zvars ='(fun,sel,stepnum,procpu,prosrb,proio,promso)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
&AMT = 'CSR '
)REINIT
REFRESH(*)
)PROC
&pfkey = &zpfkey
)HELP
)END
```

BTCH02T

```
)ATTR
       %   TYPE(TEXT) INTENS(HIGH)
       +   TYPE(TEXT) INTENS(LOW)
```

```
       _   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(GREEN)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  õ   TYPE(OUTPUT) INTENS(LOW)  COLOR(TURQ) JUST(LEFT)
  #   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY EXPAND(//)
+
%/-/ Batch Job Statistics FILTER List /-/
+Command==>#zcmd                                  / /+Scroll=>#AMT +
$zuser   +/ /$timestmp        +
+/ /[info                   / /+
            +command %FIlter+ (new Filters)
+
%D+(Delete)
+
+  JOBNAME  PGMNAME  COMMENTS
+
+-|--------|--------|-------------------------------------------------
------|
)MODEL CLEAR(fun) ROWS(ALL)
£z$fjobname$fpgmnameõfcomm
)INIT
.HELP = BTCHØ2H
.zvars ='(fun)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
&AMT = 'CSR'
)REINIT
REFRESH(*)
)PROC
&pfkey = &zpfkey
)HELP
)END
```

BTCH03T

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH)
  $   TYPE(OUTPUT) INTENS(HIGH) COLOR(TURQ)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
)BODY DEFAULT(%+#) EXPAND(//)
+
%/-/ Batch Job Statistics FILTER List /-/
+Command==>#zcmd                                  / /+Scroll=>#AMT +
$zuser   +/ /$timestmp        +
+/ /[info                   / /+
%P+(PGMNAME Filter)                 %_+Wildcard(single)
+
%J+(JOBNAME Filter)                 %*+Willcard(multiple)
```

```
+
+  FILTER   COMMENTS
+
+-|--------|-----------------------------------------------|
£z#filter01#fcomm01                                       +            +
£z#filter02#fcomm02                                       +            +
£z#filter03#fcomm03                                       +            +
£z#filter04#fcomm04                                       +            +
£z#filter05#fcomm05                                       +            +
£z#filter06#fcomm06                                       +            +
£z#filter07#fcomm07                                       +            +
£z#filter08#fcomm08                                       +            +
£z#filter09#fcomm09                                       +            +
£z#filter10#fcomm10                                       +            +
£z#filter11#fcomm11                                       +            +
£z#filter12#fcomm12                                       +            +
£z#filter13#fcomm13                                       +            +
£z#filter14#fcomm14                                       +            +
£z#filter15#fcomm15                                       +            +
£z#filter16#fcomm16                                       +            +
£z#filter17#fcomm17                                       +            +
£z#filter18#fcomm18                                       +            +
£z#filter19#fcomm19                                       +            +
£z#filter20#fcomm20                                       +            +
+-|--------|-----------------------------------------------|
)INIT
.HELP = BTCH03H
.zvars  ='(t1,t2,t3,t4,t5,t6,t7,t8,t9,ta,tb,tc,td,te,tf,tg,th,ti,tj,tk)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
&AMT = 'CSR'
)REINIT
REFRESH(*)
)PROC
&pfkey = &zpfkey
)HELP
)END
```

BTCH04T

```
)ATTR
  £   TYPE(INPUT) CAPS(ON) INTENS(HIGH)
  $   TYPE(OUTPUT) INTENS(LOW) COLOR(YELLOW)
  [   TYPE(OUTPUT) INTENS(HIGH) COLOR(RED)
  }   TYPE(OUTPUT) INTENS(LOW) COLOR(WHITE)
)BODY DEFAULT(%+#) EXPAND(//)
+
%/-/ SELECTions to be processed List /-/
+Command==>#zcmd                                  / /+Scroll=>#AMT +
$zuser   +/ /$timestmp        +
```

```
+/ /[info                    / /+
%D+(Delete)+/ / DSN = }dsnname                                          +
+-|--------|------|--------|--------|----|--------------------|
+?|JOBNAME |STEPNUM|STEPNAME|PGMNAME |GOMIN|REQUESTOR & TIME    |
+-|--------|------|--------|--------|----|--------------------|
)MODEL CLEAR(fun) ROWS(ALL)
£z$jobname $stepnum$stepname$pgmname $gomin$uid     $ds    $ts  +
)INIT
.HELP = BTCHØ4H
.zvars ='(fun)'
&timestmp = '&zday..&zmonth..&zstdyear. &ztime'
&AMT = 'CSR'
)REINIT
REFRESH(*)
)PROC
&pfkey = &zpfkey
)HELP
)END
```

BTCH00

```
BTCHØØØ  '&SMSG' .WINDOW=NORESP .TYPE=NOTIFY
'&LMSG'
```

BTCH01H

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
%/-/ BTCHSDSP Overview /-/
+
%JOBNAME STEPNAME PGMNAME RANK+- self-explanatory
%NM+- the Jobstep number - physical position of Step within Job
%PERCENTAGES+- Per Type of Service Unit - the Percentage (rounded)
      represented within the Total Service Units
     %CPU+ Central Processing Unit %SRB+ Service Request Block
     %IO+ Input Output %MSO+ Main Storage Occupancy
%Total Service Units+ displayed as average and yearly
+The average is simply calculated as the total usage divided by the
+occurrencies of the jobstep.
+The Yearly estimated usage is calculated as follows:
+The total usage divided by the number of days in the sample period
+multiplyed by the days in a year.
+If only one occurrence of a jobstep is present then it is assumed that
+the jobstep runs once a year (not once a day) and the yearly estimated
+usage will be the same as the average usage
+
+Select for command information  ===>_zcmd    +
%1+SORTS+
```

```
%2+Go
%3+HIgh nnn
%4+FILter
%5+FIND(RF)
%6+Line Command S(U)
)INIT
&ZCONT=BTCHØ1H1
)PROC
&ZIND=YES
&ZSEL=TRANS(&ZCMD
           1,*BTCHØ1H1
           2,*BTCHØ1H2
           3,*BTCHØ1H3
           4,*BTCHØ1H4
           5,*BTCHØ1H5
           6,*BTCHØ1H6
           *,'?')
)END
```

BTCH01H1

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
%/-/ SORTS /-/
+
+ Three possible sorts of the data are available:
+%SJ+- Sort on Jobname
+%SA+- Sort on average Total Service Unit usage
+%SY+- Sort on yearly estimated Total Service Unit usage
+
+ When the%HIgh nnn+command is used the physically displayed%nnn+
+ records will be selected. (Excluding those that have been filtered
+ out).
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H
&ZCONT=BTCHØ1H2
)END
```

BTCH01H2

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
%/-/ Go /-/
+
```

```
+ This command is used when all selections have been made and checked.
+ The command caused the routine to be closed and writes the selected
+ records to a dataset.
+ Automatically the routine goes into a new PANEL, which shows the
+ name of the dataset used and displays the entries that are stored
+ there. The dataset contents are an accumulation of selections since
+ the dataset was last processed and emptied, not just the entries from
+ this run of the routine.
+ This dataset is a list of records to be used to prepare Performance
+ Measurement requests in a Performance Measurement tool eg STROBE
+ from Compuware
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H1
&ZCONT=BTCHØ1H3
)END
```

BTCH01H3

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
%/-/ HIgh nnn /-/
+
+ This command enables automatic selection of bulk requests the first
+%nnn+ records at the physical top of the displayed data.
+ When no sorts have been used, this will be the default and will be
+ the %nnn+records with the highest estimated yearly Total Service Unit
+ usage.
+ This will alter when a SORT has been performed to the top%nnn+ records
+ of average TSU usage, Jobname or again yearly TSU usage respectively.
+ Selected records will then be denoted with a%S+ in front. Records that
+ have been 'filtered out' are denoted with a %-+.
+ Once selected, the records may be unselected by using the line command
+%U+
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H2
&ZCONT=BTCHØ1H4
)END
```

BTCH01H4

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
```

```
%/-/ FILter /-/
+
+ This command calls a panel showing the current 'filters' in place.
+ The filters may be deleted in this new panel or by using the command
+%FILter a second time will lead to yet another panel where filters may
+ be added.
+ Filters are compared either with the Jobname or the Program name to
+ restrict the selection of unwanted selections. eg Standard Programs
+ or non-Production one-off jobs etc.
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H3
&ZCONT=BTCHØ1H5
)END
```

BTCH01H5

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
%/-/ FIND string (RF) /-/
+
+ This is a normal Find routine that will scan the data from the
+ current position forwards until the string is found or the end of the
+ data is reached.
+ The normal RFIND command may be used as repeat find, to find the next
+ occurrence of the previously-entered string. The abbreviation RF
+ functions also as 'repeat find'.
+
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H4
&ZCONT=BTCHØ1H6
)END
```

BTCH01H6

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
+
%/-/ Lins commands S(U) /-/
+
+ These line commands may be used to singularly selected or unselect a
+ record. When%S has been used and the record is selected, it will be
+ prefixed in the display with an%S+.
+ The selection%S+ may be turned off (deselected) by using%U+.
```

```
+
)INIT
)PROC
&ZHTOP=BTCHØ1H
&ZUP=BTCHØ1H5
&ZCONT=BTCHØ1H
)END
```

BTCH02H

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
%/-/ Batch Job Statistics FILTER List /-/
+                                                                              +
+
+ This Panel list the filters in place. They are sorted alphabetically
+ per program name, filter and job name filter.
+ At this point unwanted filters may be deleted from the list by typing
+ a %D+ before the entry.
+ To create additional filters enter %FILTER+ to go to the next screen
+ where they can be entered.
+ Filters in effect are used to block selections. The selections that
+ match filters will be discarded and will be shown with a %-+ on the
+ selection panel.
)INIT
)PROC
&ZHTOP=BTCHØ2H
&ZUP=BTCHØ2H
&ZCONT=BTCHØ2H
)END
```

BTCH03H

```
)BODY DEFAULT(%+#) EXPAND(//)
+/-/ BTCHSDSP Help /-/
%/-/ BTCHØ3H /-/
+
+
+
+
%BTCHØ3T
%---------------------- Batch Job Statistics FILTER List ---------------
+Command==>
Scroll=
+AL13745
14.Ø7.2ØØ4
+
%P+(PGMNAME Filter)                 %_+Wildcard(single)
```

```
%J+(JOBNAME Filter)                 %*+Willcard(multiple)
+  FILTER   COMMENTS
+-|--------|-------------------------------------------|
+
+
+
+ This screen is for the input of new filters.
+ A single filter works on either Programname or Jobname selection,
+ but both can be combined by using both types to exclude various
+ combinations.
+
+ Examples with description  (enter %1+or page on)
+
%Select  ===>#zcmd                                                  +
%1+ ===>  Examples with description
)INIT
)PROC
&ZIND=YES
&ZSEL=TRANS(&ZCMD
           1,*BTCHØ3H1
           *,'?')
&ZHTOP=BTCHØ3H
&ZUP=BTCHØ3H
&ZCONT=BTCHØ3H1
)END
```

BTCH03H1

```
)ATTR
  $   TYPE(TEXT) INTENS(HIGH) COLOR(TURQ)
  [   TYPE(TEXT) INTENS(HIGH) COLOR(RED)
  £   TYPE(TEXT) INTENS(HIGH) COLOR(GREEN)
)BODY DEFAULT(%+#) EXPAND(//)
+/-/ BTCHSDSP Help /-/
 BTCHØ3T
 ---------------------- Batch Job Statistics FILTER List ---------------
 Command==>[FI+
Scroll=[CSR
$AL13745
$14.Ø7.2ØØ4

%P+(PGMNAME Filter)                 %_+Wildcard(single)
%J+(JOBNAME Filter)                 %*+Willcard(multiple)
   FILTER   COMMENTS
  -|--------|-------------------------------------------|
[p pg_na*   Program name filter P1
[j job*     Job name filter J1
[j j_b*     Job name filter J2
[p p___ram  Program name filter P2
```

```
[p program* Program filter P3
+
$JOBNAME %PROGNAME[Filtered out by:
£--------|--------|------------------
£USRJOBØ1 PROGNAM1 none
£JOBUSRØ1 PROGNAM1[J1 J2
£JOCUSRØ1 PROGRAM1[P3
£JAB1     PROG1   [J2
£PRDRUNØ1 COBOLØØ1 none
£PG1NAØØ2 COBOLØØ2 none
£COBOLØØ2 PG1NAØØ2[P1
£PRDRUNØ2 PROGRAM [P3 P2
£
+
+
%Recommended Filters:
+-|--------|-----------------------------------------------|
[p DFH*     STANDARD PROGRAM
[p DSNUTILB STANDARD PROGRAM
[p ICEGENER STANDARD PROGRAM
[p IDCAMS   STANDARD PROGRAM
[p IEB*     STANDARD PROGRAM
[p IEFBR14  STANDARD PROGRAM
[p IEWL     STANDARD PROGRAM
[p SORT     STANDARD PROGRAM
[p SYSSORT  STANDARD PROGRAM
)INIT
)PROC
&ZHTOP=BTCHØ3H
&ZUP=BTCHØ3H
&ZCONT=BTCHØ3H
)END
```

BTCH04H

```
)BODY DEFAULT(%+_) EXPAND(//)
+/-/ BTCHSDSP Help /-/
%/-/ BTCHØ4H /-/
%/-/ SELECTions to be processed List /-/
+                                                                              +
+
+ This list is a summary of the job steps to be forwarded for
+ performance monitoring.
+ The list will be stored in the dataset named after DSN =.
+ The content of this dataset is an accumulation of selections from
+ BTCHSDSP since the last time the dataset was processed and the content
+ discarded.
+ At this point unwanted selections may be deleted from the list by
+ typing a %D+ before the entry.
```

```
+ Duplicate selections have been automatically removed.
)INIT
)PROC
&ZHTOP=BTCHØ4H
&ZUP=BTCHØ4H
&ZCONT=BTCHØ4H
)END
```

Rolf Parker
Systems Programmer (Germany)

A peek at WLM's decision making

I once read the following 'expert' advice: 'There is no reason to write SMF type 99 records unless IBM asks you to in order to debug a problem, ie WLM not doing what it "should" be doing. They are used by WLM development and service to understand what happened when a problem with WLM's decision-making process is alleged/suspected/discovered.'

