217

MVS update

October 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.

Printed in England.

X-ray browse

This REXX is called with the parameters libtype, membername, and type of access wanted (edit or browse). It captures the output of a LISTA command and searches for the libtype specified. When the libtype is found, it checks whether the member is found in a library concatenated to this DDname, and lets you browse (default) or edit the member in the library where it is found.

I would like to get an improved version of the functionality, not necessarily a REXX, that checks the APPLID and searches any LIBDEFed, ALTLIBed, or STEPLIBXed datasets as well, and in the correct order (I'm sure it is out there somewhere, and I hope it is independent of TSO/ISPF-release).

```
/* REXX

   Does an "X-ray" browse on your allocations
   on the specified library for the element and finds it in
   the first dataset on the DDname.
   Called by : XB y element e
                                                  KMI                    */
Address ISPEXEC
"vget (ktrac)"      /* If you VPUT XB in KTRAC you will have a trace */
if ktrac = sysvar(sysicmd) then trace i
Address TSO
parse upper arg lib member edit
if lib = "" then do
   zedlmsg = ,
"Format: XB Libtype member (E) Libtypes : P - Panel,R - REXX/CLIST,   ",
"M - MSGs, S - Skeleton, T - Table, L - Steplib/ISPLLIB. E - EDIT    "
   Address ISPEXEC "setmsg msg(isrz001)"
   exit
end
lib2 = "FNIDDER"
select
  when lib = "C" | lib = "R" then do     /* 2 DDnames for CLIST/REXX */
       lib = "SYSEXEC"
       lib2 = "SYSPROC"
  end
  when lib = "P" then lib = "ISPPLIB"
  when lib = "M" then lib = "ISPMLIB"
  when lib = "T" then lib = "ISPTLIB"
```

```
  when lib = "L" | lib = "E" then do      /* 2 DDnames for PGMs     */
       lib = "ISPLLIB"
       lib2 = "STEPLIB"
  end
  when lib = "S" then lib = "ISPSLIB"
  otherwise nop
end
x=outtrap('line.')                      /* Get allocations            */
"lista st"
y=outtrap('off')
more = "1"
hit = "Ø"
first = 1
do i = 1 to line.Ø while ¬hit           /* Find DD-name               */
   if word(line.i,1) = lib | word(line.i,1) = lib2 then do
      do h = (i - 1) to line.Ø by 2 while more       /* Get dataset   */
      dsn = word(line.h,1)
      dsn = "'"||dsn||"("member")'"
      if sysdsn(dsn) = "OK" then do      /* Hello are You there ?     */
         hit = "1"
         if edit ¬= "E" then ,
         Address ISPEXEC ,
         "BROWSE DATASET("dsn")"         /* Then Browse           */
         else ,
         Address ISPEXEC ,
         "EDIT DATASET("dsn")"           /* Then Edit             */
         more = "Ø"
         hit = "1"
      end
      j = h + 3
      if words(line.j) ¬= 1 then more = "Ø"
      end
      first = first + 1
   end
   if ¬hit & lib2 ¬= "FNIDDER" & first = 2 then more = "1"
                                        /* Cheating with 2 DDnames  */
end
if ktrac = sysvar(sysicmd) then trace i
if ¬hit then do
   zedsmsg = "Member" member "not found on" lib
   zedlmsg = "Member" member "not found on" lib
   Address ISPEXEC "setmsg msg(isrzØØ1)"
end
```

Kim Michaelsen
Systems Programmer (Denmark)

Using indirect volume serial support

When installing z/OS, it could well be worth your while to use '&xxxxx' for the 'VOLUME' field while cataloguing datasets in the master catalog.

With this syntax, when a catalog entry is retrieved, the variable is translated into the volume serial number of the system residence pack (or its logical extension).

The procedure in this article illustrates how to carry out the conversion from within the current operating system.

Once the activities have completed, the master catalog will contain non-VSAM datasets (PO, PS, PDS, PDSE), all of which have a static system symbol as the volume serial number.

It should be pointed out that this technique, called 'extended indirect cataloguing', allows:

- Easy cloning of an operating system.
- Carrying out a simple restore of the master catalog in each partition with the same release.
- Sharing of master catalogs among multiple images that use different volumes with different names for the system residence volumes and their extensions.

In order to activate this feature, the sequence of operations is as follows:

- Insert the keyword IEASYM*xx* in member LOAD*xx*.

  For example:

```
----+----1----+----2----+----3----+----4----+----5----
..........
IEASYM   (Øx,L)
..........
```

- Define the necessary static system symbols in

sys1.parmlib, member IEASYM*xx*. It should be noted that the symbol '&SYSR1' always indicates the IPL volume.

For example:

```
.............

SYSDEF  SYMDEF(&SYSR2='&SYSR1(1:5).2')
SYSDEF  SYMDEF(&SYSR3='&SYSR1(1:5).3')
.............
```

- Perform an IPL.

From this point on, it will be possible to take advantage of the indirect volume serial support feature.

For instance, it is possible to catalog once more the sys1.linklib library with this DEFINE/IDCAMS:

```
DEFINE NVSAM(NAME(sys1.linklib)          -
        DEVT(0000) VOLUMES(&sysr1))      -
        CATALOG(your_master_catalog)
```

The '&sysr1' variable will point to the symbol defined in PARMLIB library member IEASYM*xx*.

In order to extend the symbolic values to all non-VSAM datasets in the master catalog, we can launch the following source JCL:

```
//.........JOB ........................................
//* -------------------------------- *
//STEP001  EXEC PGM=IKJEFT1A,PARM='%REXICFID'
//SYSEXEC  DD   DISP=SHR,DSN=your_sysexec_library
//SYSIN    DD   *
 LISTC CAT(your_master_catalog)                    -
       NVSAM                                       -
        VOL                                        -
         OFILE(FIOUT)
/*
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DUMMY
//SYSINSYM DD   *
----+----1----+----2----+----3
- FROM   - TO
----+----1----+----2----+----3
SYSRE1   &SYSR1
SYSRE2   &SYSR2
...............
/*
```

where:

- The values for the volumes to be converted from explicit to symbolic are indicated on the JCL/SYSINSYM card.
- The name of the master catalog is inserted on the JCL/SYSIN card.

At the end of the job's execution, two datasets will be written:

- &SYSUID.SYSINDEF – with all the 'DEFINE ...' commands to modify the VOLSER values.
- &SYSUID.ERRORDSN – created only in case of unrecoverable errors.

This JCL calls a procedure (REXICFID) written in REXX. The following is the source code to the REXX procedure:

```
/* REXX ---------------------------------------------------------REXX */
/* REXX FUNCTION: REXICFID 2004/V1R0M0                          REXX */
/* REXX ---------------------------------------------------------REXX */
/* LIST OF ERROR CODE:                                            */
/*  25   >   ALLOCATION ERROR: DDNAME (