With all due respect to the expert's advice, the chances that one will recognize a problem in time to preserve 10 minutes of data are, in my experience, very small – which is why I have the type 99 turned on. The problem with not having the records turned on is that, if there ever is a problem that IBM needs to look at, the SMF99 records are about the only thing they have to work with, and the problem cannot always be recreated. Therefore, I keep it for a couple of days and then send it to data archive. But, what are SMF type 99 records anyway? First of all, this record type is written by the SRM component when running in goal mode. The records contain:

- Performance data for each service class period.
- Trace codes representing the SRM actions.
- The data that SRM used to decide which actions to take.
- The controls SRM is using to manage work.

The mapping macro, IRASMF99, for this record is supplied in SYS1.AMODGEN.

The type 99 data is excellent for understanding detailed aspects of WLM decision logic not covered fully in other documentation. As is commonly known, Workload Manager periodically assesses the performance of each service class period by comparing the performance achieved by it against the performance goals defined for it. WLM does this by sampling the state of the service class four times per second. This assessment is done at each goal importance level. In this way, WLM can determine whether the service class is using resources or being delayed in a manner that may be adjustable. These sorts of delays over which WLM can exert no control are discarded in this assessment and do no contribute directly to WLM's decision making.

Briefly stated, the policy adjustment function is in charge of meeting the transaction's service class goals and resource group service rate requirements as stated in the service policy. This is achieved by distributing resources among the service classes. Throughout this process some trade-offs need to be made and in general, the 'golden rule' is to equalize PIs by importance within resource group constraints. The policy adjustment function is invoked every 10 seconds. The heart of this function is the policy adjustment loop. While the policy adjustment loop is a complex mechanism, we provide a simple description here and explain only its major steps and functions. The loop can be seen as doing the following:

- Analysing the system.
- Recognizing the service class periods that need help.
- Choosing one (and only one) service class period to be helped.
- Finding out what resources that service class period needs most.
- Looking for a service class period that can give up this resource.

- Adjusting the appropriate priority or target to reflect that new resource distribution.

What really happens, according to a WLM development team e-mail, is this:

1 At every policy adjustment cycle WLM examines the PI of all service classes.

2 If the PI is greater than 1, the service class is eligible for help.

3 WLM ranks service classes, showing which should be helped first, second, etc. Ranking depends on the importance and how far the PI is above 1.

4 Now WLM picks the highest ranked candidate and looks for its biggest problem. This is done by examining the delays of the service class. The highest delay is considered first.

5 An assessment starts to find one or multiple other service classes using the same resource and which could contribute to the service class's needing help. If the assessment comes to the conclusion that help is possible, the changes will be done and the algorithm ENDS at this point.

6 If it is not possible to help the service class for this delay (explanation follows), the second highest delay is examined and the algorithm goes to Step 5 again until either it can help the service class or it comes to the conclusion that the service class cannot be helped.

7 In cases where the service class cannot be helped, the service class period with the next highest rank is examined to find out whether it can be helped, and so on.

What does it mean, 'it cannot be helped'? It means that the anticipated changes will not achieve sufficient gain for the performance index of the service class needing help, or will not show sufficient gain for its service consumption, or the

situation will become worse for the service class(es) that are donating their resources.

What does this mean for a high PI? It just means that the likelihood of getting help might become lower, but it doesn't mean that WLM does not try to help it at all. As a matter of fact, WLM will always assess whether it can help this service class, potentially at every policy adjustment cycle (every 10 seconds). Potentially the examination depends on the rank of this service class, which could change every 10 seconds depending on whether more important work exists that needs help.

In order to document its decisions, WLM creates several SMF records (type 99) for each policy interval, or approximately once every 10 seconds. They can be useful in analysing and understanding the performance characteristics of a site's workload. The records contain performance data for each service class period, a trace of SRM actions, the data SRM used to decide which actions to take, and the internal controls SRM uses to manage work. This can help the performance analyst to determine in detail what SRM is doing to meet workload goals defined with respect to other work, and the types of delay the work is experiencing. By stepping through a sequence of type 99 records, you can actually watch as WLM recognizes a failing goal and proceeds to adjust resources to help that service class, and, more importantly, see the impact of weaning resources away from the 'donor' service class. You begin to understand why your WLM definitions may not be doing what you think they should be doing! In this respect I have found that the following subtypes of record 99 are particularly useful for understanding the decision-making process of WLM:

- Subtype 1 contains system-level data, the trace of SRM actions, and data about resource groups. The SRM actions are recorded in trace codes. All trace codes are described in *Appendix A* of the *z/OS MVS Programming: Workload Management Services* (SA22-7619) manual. A subtype 1 record is written at every policy interval.

- Subtype 2 contains data for service classes. A subtype 2 record is written every policy interval for each service class if any period in the service class had recent activity.

A detailed description of the layout of an SMF type 99 record can be obtained from the *MVS System Management Facilities* (SA22-7630) manual.

Please note that because SMF type 99 records are written approximately every 10 seconds, massive volumes of data are created, so you should write them only for certain time periods. Therefore, it is a good idea to have an SMFPRM*xx* parmlib member for general audit information that does not specify type 99, and another for detailed audit information that does specify type 99. This way, you can set the proper SMFPRM*xx* member and write SMF type 99 records only when you need them.

CODE

The code is a four-part stream. The first part (DEL99) is a clean-up step, which deletes the files to be used in later steps. In the second step (EXT99) SMF records 99 are extracted from SMF dataset to a file, which can be used as a base of archived records. In the third part (COPY99), previously extracted records (selection being defined by INCLUDE's condition) are sorted and copied to a file, which is the input to WLM9912 EXEC (WLMREXX step), and several reports are produced. These reports were tailored after the template reports one can find described in great detail in Chapter 10 (*Using SMF Record Type 99*) of the *z/OS MVS Programming: Workload Management Services* (SA22-7619) manual. I consider this chapter to be the most important factor for any useful and meaningful discussion of the decision-making process of WLM.

```
//DEL99     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE hlq.R99.DATA
    DELETE hlq.SMFCOPY.OUT
```

```
    SET MAXCC=Ø
/*
//EXT99    EXEC PGM=IFASMFDP,REGION=5M
//INDA1 DD DSN=your.smf.dataset,DISP=SHR
//OUTDA DD DSN=hlq.SMFCOPY.OUT,DISP=(NEW,PASS),
//         UNIT=SYSDA,SPACE=(CYL,(5Ø,2Ø),RLSE),
//         DCB=(your.smf.dataset)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
       DATE(yyyyddd,yyyyddd)
       START(Ø9ØØØ)
       END(17ØØØ)
       INDD(INDA1,OPTIONS(DUMP))
       OUTDD(OUTDA,TYPE(99))
/*
//SORT99   EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hlq.SMFCOPY.OUT,DISP=SHR
//SMF9912  DD DSN=hlq.R99.DATA,
//         SPACE=(CYL,(15)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø)
//TOOLIN   DD *
  SORT FROM(RAWSMF) TO(SMF9912) USING(SMF9)
//SMF9CNTL DD *
* Eliminate Header and Trailer records
* Get SMF 99.1 and 2 with valid WLM data.
* Sort by date and time
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,99,AND,(23,2,BI,EQ,1,
               OR,23,2,BI,EQ,2))
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//WLMREXX  EXEC PGM=IKJEFTØ1,REGION=ØM
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.lib
//SMF746   DD DISP=SHR,DSN=hlq.U746.DATA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%WLM9912
/*
```

WLM9912 EXEC

```
/* REXX EXEC to read and format SMF records
   pertaining to WLM: Type 99 subtypes 1 & 2 */
signal ON ERROR
ADDRESS TSO
  userid=SYSVAR(SYSUID)
```

```
  trace  =userid||'.wlm9.trace.rep'     /* Trace report DSN         */
  serv   =userid||'.wlm9.serv.rep'      /* Server report DSN        */
  qserv  =userid||'.wlm9.qserv.rep'     /* Queue server report DSN  */
  asesp  =userid||'.wlm9.asesp.rep'     /* ASID storage report DSN  */
x = MSG('ON')
IF SYSDSN(trace) = 'OK'
THEN "DELETE "trace" PURGE"
IF SYSDSN(serv) = 'OK'
THEN "DELETE "serv" PURGE"
IF SYSDSN(qserv) = 'OK'
THEN "DELETE "qserv" PURGE"
IF SYSDSN(asesp) = 'OK'
THEN "DELETE "asesp" PURGE"
"ALLOC FILE(S991) DA("trace")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(30,15) CATALOG",
   " REUSE RELEASE LRECL(160) RECFM(F B)"
"ALLOC FILE(S991A) DA("serv")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(30,15) CATALOG",
   " REUSE RELEASE LRECL(100) RECFM(F B)"
"ALLOC FILE(S992A) DA("qserv")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(30,15) CATALOG",
   " REUSE RELEASE LRECL(180) RECFM(F B)"
"ALLOC FILE(S993A) DA("asesp")",
   " UNIT(SYSALLDA) NEW TRACKS SPACE(30,15) CATALOG",
   " REUSE RELEASE LRECL(70) RECFM(F B)"
  y = 1
   /*-------------------------------------------------------------*/
  /* Header for Server data entry                                 */
   /*-------------------------------------------------------------*/
  servt.1= left('Server Sample Data:',20)
  servt.2= left(' ',61,' ') left('Delays',7),
           left('Aux paging delay samples',25)
  servt.3= left('Date',11) left('Time',12) left('PA int.id',10),
           left('Class name',11)        left('Server type',14),
           left('WLM',4) left('priv',5) left('VIO',5),
           left('HSP',5) left('swap',5) left('MPL',5)
  servt.4= left('-',96,'-')
 a=1
 b=1
   /*-------------------------------------------------------------*/
  /* Header for Queue Server data entry                           */
   /*-------------------------------------------------------------*/
  qervt.1= left('Queue Server Sample Data:',30)
  qervt.2= left(' ',70,' ') left('Queued requests:',17),
           left('Instances:',29) left('QMPL:',23) left('Spaces:',26),
           left('Appl.env.',9)
  qervt.3= left('Date',11) left('Time',12) left('PA int.id',10),
           left('Class',8) left('Env.name',10) left('Work Queue',14),
           left('Avg.',4)  left('Long',4),
           left('Inelig.',7)     left('needed',7)  left('bound',5),
```

```
            left('busy',4)        left('idle',4)    left('/serv',5),
            left('reach',6)       left('busy',4)    left('in-target',11),
            left('capacity',9)    left('removed',7) left('RGNWORK',8),
            left('Max',3)         left('Min',3)
  qervt.4= left('-',18Ø,'-')
   /*-------------------------------------------------------------*/
  /* Header for Address Space Expanded Storage                    */
   /*-------------------------------------------------------------*/
  expvt.1= left('Address Space Expanded Storage:',52),
           left('Frames:',17)
  expvt.2= left('Date',11)   left('Time',12) left('Class',9),
           left('AS name',9) left('ASId',7)  left('CS   ES  PPS',14)
  expvt.3= left('-',66,'-')
   /*-------------------------------------------------------------*/
  /* Header for Resource Group report                             */
   /*-------------------------------------------------------------*/
  rges.1= left(' ',1)
  rges.2= left('Resource Group information',4Ø)
  rges.3= left(' ',1)
  rges.4= left('Interval',24) left('RG name',8),
          left('Min.sr',6)    left('Max.sr',6) left('Act.sr',6),
          left('Spac',4)      left('Slices',6)
  rges.5= left('-',65,'-')
mq = 1
   /*-------------------------------------------------------------*/
  /*  Generic Resource Routing header                             */
   /*-------------------------------------------------------------*/
  gres.1= left(' ',1)
  gres.2= left('Generic Resource',4Ø)
  gres.3= left(' ',35,' ')  left('Sessions:',12)left('Cost',4)
  gres.4= left('Interval',22)  left('Sys name',12),
          right('Tso',3)       right('Other',6),
          right('Tso',4)       right('Avg.pi',6),
          right('Shortage',8)
  gres.5= left('-',8Ø,'-')
  mh = 1
numeric digits 2Ø
 'EXECIO * DISKR SMF ( STEM x. FINIS'
     do i = 1 to x.Ø
  SMF99RTY = c2d(SUBSTR(x.i,2,1))              /* SMF record type */
  SMF99TID = c2d(SUBSTR(x.i,19,2))             /* Record subtype  */
   /*-------------------------------------------------------------*/
  /* Unpack SMF date & decode SMF time                            */
   /*-------------------------------------------------------------*/
    SMF99DTE = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5) /* unpack SMF date */
    SMF99TME= smf(c2d(substr(x.i,3,4)))
    SMF99SID = substr(x.i,11,4)                /* System id.      */
    SMF99SSID= substr(x.i,15,4)                /* Sub System Id.  */
   /*-------------------------------------------------------------*/
  /*          Self-Defining section                               */
```

```
 /*-------------------------------------------------------------*/
POF  = c2d(SUBSTR(x.i,25,4))     /* Offset to the PRODUCT section */
PLN  = c2d(SUBSTR(x.i,29,2))     /* Length of the product section */
PON  = c2d(SUBSTR(x.i,31,2))     /* Number of product sections    */
DOF  = c2d(SUBSTR(x.i,33,4))     /* Offset to the DATA section    */
DLN  = c2d(SUBSTR(x.i,37,2))     /* Length of the data section    */
DON  = c2d(SUBSTR(x.i,39,2))     /* Number of data sections       */
 /*-------------------------------------------------------------*/
/*          PRODUCT information                                  */
 /*-------------------------------------------------------------*/
  POF = POF - 3
SMF99VN2   = c2d(SUBSTR(x.i,pof,2))       /* Record sub version   */
SMF99RVN   = c2d(SUBSTR(x.i,pof+2,2))     /* Record Version Number*/
SMF99PNM   =     SUBSTR(x.i,pof+4,8)      /* Product Name - SRM   */
SMF99SLV   =     SUBSTR(x.i,pof+12,8)     /* System level         */
SMF99SNM   =     SUBSTR(x.i,pof+2Ø,8)     /* System name          */
 Select
   when SMF99TID =  2  Then call subt2 dof
   otherwise do
 /*-------------------------------------------------------------*/
/*     Subtype 1: Self-defining section                        */
 /*-------------------------------------------------------------*/
DOF  = DOF - 3
TOF  = c2d(SUBSTR(x.i,dof,4))     /* Offset to TRACE sec.         */
TLN  = c2d(SUBSTR(x.i,dof+4,2))   /* Length of Trace sec.         */
TON  = c2d(SUBSTR(x.i,dof+6,2))   /* Number of Trace sec.         */
SSOF = c2d(SUBSTR(x.i,dof+8,4))   /* Offset to SYSTAM STATE sec.  */
SSLN = c2d(SUBSTR(x.i,dof+12,2))  /* Length of System state sec.  */
SSON = c2d(SUBSTR(x.i,dof+14,2))  /* Number of system state sec.  */
PPOF = c2d(SUBSTR(x.i,dof+16,4))  /* Offset to PAGING plot sec.   */
PPLN = c2d(SUBSTR(x.i,dof+2Ø,2))  /* Length of Paging plot sec.   */
PPON = c2d(SUBSTR(x.i,dof+22,2))  /* Number of paging plot sec.   */
PTOF = c2d(SUBSTR(x.i,dof+24,4))  /* Offset to PRIORITY table sec.*/
PTLN = c2d(SUBSTR(x.i,dof+28,2))  /* Length of Priority table sec.*/
PTON = c2d(SUBSTR(x.i,dof+3Ø,2))  /* Number of Priority table sec.*/
RGOF = c2d(SUBSTR(x.i,dof+32,4))  /* Offset to RESOURCE Group sec.*/
RGLN = c2d(SUBSTR(x.i,dof+36,2))  /* Length of resource group sec.*/
RGON = c2d(SUBSTR(x.i,dof+38,2))  /* Number of resource group sec.*/
GROF = c2d(SUBSTR(x.i,dof+4Ø,4))  /* Offset to generic resouce se.*/
GRLN = c2d(SUBSTR(x.i,dof+44,2))  /* Length of generic resource   */
GRON = c2d(SUBSTR(x.i,dof+46,2))  /* Number of generic resource se*/
SLOF = c2d(SUBSTR(x.i,dof+48,4))  /* Offset to SW licensing sec.  */
SLLN = c2d(SUBSTR(x.i,dof+52,2))  /* Length of SW licensing sec.  */
SLON = c2d(SUBSTR(x.i,dof+54,2))  /* Number of SW licensing sec.  */
SLTOF= c2d(SUBSTR(x.i,dof+56,4))  /* Offset to SW licensing       */
                                  /* service table sec.           */
SLTLN= c2d(SUBSTR(x.i,dof+6Ø,2))  /* Length of SW licensing       */
                                  /* service table sec.           */
SLTON= c2d(SUBSTR(x.i,dof+62,2))  /* Number of SW licensing       */
                                  /* service table sec.           */
```

```
    /*-------------------------------------------------------------*/
    /*    Software Licensing Information -                         */
    /*    Contains information about License Manager.              */
    /*-------------------------------------------------------------*/
if  (SLOF <> Ø) & (SLLN <> Ø) Then
DO
slof = slof - 3
    /*-------------------------------------------------------------*/
    /*           Flags                                             */
    /*-------------------------------------------------------------*/
SLCONFFLG=x2b(c2x(SUBSTR(x.i,slof,1)))  /* Configuration flags       */
SLSTSI = substr(SLCONFFLG,1,1)           /* Indicates that the machine*/
                                         /* supports the store system */
Select                                   /* information instruction.  */
  when SLSTSI = 1 then op1="Store Sys.Info suported/";
  otherwise         op1= "Store Sys.Info not supported ";
End
SLLPAR = substr(SLCONFFLG,2,1)         /*Indicates that MVS is running*/
                                         /* in a logical partition    */
Select
  when SLLPAR = 1 then op2="Running in LPAR/";
  otherwise         op2= " ";
End
SLVM   = substr(SLCONFFLG,3,1)         /*Indicates that MVS is running*/
                                         /* in a virtual machine      */
Select
  when SLVM   = 1 then op3="Running in VM";
  otherwise         op3= "";
End
SLSHLOG = substr(SLCONFFLG,4,1)          /* Indicates that the        */
               /* logical CPUs are shared  with other partitions     */
Select
  when SLSHLOG= 1 then op4="CPU shared/";
  otherwise         op4= "CPU not shared";
End
SLCAPCUST = substr(SLCONFFLG,5,1)        /* Indicates that the logical*/
          /* partition is configured to be capped (as opposed to     */
                                         /* being capped by WLM)      */
Select
  when SLCAPCUST = 1 then op5="LPAR to be capped";
  otherwise         op5= "WLM capped";
End
Config =op1||op2||op3||op4||op5
SLSTATFLGS=x2b(c2d(SUBSTR(x.i,slof+1,1))) /* State flags.            */
SLCAPPYWLM=substr(SLSTATFLGS,1,1)         /*Indicates that the logical*/
                                          /*partition is capped by WLM*/
Select
  when SLCAPPYWLM = 1 then State= "LPAR is actually WLM capped";
  otherwise                State= "LPAR is no actually WLM capped";
End
```

```
   /*-----------------------------------------------------------------*/
   /*    The following fields describe the/MVS image                  */
   /*    and CEC capacity.                                            */
   /*-----------------------------------------------------------------*/
SLIMGCAPACITY=c2d(SUBSTR(x.i,slof+4,4))   /*Capacity available to MVS*/
    /* image in millions of service units per hour, when not running*/
   /* as VM guest. If running as VM guest, capacity available to VM.*/
SLCECCAPACITY=c2d(SUBSTR(x.i,slof+8,4)) /*Capacity of CEC in millions*/
                                        /* of service units per hour.*/
   /*-----------------------------------------------------------------*/
   /*    The following fields describe the/CEC and MVS                */
   /*    image configuration. They are used to calculate the CPU      */
   /*    capacity of the CEC and MVS image                            */
   /*-----------------------------------------------------------------*/
SLCECCPUNUM  =c2d(SUBSTR(x.i,slof+12,2)) /*Number of available CPUs  */
   /* in the CEC. This includes online and offline CPUs. It does not*/
   /* include reserved CPUs (CPUs that can be added via             */
                                  /* Capacity Upgrade on Demand).    */
SLLOGICALCPUNUM=c2d(SUBSTR(x.i,slof+14,2))
    /* Number of available CPUs in the logical partition. This      */
    /* includes online and offline CPUs. It does not include        */
    /* reserved CPUs (CPUs that can be added via Cap.Upgrade on     */
                                   /* Demand).                      */
SLCECSUSECSHARE=c2d(SUBSTR(x.i,slof+16,4)) /*The CEC capacity in    */
    /* basic-mode service units per second that is available for    */
    /* sharing among partitions using shared logical processors.    */
SLIMGMSUCURRWEIGHT = ,             /* MVS image capacity in millions  */
            c2d(SUBSTR(x.i,slof+2Ø,4))  /* of service units per hour */
    /* that is represented by the partition's current weight.       */
   /*-----------------------------------------------------------------*/
   /*   The following fields describe the logical                     */
   /*   partition's actual CPU usage.                                 */
   /*-----------------------------------------------------------------*/
SLAVGMSU    =c2d(SUBSTR(x.i,slof+28,4))  /* Average service rate in  */
   /* millions of service units per hour. This is a long-term average*/
SLAVGMSUCAP =c2d(SUBSTR(x.i,slof+32,4))   /* Average service rate in */
       /* millions of service units per hour while the partition was */
                                /* capped. This is a short-term average.*/
SLAVGMSUUNCAP=c2d(SUBSTR(x.i,slof+36,4))  /*Average service rate in  */
          /* millions of service units  per hour while the partition */
          /* was uncapped. This is a short-term average              */
SLINTERVALSER =c2d(SUBSTR(x.i,slof+4Ø,4)) /* Service units over last */
       /* policy adjustment interval. NOTE: The service units are    */
       /* calculated using the MP factor for the number of           */
       /* physical CPUs, not the number of logical CPUs. This is     */
       /* consistent with how capacity is measured for software      */
       /* licensing. These service units cannot be directly          */
       /* compared to other service units calculated by SRM.         */
SLINTERVALTIME=c2d(SUBSTR(x.i,slof+44,4)) /* Elapsed time over last  */
       /* policy adjustment interval in  1.Ø24 milliseconds          */
```

```
SLROLLINTERVAL=c2d(SUBSTR(x.i,slof+52,2))        /* Number of policy */
                        /* adjustment intervals between computation   */
                                    /* of average service rate.        */
SLSERVTABINT  =c2d(SUBSTR(x.i,slof+54,2)) /* Number of consecutive   */
    /* policy adjustment intervals that have passed since the last   */
    /* time that the service table was updated.                      */
    /*---------------------------------------------------------------*/
   /*   The following fields describe the cap pattern               */
    /*---------------------------------------------------------------*/
SLINTRCA3=c2d(SUBSTR(x.i,slof+56,2))      /* Number of consecutive   */
                /* policy adjustment intervals to cap the partition. */
SLINTRUNCAP=c2d(SUBSTR(x.i,slof+58,2))    /* Number of consecutive   */
            /* policy adjustment intervals   to uncap the partition. */
SLPATINTRVNUM=c2d(SUBSTR(x.i,slof+60,2))  /* Number of consecutive   */
      /* policy adjustment intervals that have passed in the current */
               /* cap/uncap state indicated  by SLCap- pedByWlm.     */
    /*---------------------------------------------------------------*/
   /*           The following fields are response codes             */
    /*---------------------------------------------------------------*/
SLQUERYRC =c2d(SUBSTR(x.i,slof+64,4))  /*Response code from the last */
                                      /*'query' for lpar information */
SLSETCAPRC =c2d(SUBSTR(x.i,slof+68,4))  /*Response code from the last*/
 Select
  when  i = 1 then do
  binfo.1= left('System id   :',13) left(SMF99SID,5)
  binfo.2= left('System level:',13) left(SMF99SLV,8)
  binfo.3= left('System name :',13) left(SMF99SNM,8)
  binfo.4= left('Image  MSU  :',13) left(SLIMGCAPACITY,4),
           left(' CPU num.:',10)      left(SLCECCPUNUM,4)
  binfo.5= left('System configuration FLAGS:',30)
  binfo.6= left(config,90)
  binfo.7= left('System configuration STATE:',30)
  binfo.8= left(state,45)
  binfo.9= left('SU/sec LPAR shared:',20) left(SLCECSUSECSHARE,6)
  binfo.10=left('Policy adj.computational intervals:',36),
           left(SLROLLINTERVAL,4)
   "EXECIO * DISKW s991 (STEM binfo.)"
  End
  OTHERWISE    nop
End
END                             /* of Software Licensing Information*/
    /*---------------------------------------------------------------*/
   /*  Software Licensing table                                     */
   /*  Information - Contains information about License Manager     */
    /*---------------------------------------------------------------*/
 if  (SLTOF <> 0) & (SLTLN <> 0) Then
 DO
 sltof = sltof - 3
    /*---------------------------------------------------------------*/
   /* The following fields describe the logical partition's actual   */
```

```
   /* CPU usage. The data will be filled in from oldest to newest    */
   /* table entries.                                                 */
    /*---------------------------------------------------------------*/
SLTSERVICEUNCAPPED = ,              /* Basic-mode service units      */
     c2d(SUBSTR(x.i,sltof,4))       /*accumulated while the partition*/
       /* was uncapped. NOTE: The service units are calculated using */
       /* the MP factor for the number of physical CPUs, not the     */
      /* number of logical CPUs. This is consistent with how capacity*/
      /* is measured for software licensing.  These service units    */
      /* cannot be directly compared to other service units          */
                           /* calculated in SRM.                     */
SLTSERVICECAPPED  = ,               /* Basic-mode service units      */
     c2d(SUBSTR(x.i,sltof+4,4))     /*accumulated while the partition*/
    /* was capped. NOTE: The service units are calculated using the  */
    /* MP factor for the number of physical CPUs, not the number of  */
    /* logical CPUs. This is consistent with how capacity is measured*/
    /* for software licensing. These service units cannot be directly*/
    /* compared to other service units calculated in SRM.      */
SLTSERVICEUNCAPPEDCOUNT  = ,        /* Number of seconds that the    */
      c2d(SUBSTR(x.i,sltof+8,2))    /* partition was uncapped.       */
SLTSERVICECAPPEDCOUNT     = ,       /* Number of seconds that the    */
     c2d(SUBSTR(x.i,sltof+1Ø,2))    /* partition was capped.         */
SLTSERVICELASTUPDATEINTERVAL  = ,   /* Policy adjustment interval id */
     c2d(SUBSTR(x.i,sltof+12,2)) /* when this entry was last updated.*/
     /* byte, it will wrap multiple times over the course of the     */
     /* table. (That is, the time span of the table is greater than  */
     /* 255 intervals so the interval ids will wrap around.)         */
END                          /* of Software Licensing table         */
   /*---------------------------------------------------------------*/
  /* System State information -                                     */
  /* Contains information about the general state of the system     */
   /*---------------------------------------------------------------*/
if  (SSOF <> Ø) & (SSLN <> Ø) Then
DO
ssof = ssof - 3
CPUA=c2d(SUBSTR(x.i,ssof,2))  /*Avg.Processor Utilization scaled by 16*/
UMP =c2d(SUBSTR(x.i,ssof+2,2))        /* Recent unmanaged paging and */
                                                  /* swap cost %  */
UIC1=c2d(SUBSTR(x.i,ssof+4,4))    /* Frames in UIC bucket 1 - The    */
      /* central and expanded UIC buckets contain a count of frames  */
      /* that have not been referenced for specified periods of      */
      /* time.  So, UIC bucket 1 is a count of the frames that have  */
      /* most recently been referenced, where bucket 4 contains a    */
      /* count of the frames that have not been referenced in a long */
      /* time. The UIC delimiter values specify the cutoff points for*/
     /* each bucket. For example, EUIC3 will contain the count of all*/
      /* the expanded storage frames whose time since they were last*/
      /* referenced is between ESTB2 and ESTB3.                      */
UIC2=c2d(SUBSTR(x.i,ssof+8,4))    /* Frames in UIC bucket 2          */
UIC3=c2d(SUBSTR(x.i,ssof+12,4))   /* Frames in UIC bucket 3          */
```

```
UIC4=c2d(SUBSTR(x.i,ssof+16,4))   /* Frames in UIC bucket 4          */
EUIC1 =c2d(SUBSTR(x.i,ssof+20,4)) /* Frames in Expanded UIC bucket 1 */
EUIC2 =c2d(SUBSTR(x.i,ssof+24,4)) /* Frames in Expanded UIC bucket 2 */
EUIC3 =c2d(SUBSTR(x.i,ssof+28,4)) /* Frames in Expanded UIC bucket 3 */
EUIC4 =c2d(SUBSTR(x.i,ssof+32,4)) /* Frames in Expanded UIC bucket 4 */
FRV1  =c2d(SUBSTR(x.i,ssof+36,2)) /* UIC delimiter value 1           */
FRV2  =c2d(SUBSTR(x.i,ssof+38,2)) /* UIC delimiter value 2           */
FRV3  =c2d(SUBSTR(x.i,ssof+40,2)) /* UIC delimiter value 3           */
ESTB1 =c2d(SUBSTR(x.i,ssof+42,2)) /*Expanded storage UIC del. value 1*/
ESTB2 =c2d(SUBSTR(x.i,ssof+44,2)) /*Expanded storage UIC del. value 2*/
ESTB3 =c2d(SUBSTR(x.i,ssof+46,2)) /*Expanded storage UIC del. value 3*/
W2MIG =c2d(SUBSTR(x.i,ssof+48,4)) /* Write to migrate percentage     */
PTAVAIL=smf(c2d(SUBSTR(x.i,ssof+52,4))) /* Total CPU time available  */
                            /* incl. captured time plus wait time   */
SHFLAGS = x2b(c2x(SUBSTR(x.i,ssof+56,1)))
    CSS=  substr(SHFLAGS,1,1)
    AUF=  substr(SHFLAGS,2,1)
    AUC=  substr(SHFLAGS,3,1)
    SQF=  substr(SHFLAGS,4,1)
    SQC=  substr(SHFLAGS,5,1)
 if CSS = 1      then shortage='Real storage shortage'
 else if AUF = 1 then shortage='First level Aux shortage'
 else if AUC = 1 then shortage='Critical Aux shortage'
 else if SQF = 1 then shortage='First level SQA shortage'
 else if SQC=  1 then shortage='Critical SQA shortage'
 else                 shortage='No Storage shortage'
STFLAGS = x2b(c2x(SUBSTR(x.i,ssof+57,1)))
   DCMON   = substr(STFLAGS,1,1)
   DCMGOAL = substr(STFLAGS,2,1)
   COMPAT  = substr(STFLAGS,3,1)
 if DCMON=1        then status='Dynamic chpid active'
 else if DCMGOAL=1 then status='Dynamic chpid goal is active'
 else if COMPAT=1  then status='in COMPAT mode'
 else                   status='GOAL Mode, no Dynamic chpid'
TOTPAGCOST = c2d(SUBSTR(x.i,ssof+58,2))  /* Recent total paging and  */
                                        /* swap cost percentage * 10 */
Select
  when TOTPAGCOST >'0' then TOTPAGCOST=format(TOTPAGCOST/10,3,2)
  otherwise nop
End
CPPS       = c2d(SUBSTR(x.i,ssof+60,4)) /*Common protective processor*/
                                         /* storage target in frames */
ILSUARRAY  = c2x(SUBSTR(x.i,ssof+64,32))
    k = 1                              /* Array of Importance Level SU */
    j = 0
 do while j < 8
   imp.j  = c2d(substr(ILSUARRAY,k,8))  /*A single entry in the array*/
/*of  say  imp.j   */              /* Importance Level Service Units */
   j = j + 1                            /* consumed by work at this  */
   k = k + 8                       /* this Importance Level over the */
```

```
 end                                     /* last ten seconds          */
STWSS    = c2d(SUBSTR(x.i,ssof+96,4))  /*Protective processor storage*/
                                  /* target for shared area, measured */
                                        /* in frame counts.          */
NUMEXTSC = c2d(SUBSTR(x.i,ssof+100,4)) /*No. of external serv. classes*/
DEFIOVEL = c2d(SUBSTR(x.i,ssof+104,4))  /* Default I/O velocity.     */
     /* Calculated by IOS at the beginning of each measurement       */
     /* interval during data gathering.                              */
SUIFACTOR= c2d(SUBSTR(x.i,ssof+108,4))
                                      /*Service unit inflation factor.*/
 acpu = format(CPUA/16,3,2)
  sysst.1= left(' ',1)
  sysst.2= left(date('n',SMF99DTE,'j'),11)
  sysst.3= left('Part 1: System State information at interval:',50),
           left(SMF99TME,12) left(' ',28,' ') left('SU',7),
           left('Def.',4)
  sysst.4= left('  CPU%',6) left('CPU available',13),
           left('uic1',5) left('uic2',5) left('uic3',5) left('uic4',4),
           left('Storage shortage',21)   left('UMPS%',5),
           left('Paging%',7) left('CPPS',4) left('STWSS',5),
           left('inflat.',7) left('IO Vel.',7)
  sysst.5= left('-',105,'-')
  sysst.6= left(acpu,6)  left(PTAVAIL,12),
           right(uic1,5)     right(uic2,5)  right(uic3,5) right(uic4,5),
           left(shortage,20) right(UMP,6)   right(TOTPAGCOST,6),
           right(cpps,4)     right(STWSS,4) right(SUIFACTOR,5),
           right(DEFIOVEL,6)
  sysst.7= left('-',105,'-')
    "EXECIO * DISKW s991 (STEM sysst.)"
END
   /*-----------------------------------------------------------------*/
  /*  Paging plot information - Contains the information plotted   */
  /*  on the system paging responsivness plot.                     */
   /*-----------------------------------------------------------------*/
 if  (PPOF <> 0) & (PPLN <> 0) Then
 DO
 ppof = ppof - 3
 PAGP_BW        = c2d(SUBSTR(x.i,ppof,4))        /*  Bucket width    */
 PAGP_LSTX      = c2d(SUBSTR(x.i,ppof+4,4))   /*Last plotted x bucket*/
 PAGP_POINTS_OF = c2d(SUBSTR(x.i,ppof+12,4)) /*Offset of point entries*/
 PAGP_POINTS_ON = c2d(SUBSTR(x.i,ppof+16,4)) /*Number of point entries*/
 PAGP_POINTS_LN = c2d(SUBSTR(x.i,ppof+18,4)) /*Length of a point entry*/
 END
  /*-----------------------------------------------------------------*/
  /*  Resource Group entry - Contains                              */
  /*  information about resource groups                            */
   /*-----------------------------------------------------------------*/
 if  (RGOF <> 0) & (RGLN <> 0) Then
 DO
 rgof = rgof - 3
```

```
RGNAME=     SUBSTR(x.i,rgof,8)     /* Resource group name           */
MINSR = c2d(SUBSTR(x.i,rgof+8,4)) /*Minimum service rate for the group*/
MAXSR = c2d(SUBSTR(x.i,rgof+12,4))   /*Maximum service rate for the  */
                    /*group (FFFFFFFF if not specified in the policy)*/
ACTSR = c2d(SUBSTR(x.i,rgof+16,4)) /* Service rate received in last  */
                                   /* interval on the local system   */
SPAS  = c2d(SUBSTR(x.i,rgof+20,4)) /* Service per awake slice        */
SLICES= c2d(SUBSTR(x.i,rgof+24,2)) /* Number of time slices that     */
                                    /* work in this group was capped */
RHELP0= c2d(SUBSTR(x.i,rgof+26,2)) /* A count of the remote systems  */
                             /* that can help work at importance 0    */
RHELP1= c2d(SUBSTR(x.i,rgof+28,2)) /* A count of the remote systems  */
                              /* that can help work at importance 1  */
RHELP2= c2d(SUBSTR(x.i,rgof+30,2)) /* A count of the remote systems  */
                              /* that can help work at importance 2  */
RHELP3= c2d(SUBSTR(x.i,rgof+32,2)) /* A count of the remote systems  */
                              /* that can help work at importance 3  */
RHELP4= c2d(SUBSTR(x.i,rgof+34,2)) /* A count of the remote systems  */
                              /* that can help work at importance 4  */
RHELP5= c2d(SUBSTR(x.i,rgof+36,2)) /* A count of the remote systems  */
                              /* that can help work at importance 5  */
RHELP6= c2d(SUBSTR(x.i,rgof+38,2)) /* A count of the remote systems  */
                              /* that can help work at importance 6  */
LHELPF= x2b(c2d(SUBSTR(x.i,rgof+40,1))) /*Local system can help work */
              /* at each importance. 1 - can help  0 - cannot help   */
LHELP6 =  substr(LHELPF,2,1) /*Local system help work at importance 0*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP5 =  substr(LHELPF,3,1) /*Local system help work at importance 1*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP4 =  substr(LHELPF,4,1) /*Local system help work at importance 2*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP3 =  substr(LHELPF,5,1) /*Local system help work at importance 3*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP2 =  substr(LHELPF,6,1) /*Local system help work at importance 4*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP1 =  substr(LHELPF,7,1) /*Local system help work at importance 5*/
                                   /* 1 - can help, 0 - cannot help  */
LHELP0 =  substr(LHELPF,8,1) /*Local system help work at importance 6*/
                                   /* 1 - can help, 0 - cannot help  */
RGFLAGS=x2b(c2d(SUBSTR(x.i,rgof+41,1))) /* Resource group flags      */
RGDYNAMIC = substr(RGFLAGS,1,1)  /*Indicates that the resource group */
                                   /* is a dynamic                   */
Select
  when RGDYNAMIC = '1' then rgdyn="Res.Group is dynamic"
  otherwise                 rgdyn="Res.Group is not dynamic"
End
  rgeg.mq= left(date('n',SMF99DTE,'j'),11),
           left(SMF99TME,12),
           left(rgname,8),             /* Resource group name    */
          right(minsr,5),              /* Group min.service rate */
```

```
          right(maxsr,5),              /* Group max.service rate */
          right(actsr,5),              /* Service rate received  */
          right(spas,5),               /* Service per awake slice*/
          right(slices,5)              /* No. of time slices     */
 mq = mq + 1
END                                   /* of Resource Group entry */
/*-----------------------------------------------------------------*/
/*     Generic Resource entry -                                    */
/*     Contains information about Generic Resource Routing         */
/*-----------------------------------------------------------------*/
if  (GROF <> Ø) & (GRLN <> Ø) Then
DO
grof = grof - 3
GRSYSNAME =    (SUBSTR(x.i,grof,8))    /*Name of sys. sessions were  */
GRTSO     = c2d(SUBSTR(x.i,grof+8,4))  /* routed. Number of TSO      */
                                /* sessions routed in the last 1Ø sec. */
GRNONTSO  = c2d(SUBSTR(x.i,grof+12,4)) /* Number of non-TSO sessions */
                                       /* routed in last 1Ø sec.     */
GRTSOAVG  = c2d(SUBSTR(x.i,grof+16,4)) /* Average cost of TSO session*/
                                 /* in raw CPU service units on system */
GRTSOPI   = c2d(SUBSTR(x.i,grof+2Ø,4)) /* Weighted average PI of     */
                /* service class periods running TSO work on system */
GRFLAGS   = x2b(c2x(SUBSTR(x.i,grof+24,4))) /*Flags                 */
GRSHORTAGE= c2d(SUBSTR(GRFLAGS,1,1))   /* GRSYSNAME had a shortage   */
       /* that may have caused sessions not to be routed to it       */
GRSERVIMP = c2x(SUBSTR(x.i,grof+28,32))  /* A single entry in the    */
          /* array of Importance Level Service Units, containing the */
          /* number of Service Units consumed by work at this        */
          /* Importance Level (or unused) over the last ten seconds. */
          /* The entries are indexed with an origin of zero so that  */
          /* the index matches the Importance Level to which the     */
          /* entry pertains. Index of zero corresponds to system work*/
          /* and index of 7 to unused capacity                       */
    z = 1
    w = Ø
 do while w < 8
  simp.w  = c2d(substr(GRSERVIMP,z,8))
   w = w + 1
   z = z + 8
 end
  gges.mh= left(date('n',SMF99DTE,'j'),11),
           left(SMF99TME,12),
          right(GRSYSNAME,8),
          right(grtso,5),                /* No.of TSO sess. routed */
          right(grnontso,5),             /* No.of non-TSO sess.    */
          right(grtsoavg,5),             /* Avg. cost of TSO sess. */
          right(grtsopi,5),              /* Weighted average PI    */
          right(GRSHORTAGE,5)
    mh = mh + 1
END                                     /* of Generic Resource entry*/
```

```
   /*-------------------------------------------------------------*/
   /*  Priority table entry - Contains                            */
   /*  information pertaining to the demand being put on the      */
   /*  processor by work at different dispatch priorities.        */
   /*-------------------------------------------------------------*/
  ptt.1  = left(' ',1)
  ptt.2  = left('Part 2: PRIORITY TABLE at interval',5Ø)
  ptt.3  = left(' ',9,' ') left('% CPU demand',16) left('CPU used',12),
           left('Cum. max demand:',16) left('MTTW avg',12),
           left('Cpu samp.',11) left('Sample based',12)
  ptt.4  = left(' DP',5)  left('DPC',3)    left('init',4)
left('proj.',5),
           left('W2UR',5) left('actual',6) left('proj.',5),
           left('ach.',4) left('init',5)   left('proj.',5),
           left('init',5) left('proj.',6)  left('use',5)  left('dly',5),
           left('use',5)  left('dly',4)
  ptt.5  = left('-',92,'-')
if  (PTOF <> Ø) & (PTLN <> Ø) Then
    "EXECIO * DISKW s991 (STEM ptt. )"
DO
  pty.ext = left(' ',1)
     do r = Ø to PTON -1
       ptff = (PTOF + (r*PTLN))- 3
ext      = r+1
PTPRTY=c2d(SUBSTR(x.i,ptff,2))    /* Dispatch priority               */
PTNP  =c2d(SUBSTR(x.i,ptff+2,2))  /* New dispatch priority (Ø = not  */
PTIMDP=c2d(SUBSTR(x.i,ptff+4,4))  /* changed). Initial max. % of     */
            /* processor demanded at priority, initial value before */
                              /* any priority moves or slice changes */
PTPMDP=c2d(SUBSTR(x.i,ptff+8,4))  /* Projected max. % of processor   */
                                  /* demanded at priority            */
PTCPUU=c2d(SUBSTR(x.i,ptff+12,4)) /* CPU using samples at priority   */
PTCPUD=c2d(SUBSTR(x.i,ptff+16,4)) /* CPU delay samples at priority   */
PTW2UR=c2d(SUBSTR(x.i,ptff+2Ø,4)) /* Wait-to-using ratio             */
W2UR  = format(PTW2UR/16,3,2)                     /* at priority *16 */
PTAPU =c2d(SUBSTR(x.i,ptff+24,4))
                                 /*Actual measured proc.used at priority*/
PTPPU =c2d(SUBSTR(x.i,ptff+28,4))
                           /* Projected proc.time to be used at prty  */
PTACMD=c2d(SUBSTR(x.i,ptff+32,4)) /* Achievable cumulative max demand*/
                             /* for priorities affected by a move */
PTACMD= format(PTACMD/1Ø,3,1)
PTIMAXD=c2d(SUBSTR(x.i,ptff+4Ø,4))  /* Initial cumulative max demand */
PTIMAXD=format(PTIMAXD/1Ø,3,1)
PTWMAXD=c2d(SUBSTR(x.i,ptff+44,4))  /*Projected cumulative max demand*/
PTWMAXD=format(PTWMAXD/1Ø,3,1)
PTIAMTW=c2d(SUBSTR(x.i,ptff+48,4))  /* Initial avg. mean time to wait*/
Select
 when PTIAMTW >'Ø' then  P1IAMTW=format(PTIAMTW/1ØØØ,5,2)
 otherwise               P1IAMTW = PTIAMTW
```

```
End
PTWAMTW=c2d(SUBSTR(x.i,ptff+52,4)) /*Projected avg. mean time to wait*/
Select
 when PTWAMTW >'Ø' then  P1WAMTW=format(PTWAMTW/1ØØØ,5,2)
 otherwise               P1WAMTW = PTWAMTW
End
PTSCPUU=c2d(SUBSTR(x.i,ptff+56,4))  /* Sample based CPU using samples*/
                                    /* at priority                  */
PTSCPUD=c2d(SUBSTR(x.i,ptff+6Ø,4))  /* Sample based CPU delay samples*/
                                    /* at priority                  */
                                    /* Printed variables:           */
  pty.ext = right(PTPRTY,4),        /* Dispatch priority            */
            right(PTNP,4),          /* New dispatch priority        */
            right(PTIMDP,4),        /* % of CPU demanded - init     */
            right(PTPMDP,4),        /*                   - projected */
            right(W2UR,5),          /* Wait-to-using ratio          */
            right(PTAPU,5),         /* CPU used - actual measured   */
            right(PTPPU,5),         /*          - projected         */
            right(PTACMD,5),        /* Cum. max demand - achievable  */
            right(PTIMAXD,5),       /*                 - initial    */
            right(PTWMAXD,5),       /*                 - projected  */
            right(P1IAMTW,5),       /* MTTW - initial avg           */
            right(P1WAMTW,5),       /*      - projected avg.        */
            right(PTCPUU,5),        /* Cpu samples - using          */
            right(PTCPUD,5),        /*             - delay          */
            right(PTSCPUU,5),     /* Sample based CPU samples - using */
            right(PTSCPUD,5)        /*                          - delay */
end                                 /* of extent                    */
    "EXECIO * DISKW s991(STEM pty. )"
END                                 /*       of Priority table     */
   /*-----------------------------------------------------------------*/
  /*  TRACE table entry - The trace                                  */
  /*  table contains information about the policy and resource       */
  /*  adjustment decisions made during the last policy interval*/
   /*-----------------------------------------------------------------*/
if  (TOF <> Ø) & (TLN <> Ø) Then
 do
     trcc.1 = left(' ',1)
     trcc.2 = left('Part 3: TRACE entry at interval for period > Ø',6Ø)
     trcc.3 = left('Class',6) left('Period',7)           left('LPI',4),
              left('SPI',5)   left('WLM action code',26)
left('PAIid',6),
              left('RAIid',5)
     trcc.4 = left('-',65,'-')
               "EXECIO * DISKW s991 (STEM trcc. )"
   do j = Ø to TON -1
     tff  = (TOF + (j*TLN))- 3
EXTnum   = j+1
TPID =c2d(SUBSTR(x.i,tff,1))      /* Policy adjustment interval id.  */
TRID =c2d(SUBSTR(x.i,tff+1,1))    /* Resource adjustment interval id.*/
```

```
TCOD =c2d(SUBSTR(x.i,tff+2,2))    /* Policy adjustment action code   */
TJOB =    SUBSTR(x.i,tff+4,8)     /* Job name of address space       */
                                  /* affected by action being traced */
                                  /* (blank if not an address space  */
                                  /* level action)                   */
TLPI =c2d(SUBSTR(x.i,tff+12,4))       /* Projected local performance */
                                  /* index to be achieved as a result*/
                                  /* of the traced action            */
TSPI =c2d(SUBSTR(x.i,tff+16,4))    /* Projected sysplex performance  */
                                  /* index to be achieved as a result*/
                                  /* of the traced action            */
TGSR =c2d(SUBSTR(x.i,tff+20,4))   /* Projected resource group        */
                                  /* service rate be achieved as a   */
                                  /* result of the traced action     */
TDT1 =c2d(SUBSTR(x.i,tff+24,4))       /* System use                  */
TDT2 =c2d(SUBSTR(x.i,tff+28,4))       /* System use                  */
TDT3 =c2d(SUBSTR(x.i,tff+32,4))       /* System use                  */
TRGN =    SUBSTR(x.i,tff+36,8)        /* Resource group Name         */
                                     /* (blank if no resource group) */
TCNM =    SUBSTR(x.i,tff+44,8)        /* Service Class Name          */
TPER =c2d(SUBSTR(x.i,tff+52,2))       /* Service period number       */
TASID=c2d(SUBSTR(x.i,tff+54,2))       /* Address Space ID number     */
LPI  = TLPI/100
SPI  = TSPI/100
c =1
Select
  when  TPER > '0' then do            /*  Printed variables:         */
     trac.c =  left(TCNM,8),          /*  Service Class Name         */
               right(tper,2),         /*  Service period number      */
               right(lpi,5),          /*  Projected Local PI         */
               right(spi,5),          /*  Projected Sysplex PI       */
               right(tcod,5),         /*  Policy adj.action code     */
               left(Tctext(TCOD),22),
               right(tpid,4),         /*  Policy adj. interval id    */
               right(trid,5)          /*  Resource adj.interval id   */
               "EXECIO * DISKW s991 (STEM trac. )"
                c =c +1
  end
 otherwise nop
  End
END /* select */
  s2nfo.1= left(' ',1)
  s2nfo.2= left('Part 4: CLASS / PERIOD data at interval',50)
  s2nfo.3= left('Class',6)     left('Help',5),
           left('Period',6)    left('Imp',3),
           left('Goal ',11)    left('SPI ',4) left('LPI ',5) ,
           left('DP ', 4)      left('I/O DP',6),
           left('Serv',5)      left('MDP',4),
           left('MPLI ',5)     left('MPLO ', 5) ,
           left('Swap PT ',7)  left('PSI',8) ,
```

```
           left('EXP Policies',14) ,
           left('Delays ', 9)
  s2nfo.4= left('-',12Ø,'-')
   "EXECIO * DISKW s991 (STEM s2nfo.)"
end
end
end
END  /* of main loop */
   "EXECIO * DISKW s991a (STEM servt.)"
   "EXECIO * DISKW s991a (STEM sdrec.)"
   "EXECIO * DISKW s992a (STEM qervt.)"
   "EXECIO * DISKW s992a (STEM sqrec.)"
   "EXECIO * DISKW s993a (STEM expvt.)"
   "EXECIO * DISKW s993a (STEM exrec.)"
   "EXECIO * DISKW s991 (STEM rges.)"
   "EXECIO * DISKW s991 (STEM rgeg.)"
   "EXECIO * DISKW s991 (STEM gres.)"
   "EXECIO * DISKW s991 (STEM gges.)"
/* Close & free all allocated files  */
 "EXECIO Ø DISKW S991(FINIS "
 "EXECIO Ø DISKW S991A(FINIS "
 "EXECIO Ø DISKW S992A(FINIS "
 "EXECIO Ø DISKW S993A(FINIS "
 say
 say 'WLM trace report dsn ..............:'trace
 say 'WLM Server report dsn .............:'serv
 say 'WLM Queue server report dsn .......:'qserv
 say 'WLM Address Space Expanded Storage :'asesp
 say
 "free FILE(S991 S991A S992A S993A)"
exit
/*-------------------------------------------------------------*/
/* Error exit routine                                          */
/*-------------------------------------------------------------*/
ERROR: say 'The following command produced non-zero RC =' RC
       say SOURCELINE(SIGL)
       exit
SUBT2:
parse arg dof
 /* Subtype 2 main loop - start                                */
  /*-----------------------------------------------------------*/
 /*    Subtype 2 self defining section                         */
  /*-----------------------------------------------------------*/
DOF  = dof -3
COF    = c2d(SUBSTR(x.i,dof,4))        /*Offset to class information */
CLN    = c2d(SUBSTR(x.i,dof+4,2))      /*Length of class information */
CON    = c2d(SUBSTR(x.i,dof+6,2))      /*Number of class information */
CPOF   = c2d(SUBSTR(x.i,dof+8,4))    /*Offset to class period section*/
CPLN   = c2d(SUBSTR(x.i,dof+12,2))   /*Length of class period section*/
CPON   = c2d(SUBSTR(x.i,dof+14,2))   /*Number of class period section*/
```

```
 /*-------------------------------------------------------------*/
 /*  SMF99 subtype 2 Class data section - Contains              */
 /*  information identifying the Service Class described in this*/
 /*  record.                                                    */
 /*-------------------------------------------------------------*/
if  (COF <> Ø) & (CLN <> Ø) Then
DO
 cof = cof - 3
S2CNAM =   SUBSTR(x.i,cof,8)              /*      Service class name */
S2CGRN =   SUBSTR(x.i,cof+8,8)       /* Group name or blank if class */
                                        /* doesn't belong to a group */
S2CNUMP=c2d(SUBSTR(x.i,cof+1Ø,2))   /* Number of periods in the class */
 /*-------------------------------------------------------------*/
 /* SMF99 subtype 2 Period data section - Contains             */
 /* information about the Service Periods that are part of the  */
 /* Service Class described in this record.                    */
 /*-------------------------------------------------------------*/
if  (CPOF <> Ø) & (CPLN <> Ø) Then
DO
   do zz = Ø to CPON -1
    cfof  = (CPOF + (zz*CPLN))- 3
S2PCNM  =    SUBSTR(x.i,cfof,8)        /*Internal service class name */
S2PNUM  =c2d(SUBSTR(x.i,cfof+8,2))     /* Period number             */
S2PGOALTYP = ,                            /*    Goal type          */
         c2d(SUBSTR(x.i,cfof+1Ø,1))
Select
   when S2PGOALTYP = 1 then Goal ="Short RT"
                                      /* 1 - short response time     */
   when S2PGOALTYP = 2 then Goal ="Long RT " /*2 - long response time*/
   when S2PGOALTYP = 3 then Goal ="Velocity"  /* 3 - velocity       */
   when S2PGOALTYP = 4 then Goal ="Discr"     /* 4 - discretionary  */
   otherwise                Goal ="Sys.Asid  SYSSTC or Server"
End
S2PGOALVAL = ,                      /* Goal value: response time goal */
         c2d(SUBSTR(x.i,cfof+12,4))       /* goal in milliseconds,  */
                                          /* velocity - velocity,   */
                                          /* discretionary - zero   */
S2PIMPOR = c2d(SUBSTR(x.i,cfof+16,2))     /* Importance             */
S2PBDP   = c2d(SUBSTR(x.i,cfof+18,1))    /*Base dispatching priority */
S2PMPLI  = c2d(SUBSTR(x.i,cfof+2Ø,2))     /* MPL in-target          */
S2PMPLO  = c2d(SUBSTR(x.i,cfof+22,2))     /* MPL out-target         */
S2PAMTA  = c2d(SUBSTR(x.i,cfof+24,4))   /*Average Maximum MPL Target */
                                          /* achieved               */
S2PRUA   = c2d(SUBSTR(x.i,cfof+28,4))     /* Ready user average     */
S2PLRUA  = c2d(SUBSTR(x.i,cfof+32,4))  /*Long term Ready User Average*/
                                          /* times 16               */
S2PPSPT  = c2d(SUBSTR(x.i,cfof+36,4))  /*Length of time swapped ASID */
                                     /* protected in processor storage */
S2PPSITAR= c2d(SUBSTR(x.i,cfof+4Ø,4))  /*Storage isolation target for*/
      /* each ASID in period, valid only for work with short response*/
```

```
Select                              /* time goals, zero otherwise     */
   when S2PGOALTYP = 1 then PSI = S2PPSITAR
   otherwise                PSI = "n/a"
End
S2PESPOL = c2d(SUBSTR(x.i,cfof+44,1))   /*Exp. storage access policy */
      /* for demand pages - valid for periods with short resp        */
                                    /* time goals only               */
Select
  when S2PESPOL = 1 then ExpSt = "P"       /*  Protected             */
  when S2PESPOL = 2 then ExpSt = "L"       /*  LRU                   */
  when S2PESPOL = 3 then ExpSt = "S"       /*  Space Available       */
  otherwise              ExpSt = " "
End
S2PESVIO = c2d(SUBSTR(x.i,cfof+45,1))  /* Exp.storage access policy  */
             /* for VIO pages - valid for periods with short resp    */
Select
  when S2PESVIO = 1 then ExpV ="P"         /*  Protected             */
  when S2PESVIO = 2 then ExpV ="L"         /*  LRU                   */
  when S2PESVIO = 3 then ExpV ="S"         /*  Space Available       */
  otherwise              ExpV =" "
End
S2PESHSP = c2d(SUBSTR(x.i,cfof+46,1))  /*Exp. storage access policy  */
      /* for hiperspace pages - valid for periods with short         */
                                    /* resp time goals only          */
Select
  when S2PESHSP = 1 then ExpH = "P"        /*  Protected             */
  when S2PESHSP = 2 then ExpH = "L"        /*  LRU                   */
  when S2PESHSP = 3 then ExpH = "S"        /*  Space Available       */
  otherwise              ExpH = " "
End
blk =left('  ',3)
S2PESSWAP= c2d(SUBSTR(x.i,cfof+47,1))   /*Exp. storage access policy */
        /* for swap pages - valid for periods with short resp        */
Select
  when S2PESSWAP= 1 then ExpS = "P"        /*  Protected             */
  when S2PESSWAP= 2 then ExpS = "L"        /*  LRU                   */
  when S2PESSWAP= 3 then ExpS = "S"        /*  Space Available       */
  otherwise              ExpS = " "
End
expp = ExpS||blk||ExpSt||blk||ExpV||blk||ExpH
S2PPROT  = c2d(SUBSTR(x.i,cfof+48,2))      /* Num. of ASID with      */
            /* demand pages protected in  processor storage - NA for */
                   /* periods with short resp time goals             */
S2PLRU   = c2d(SUBSTR(x.i,cfof+5Ø,2))      /* Num. of ASIDs with     */
       /* demand pages subject to lru expanded storage policy -      */
              /* NA for periods with short resp time goals           */
S2PSPAV  = c2d(SUBSTR(x.i,cfof+52,2))      /* Num. of ASIDs with     */
       /* demand pages subject to space available expanded storage   */
       /*  policy - NA for periods with short resp time goals        */
S2PVIOL  = c2d(SUBSTR(x.i,cfof+54,2))      /* Num. of ASIDs with     */
```

```
          /* VIO pages subject to lru expanded storage policy -      */
         /* NA for periods with short resp time goals                */
S2PVIOS  = c2d(SUBSTR(x.i,cfof+56,2))      /* Num. of ASIDs with     */
           /* VIO pages subject to space available expanded storage  */
              /* policy - NA for periods with short resp time goals  */
S2PHSPL  = c2d(SUBSTR(x.i,cfof+58,2))      /* Num. of ASIDs with     */
         /* hiperspace pages subject to lru expanded storage policy  */
               /* - NA for periods with short resp time goals        */
S2PHSPS  = c2d(SUBSTR(x.i,cfof+60,2))      /* Num. of ASIDs with     */
      /* hiperspace pages subject to space available expanded storage*/
               /* policy - NA for periods with short resp time goals */
S2PESCS  = c2d(SUBSTR(x.i,cfof+62,2))   /* Num. of explicit storage  */
                                     /* critical classified ASIDs     */
S2PLPI   = c2d(SUBSTR(x.i,cfof+64,4))    /* Local performance index  */
S2PSPI   = c2d(SUBSTR(x.i,cfof+68,4))   /* Sysplex performance index */
S2PSERV  = c2d(SUBSTR(x.i,cfof+72,4))   /* Service accumulated during*/
                                        /* interval                  */
S2PMDP   = c2d(SUBSTR(x.i,cfof+76,4))      /* Maximum % of processor */
                                        /* time demanded             */
S2PLCPUU = c2d(SUBSTR(x.i,cfof+80,4))     /*CPU using samples during */
                                        /* last interval             */
S2PLCPUD = c2d(SUBSTR(x.i,cfof+84,4))   /* CPU delay samples during  */
                                        /* last interval             */
S2PMTTWA = c2d(SUBSTR(x.i,cfof+88,4))  /* Mean time to wait adjusted */
S2PADP   = c2d(SUBSTR(x.i,cfof+92,4))      /* Working variable for   */
                                        /* achievable demand %       */
S2PASERC = c2d(SUBSTR(x.i,cfof+96,4))   /*Avg. service consumed over */
S2PPRSER = c2d(SUBSTR(x.i,cfof+100,4))   /*window. Projected service */
S2PIDLE  = c2d(SUBSTR(x.i,cfof+104,4))     /* Idle samples           */
S2POTHR  = c2d(SUBSTR(x.i,cfof+108,4))     /* All other states       */
S2PCPUU  = c2d(SUBSTR(x.i,cfof+112,4))   /* Processor using samples */
S2PCPUD  = c2d(SUBSTR(x.i,cfof+116,4))   /* Processor delay samples */
S2PAUXP  = c2d(SUBSTR(x.i,cfof+120,4))     /* Primary private area  */
                                        /* paging delay aux      */
Select
 when S2PAUXP > 0 then d0=" Auxp/"||S2PAUXP
 otherwise             d0=""
End
S2PAUXC  = c2d(SUBSTR(x.i,cfof+124,4))   /* Common area paging delay */
                                        /* from aux              */
Select
 when S2PAUXC > 0 then d1=" Auxc/"||S2PAUXC
 otherwise             d1=""
End
S2PVIO   = c2d(SUBSTR(x.i,cfof+128,4))     /* VIO delay from aux     */
Select
 when S2PVIO  > 0 then d3=" Vio/"||S2PVIO
 otherwise             d3=""
End
S2PHSS   = c2d(SUBSTR(x.i,cfof+132,4)) /* Scrolling hyperspace delay */
```

```
Select
 when S2PHSS  > Ø then d4=" Hss/"||S2PHSS
 otherwise             d4=""
End
S2PHSC   = c2d(SUBSTR(x.i,cfof+136,4))     /* Cache hyperspace delay */
Select
 when S2PHSC  > Ø then d5=" Hsc/"||S2PHSC
 otherwise             d5=""
End
S2PASWP  = c2d(SUBSTR(x.i,cfof+14Ø,4))     /* Aux swap delay         */
Select
 when S2PASWP > Ø then d2=" Aswp/"||S2PASWP
 otherwise             d2=""
End
S2PMPLD  = c2d(SUBSTR(x.i,cfof+144,4))     /* MPL Delay              */
Select
 when S2PMPLD > Ø then d6=" Mpld/"||S2PMPLD
 otherwise             d6=""
End
S2PCAPD  = c2d(SUBSTR(x.i,cfof+148,4))     /* CPU CAP delay          */
Select
 when S2PCAPD > Ø then d7=" Capd/"||S2PCAPD
 otherwise             d7=""
End
S2PXMO   = c2d(SUBSTR(x.i,cfof+152,4))     /* Xmem other delays      */
Select
 when S2PXMO  > Ø then d8=" Xmo/"||S2PXMO
 otherwise             d8=""
End
XMEM_OF  = c2d(SUBSTR(x.i,cfof+156,4))     /* Offset to XMEM delay   */
XMEM_LN  = c2d(SUBSTR(x.i,cfof+16Ø,2))  /*Length of each XMEM delay */
XMEM_ON  = c2d(SUBSTR(x.i,cfof+162,2))  /*entry. Num. of XMEM delay */
                                                 /* samples     */
Select
  when XMEM_ON  > Ø then call XMEM  XMEM_OF XMEM_LN XMEM_ON
  otherwise nop
End
PSERV_OF = c2d(SUBSTR(x.i,cfof+164,4))  /* Offset to server section  */
PSERV_LN = c2d(SUBSTR(x.i,cfof+168,2))  /*Length of each server entry*/
PSERV_ON = c2d(SUBSTR(x.i,cfof+17Ø,2))  /* Num. of server entries    */
Select
  when PSERV_ON > Ø then call SERV  PSERV_OF PSERV_LN PSERV_ON
  otherwise nop
End
PESP_OF  = c2d(SUBSTR(x.i,cfof+172,4))  /*Offset to ASID exp.storage */
                                           /*access policies section */
PESP_LN  = c2d(SUBSTR(x.i,cfof+176,2))  /*Length of each AS ESP entry*/
PESP_ON  = c2d(SUBSTR(x.i,cfof+178,2))     /* Num. of AS ESP entries */
Select
  when PESP_ON  > Ø then call ASESP PESP_OF PESP_LN PESP_ON
```

```
  otherwise nop
End
S2PCDCLOCK=c2d(SUBSTR(x.i,cfof+18Ø,2))     /* Policy adjustment count*/
    /* -down clock, no policy action is taken until clock is zero    */
S2PNH     =c2d(SUBSTR(x.i,cfof+182,1))  /*Period experienced processor*/
   /* access or storage delay during last policy adjustment interval */
Select
 when S2PNH   > Ø then d9=" Othr/"||S2PNH
 otherwise             d9=""
End
S2PRTP    =c2d(SUBSTR(x.i,cfof+183,1))        /* Response time goal  */
                       /* percentile. Zero if period does not have a */
                                  /* percentile response time goal. */
S2PAUXS   =c2d(SUBSTR(x.i,cfof+184,4))     /* Shared storage paging  */
                                           /* samples from aux       */
S2PIOU    =c2d(SUBSTR(x.i,cfof+188,4))     /* I/O using samples      */
S2PIOD    =c2d(SUBSTR(x.i,cfof+192,4))     /* I/O delay samples      */
S2PIO_MDP =c2d(SUBSTR(x.i,cfof+196,4))  /* Maximum % of time period  */
                                  /* could demand I/O.% scaled by 1Ø*/
S2PIODP   =c2d(SUBSTR(x.i,cfof+2ØØ,1))     /* I/O priority           */
S2FLAGS =x2b(c2d(SUBSTR(x.i,cfof+2Ø1,1)))  /* Flags                  */
     w1 =  substr(S2FLAGS,1,1)             /* Period experienced     */
              /* some type of delay within the sysplex during last   */
              /* policy adj.interval Period experienced              */
     w2 =  substr(S2FLAGS,2,1)             /* Period is CPU critical */
     w3 =  substr(S2FLAGS,3,1)             /* Period belongs to a    */
       /* service class that was assigned storage protection         */
       /* (stg critical) in the active service policy. The service   */
       /* class was used in subsystem type CICS or IMS and the rule  */
       /* specified storage critical= yes. Also on for               */
       /* transactionserver DISPs serving protected service classes. */
  if w1 = 1         then Help ="Needs sysplex help/"
     else if w1 = Ø then Help =""
  if w2 = 1         then CpuC ="Cpu critical/"
     else if w2 = Ø then CpuC =""
  if w3 = 1         then Stci ="Stor crit_implicit"
     else if w3 = Ø then Stci =""
flagging=Help||CpuC||Stci
Select
  when w1 = 1         then Hlp ="Y"
  otherwise           Hlp ="N"
End
S2PDEVCL  =c2d(SUBSTR(x.i,cfof+2Ø4,4)) /*Identifier of device cluster*/
     /* period belongs to. This identifier can be used to             */
     /* associated the period with device cluster data in the         */
     /* subtype 4 record. Zero if the period is not associated with a*/
                                  /* device cluster                 */
S2PSERVER_TYPE = ,             /* Server type. No bits will be on if */
       x2b(c2d(SUBSTR(x.i,cfof+2Ø8,4)))  /* period is not a server   */
       z1 =  substr(S2PSERVER_TYPE,1,1)
```

```
       z2 =  substr(S2PSERVER_TYPE,2,1)
       z3 =  substr(S2PSERVER_TYPE,3,1)
  Select
   when z1 =1 then Server ="Transaction server"
   when z2 =1 then Server ="Enclave server    "
   when z3 =1 then Server ="Queue server      "
   otherwise        Server =
  End
PSDATA_OF = c2d(SUBSTR(x.i,cfof+212,4))  /* Offset to server samples */
PSDATA_LN = c2d(SUBSTR(x.i,cfof+216,2))  /* sec. Length of each      */
                                           /* server samples entry   */
PSDATA_ON = c2d(SUBSTR(x.i,cfof+218,2))    /* Num. of server samples */
                                           /* entries                */
Select
when PSDATA_ON > 0 then call SERVS  PSDATA_OF PSDATA_LN PSDATA_ON
  otherwise nop
End
PQDATA_OF = c2d(SUBSTR(x.i,cfof+220,4))
                                    /* Offset to queue server section */
PQDATA_LN = c2d(SUBSTR(x.i,cfof+224,2)) /*Length of each queue server*/
                                           /* entry                  */
PQDATA_ON = c2d(SUBSTR(x.i,cfof+226,2)) /*No. of queue server entries*/
Select
  when PQDATA_ON > 0 then call QSERV PQDATA_OF PQDATA_LN PQDATA_ON
  otherwise nop
End
S2PAVG_SIZE = c2d(SUBSTR(x.i,cfof+228,4)) /*Average size in processor*/
        /* storage (frame # ) of ASIDs in the period. zero otherwise */
S2PGRN      =     SUBSTR(x.i,cfof+232,8)   /* Group name or blank if */
                                  /* period doesn't belong to a group */
S2PSYS_CPUU = c2d(SUBSTR(x.i,cfof+240,4))
                                    /* Sysplex-wide CPU using samples */
S2PSYS_NONIDLE = ,                  /* Sysplex-wide non-idle          */
              c2d(SUBSTR(x.i,cfof+244,4))  /* samples                */
S2PSYS_IDLE = c2d(SUBSTR(x.i,cfof+248,4)) /*Sysplex-wide idle samples*/
S2PSYS_OTHER= c2d(SUBSTR(x.i,cfof+252,4))
                                       /* Sysplex-wide other samples */
IOSUBSAMOF= c2d(SUBSTR(x.i,cfof+256,4))   /* Offset to I/O subsystem */
                                           /* samples data           */
IOSUBSAMLN= c2d(SUBSTR(x.i,cfof+260,2))  /*Length of a I/O subsystem */
                                           /* samples data section   */
IOSUBSAMON= c2d(SUBSTR(x.i,cfof+262,2))    /* Num. of I/O subsystem  */
                                           /* samples data sections  */
S2SPMDP     = c2d(SUBSTR(x.i,cfof+264,4))  /* Saved copy of maximum  */
                                    /* % of processor time demanded   */
S2SWCT      = c2d(SUBSTR(x.i,cfof+276,4))
                                    /* Short wait count accumulator   */
S2NUM_SAMP_HIST_ROWS_USED = ,       /* Num. of sample history rows    */
              c2d(SUBSTR(x.i,cfof+282,2))  /*used to build sample set*/
S2CADP      = c2d(SUBSTR(x.i,cfof+284,4))
```

```
                                    /* Current Achievable demand %    */
S2SBCPUU    = c2d(SUBSTR(x.i,cfof+288,4))  /* Sample based CPU usings*/
S2SBCPUD    = c2d(SUBSTR(x.i,cfof+292,4))  /* Sample based CPU delays*/
S2PSYS_IO_DLY = ,                          /* Sysplex-wide IO delay  */
              c2d(SUBSTR(x.i,cfof+296,4))
Select
 when S2PSYS_IO_DLY  > 0 then d10=" SYSio/"||S2PSYS_IO_DLY
 otherwise              d10=""
End
S2PSYS_NON_IO_DLY = ,                   /* Sysplex-wide non-I/O delay */
              c2d(SUBSTR(x.i,cfof+300,4))
Select
 when S2PSYS_NON_IO_DLY> 0 then d11=" SYSoth/"||S2PSYS_NON_IO_DLY
 otherwise              d11=""
End
 Del=d9||d0||d1||d2||d3||d4||d5||d6||d7||d8||d10||d11
s2rec.1= left(S2PCNM,8)        left(Hlp,4)   right(S2PNUM,4),
         right(S2PIMPOR,3)     left(Goal,10),
         right(S2PSPI/100,4)   right(S2PLPI/100,4),
         right(S2PBDP,5)       right(S2PIODP,5),
         right(S2PSERV,6)      right(S2PMDP,4) ,
         right(S2PMPLI,5)      right(S2PMPLO,5),
         right(S2PPSPT/1000,8) right(PSI,4),
         right(expp,18)        left(Del,45)
   "EXECIO * DISKW s991 (STEM s2rec.)"
end     /* of extent loop*/
end     /* of period loop*/
return
/* subtype 2 main loop - end  */
XMEM:
parse arg off len num
if (len <> 0) Then do
  do zp = 0 to num -1
   ofp  = (off + (zp*len))- 3
    /*-----------------------------------------------------------------*/
   /*  SMF99 subtype 2: Cross memory delay data                       */
    /*-----------------------------------------------------------------*/
S2XMEM_JOBN  =    SUBSTR(x.i,ofp,8)   /* ASID name causing XMEM delay*/
S2XMEM_SAMPS =c2d(SUBSTR(x.i,ofp+8,4))     /* Number of xmem samples */
end
Return
SERV:
parse arg off len num
if (len <> 0) Then do
  do zy = 0 to num -1
   ofo  = (off + (zy*len))- 3
    /*-----------------------------------------------------------------*/
   /*  SMF99 subtype 2: Server Data Entry Section                     */
    /*-----------------------------------------------------------------*/
S2SERCNM   =     SUBSTR(x.i,ofo,8)  /* Class that period belongs to */
```

```
   /*If the service class is a server, then this is the name of the */
   /* service class being served. If the service class is being     */
   /*served, then this is the name of the server service class.     */
S2SERPNUM=c2d(SUBSTR(x.i,ofo+8,4))  /*       Service period No.     */
S2SEROBS =c2d(SUBSTR(x.i,ofo+12,4)) /*Percentage of time this server*/
                                    /*       was serving the period */
end
Return
SERVS:
parse arg off len num
if (len <> 0) Then do
  do zy = 0 to num -1
   ofo  = (off + (zy*len))- 3
    /*-----------------------------------------------------------------*/
   /* SMF99 subtype 2: Server Sample Data Entry Section              */
    /*-----------------------------------------------------------------*/
S2SDAWQDEL   =c2d(SUBSTR(x.i,ofo,4))     /* Delay samples waiting on */
                                         /* WLM-managed work queue   */
S2SDAENC_AUX =c2d(SUBSTR(x.i,ofo+4,4))   /* Aux private paging delay */
     /* samples experienced by enclave work units known to be        */
     /* associated with an ASID                                      */
S2SDAENC_VIO =c2d(SUBSTR(x.i,ofo+8,4))   /* Aux VIO paging delay     */
            /* samples experienced by enclave work units known to be */
                            /* associated with an ASID              */
S2SDAENC_HSP =c2d(SUBSTR(x.i,ofo+12,4))  /* Aux standard hiperspace  */
           /* paging delay samples experienced by enclave work units */
                       /* known to be associated with an ASID       */
S2SDAENC_MPLD=c2d(SUBSTR(x.i,ofo+16,4))  /* MPL delay samples        */
                  /* experienced by enclave work units known to be   */
                                /* associated with an ASID          */
S2SDAENC_ASWP=c2d(SUBSTR(x.i,ofo+20,4))  /* Aux swap delay samples   */
         /* experienced by enclave work units known to be associated */
                        /* with an  ASID                            */
S2SSERVER_CLNAME=SUBSTR(x.i,ofo+24,8)   /*Class name of disp serving */
                                  /* this period or blank for batch */
S2SSERVER_TYPE= ,                       /* Type byte                */
         c2d(SUBSTR(x.i,ofo+32,1))
Select
when S2SSERVER_TYPE='128' then Server ='Enclave/queue'
when S2SSERVER_TYPE='64'  then Server ='Batch queue'
otherwise nop
End
S2SDASUBSYS_TYPE =SUBSTR(x.i,ofo+36,4)   /*Subsystem type of owner of*/
                 /* the queue. Only applies to batch queue servers.  */
S2SDA_SUBSYS_NAME=SUBSTR(x.i,ofo+40,8)   /*Subsystem name of owner of*/
                 /* the queue. Only applies to batch queue servers.  */
sdrec.y = left(date('n',SMF99DTE,'j'),11),
          left(SMF99TME,12) right(TPID,10),
          left(S2SSERVER_CLNAME,11)   left(SERVER,14),
          right(S2SDAWQDEL,2)         right(S2SDAENC_AUX,4),
```

```
          right(S2SDAENC_VIO,5)       right(S2SDAENC_HSP,5),
          right(S2SDAENC_ASWP,5)      right(S2SDAENC_MPLD,5)
y = y +1
end
Return
QSERV:
parse arg off len num
if (len <> Ø) Then do
  do zy = Ø to num -1
   ofo  = (off + (zy*len))- 3
   /*-------------------------------------------------------------*/
   /*  SMF99 subtype 2: Queue Server Data Entry Section           */
   /*-------------------------------------------------------------*/
S2QD_ENV_NAME =    SUBSTR(x.i,ofo,32)    /* Associated env. name for */
                                         /* the queue               */
S2QD_SERVER_CLASS_NAME = ,       /* Class name of disp serving the   */
                  SUBSTR(x.i,ofo+32,8)   /*period represented by this*/
       /* subt.2 record. Only applies to queue manager type servers. */
S2QD_SERVER_WANT  = ,            /* No. of server instances needed to*/
              c2d(SUBSTR(x.i,ofo+4Ø,4))       /* address queue delay */
      /* according to policy adjustment. This is a queue-wide count. */
S2QD_SERVER_HAVE  = ,                /* Current actual no. of server */
              c2d(SUBSTR(x.i,ofo+44,4))  /* instances bound to the   */
                                /* queue. This is a queue-wide count. */
S2QD_SERVER_ACTIVE = ,               /* Current actual no. of server */
              c2d(SUBSTR(x.i,ofo+48,4))  /* instances bound to the   */
         /* queue and between Begin and End. This is a subset of the */
                /* Have count. Have minus Active equals Idle.        */
S2QD_AS_CAPACITY = ,             /* ASID server instance capacity    */
              c2d(SUBSTR(x.i,ofo+52,4))  /* instances bound to the   */
S2QD_ACHIEVED_QMPL = ,          /* queue and Average no.(over policy */
              c2d(SUBSTR(x.i,ofo+56,4))       /* interval) of server */
       /* instances that are in swapped in spaces in the DISP.Only   */
       /* counts server instances serving the external service class */
       /* associated with the queue. Scaled by 16. Not used for      */
                            /*batch queue servers.                   */
S2QD_ACTIVE_QMPL = ,                     /* Average of no.of server  */
              c2d(SUBSTR(x.i,ofo+6Ø,4))  /* instance between Begin   */
      /* and End during policy interval. Scaled by 16. For batch     */
      /* queue servers this is the no. of initiators with            */
                               /* active jobs sysplex-wide.         */
S2QD_QMPL_IN_TAR = ,                 /* No. of ASIDs suggested to be */
              c2d(SUBSTR(x.i,ofo+64,4))  /* started in the DISP on   */
      /* behalf of the period represented by this subtype does not   */
                 /* apply to batch queue servers.                   */
S2QD_AVG_QUEUED_REQUESTS  = ,      /* Average no. of queued requests */
              c2d(SUBSTR(x.i,ofo+68,4))  /* over a policy interval   */
S2QD_LT_TOTAL_REQUESTS = ,
                     /* scaled(16) Long term average total requests */
              c2d(SUBSTR(x.i,ofo+72,4))
```

```
                                  /* (queued + active) scaled by 16   */
S2QD_SERVER_IDLE = ,                 /* Average idle server instances */
             c2d(SUBSTR(x.i,ofo+76,4))  /* over policy interval     */
S2QD_Q_TYPE = c2d(SUBSTR(x.i,ofo+8Ø,1))  /* Work Queue type byte     */
Select
when S2QD_Q_TYPE   ='128' then WrkQueue ='Queue manager'
when S2QD_Q_TYPE   ='64'  then WrkQueue ='Batch work queue'
otherwise nop
End
S2QDWQQ= c2d(SUBSTR(x.i,ofo+81,1))      /* Work Queue Qualifier      */
Select
when S2QDWQQ ='128' then WrkQueueQ ='Tasks WLM managed'
                                  /* Server instances are WLM managed */
when S2QDWQQ ='64'  then WrkQueueQ ='Spaces moved by HSM'
       /* ASIDs have been moved from this work queue to enforce the  */
       /* minimum no. of servers of another work queue of the same   */
                                  /* app. env.                        */
when S2QDWQQ ='32'  then WrkQueueQ ='Spaces moved by PA'
      /* ASIDs have been moved during policy adjustment because the  */
      /* maximum no. of servers has been already started for the     */
                                                 /* app. env.        */
when S2QDWQQ ='16'  then WrkQueueQ ='Spread admin'
       /* Minimum no. of ASIDs must be distributed across all        */
                                  /* work queues of the app.env.      */
otherwise    WrkQueueQ =' '
End
S2QD_ACTIVE_RGNWORK = ,           /* Current active no. of server     */
             c2d(SUBSTR(x.i,ofo+82,2))  /* processing work requests */
       /* which have been routed directly to the server region. This */
       /* no. is neither included in the idle nor the active server  */
                                                  /* count.          */
  QD_RQDATA_OF = ,                /* Offset to remote queue data      */
             c2d(SUBSTR(x.i,ofo+84,4))  /* section. Only applies to */
                                        /* batch queue server       */
  QD_RQDATA_LN = ,                      /* Length of remote queue   */
             c2d(SUBSTR(x.i,ofo+88,2))
  QD_RQDATA_ON = ,                /* No. of remote queue data entries */
             c2d(SUBSTR(x.i,ofo+9Ø,2))
Select
when QD_RQDATA_ON >Ø then call RQS  QD_RQDATA_OF QD_RQDATA_LN
QD_RQDATA_ON
  otherwise nop
End
S2QD_SUBSYS_TYPE = ,              /* Subsystem type of owner of       */
                  SUBSTR(x.i,ofo+92,4)   /*the queue. Only applies to*/
                                         /* batch queue servers.     */
S2QD_SUBSYS_NAME = ,                    /* Subsystem name of owner of  */
                  SUBSTR(x.i,ofo+96,8)  /* the queue. Only applies to*/
                                         /* batch queue servers.     */
S2QD_INST_PER_SERVER  = ,         /* No. of server instances/server.  */
```

```
              c2d(SUBSTR(x.i,ofo+1Ø4,2))
S2QD_SPACES_MOVED = ,                   /* No. of server ASIDs moved  */
              c2d(SUBSTR(x.i,ofo+1Ø6,2)) /* away from this queue.    */
S2QD_AE_MAXLIMIT = ,              /* Max.no. of servers for app. env. */
              c2d(SUBSTR(x.i,ofo+1Ø8,4))
S2QD_AE_MINLIMIT = ,              /* Min.no. of servers for  app. env.*/
              c2d(SUBSTR(x.i,ofo+112,2))
S2QD_AVG_INELIGIBLE_REQUESTS = ,        /* Average no. of ineligible  */
              c2d(SUBSTR(x.i,ofo+114,2)) /* queued requests over a   */
                              /* policy interval scaled by 16.    */
                         /* Currently applies to batch queues only.  */
sqrec.a = left(date('n',SMF99DTE,'j'),11),
          left(SMF99TME,12) right(TPID,1Ø),
          left(S2QD_SERVER_CLASS_NAME,9) left(S2QD_ENV_NAME,9),
          left(WrkQueue,15),
          right(S2QD_AVG_QUEUED_REQUESTS,3),
          right(S2QD_LT_TOTAL_REQUESTS,3),
          center(S2QD_AVG_INELIGIBLE_REQUESTS,8),
          right(S2QD_SERVER_WANT,5),
          right(S2QD_SERVER_HAVE,5),
          right(S2QD_SERVER_ACTIVE,5),
          right(S2QD_SERVER_IDLE,5),
          right(S2QD_INST_PER_SERVER,5),
          right(S2QD_ACHIEVED_QMPL,4),
          right(S2QD_ACTIVE_QMPL,5),
          right(S2QD_QMPL_IN_TAR,5),
          right(S2QD_AS_CAPACITY,12),
          right(S2QD_SPACES_MOVED,8),
          right(S2QD_ACTIVE_RGNWORK,6),
          right(S2QD_AE_MAXLIMIT,6),
          right(S2QD_AE_MINLIMIT,6)
a = a + 1
end
Return
ASESP:
parse arg off len num
if (len <> Ø) Then do
  do zk = Ø to num -1
   ofo  = (off + (zk*len))- 3
    /*-----------------------------------------------------------------*/
   /*  SMF99 subtype 2 Address Space Expanded Storage                 */
   /*  access policy                                                  */
    /*-----------------------------------------------------------------*/
S2AS_ANAM=    SUBSTR(x.i,ofo,8)          /* Address space name       */
S2AS_AP=    c2d(SUBSTR(x.i,ofo+8,1))  /*ESP access policy for demand */
Select
when S2AS_AP ='Ø' then Demand ='Protected'
when S2AS_AP ='1' then Demand ='LRU'
when S2AS_AP ='2' then Demand ='Space available'
otherwise              Demand =' '
```

```
End
S2AS_VP=    c2d(SUBSTR(x.i,ofo+9,1))     /* ESP access policy for VIO*/
Select
when S2AS_VP ='0' then VIO    ='Protected'
when S2AS_VP ='1' then VIO    ='LRU'
when S2AS_VP ='2' then VIO    ='Space available'
otherwise              VIO    =' '
End
S2AS_HP=    c2d(SUBSTR(x.i,ofo+10,1))
                               /* ESP access policy for hiperspace */
Select
when S2AS_HP ='0' then HiperSp    ='Protected'
when S2AS_HP ='1' then HiperSp    ='LRU'
when S2AS_HP ='2' then HiperSp    ='Space available'
otherwise              HiperSp    =' '
End
S2AS_ASID=c2d(SUBSTR(x.i,ofo+11,2))      /* Address space ID        */
S2AS_FLAGS= c2d(SUBSTR(x.i,ofo+13,1))    /* Flags                   */
Select
when S2AS_FLAGS='128' then ASflag ='AS stg protected now'
   /* Storage is protected at this instant protected at this instant*/
when S2AS_FLAGS='64'  then ASflag ='AS stgcrit explicit'
   /* Storage protected assigned to space by classification rule    */
when S2AS_FLAGS='32'  then ASflag ='AS managed to region goals'
                           /* Address space is currently managed   */
                           /* to region's goal rather than         */
                           /* transaction server's goal            */
otherwise                ASflag ='Reserved'
End
```

Editor's note: this article will be concluded next month.

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Reader's letter

I am writing with reference to the August 2004 *Mainframe Month* (www.xephon.com/mfm) topic *Editing a new member when in Option 3.4* (or see *MVS Update*, Issue 191, August 2002).

I would like to add that, in a similar manner to the proposed

solution, you could create new members in a PDS like this in Option 3.4:

```
E          DATA.SET.NAME(newmember)
```

press *Enter/return*.

It will open up the PDS member(new), where we can type the required code and then save it.

Vinod KR
vinod_engg@rediffmail.com

Mainstar Software has announced Release 8.101 of its FastAudit/390 Suite.

FastAudit/390 Suite is a set of metadata interrogation tools designed to keep the OS/390 environment trouble-free with audits. HSM FastAudit-MediaControls allows users to audit their HSM migration and back-up tapes; HSM FastAudit evaluates the Migration Control Dataset (MCDS), Back-up Control Dataset (BCDS), and the Offline Control Dataset (OCDS) to resolve structural and logical discrepancies that can prevent migrated data from being recalled, or back-up data from being restored; Catalog and DASD Audit evaluates all master and user catalog dataset entries, and reconciles against all DASD VTOC information on currently-mounted volumes; and Tape Audit correlates entries in the tape management catalog with ICF catalog status.

New enhancements include a complete INI file and JCLBUILD process as well as a fast VTOC and TMC reader.

For further information contact:
Mainstar Software, PO 4132, Bellevue, WA 98009-4132, USA.
Tel: (425) 455 3589.
URL: www.mainstar.com/pdf/006-0107_MFA8101_RA.pdf.

* * *

BMC Software has announced Version 6.1.03 of Batch Impact Manager, a dynamic batch management product designed so customers can proactively manage batch processes according to their importance to the business.

Batch Impact Manager provides a picture of the status of batch business services. It also proactively detects potential delays as well as errors in the batch business process, so that corrective actions take place before the business service is affected.

Batch Impact Manager allows the definition of 'must complete by' times for critical jobs; scans predecessor/successor chains to determine and analyse critical paths; and provides early notification of potential batch delays.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042, USA.
Tel: (713) 918 8800.
URL: www.bmc.com/supportu/hou_Support_ProdVersion/0,3648,19097_0_102414_0,00.html.

* * *

Serena Software has announced Version 5.5 of Serena StarTool DA, which helps to diagnose the causes of mainframe outages. The new version utilizes XML services to provide integration with Serena ChangeMan, its change management software.

For further information contact:
Serena Software, 2755 Campus Drive, 3rd Floor, San Mateo, CA 94403, USA.
Tel: (650) 522 6600.
URL: www.serena.com/Products/startool/home.asp

* * *

Laid Back Software has announced Version 2.8.5 of TSO Environment and Administration Manager (TEAM). This gives TSO users the ability to customize their environment without involving support staff. Administrative personnel can use the tools to assist TSO users.

URL: pages.sbcglobal.net/kemoe/LaidBackSoftware/team.html.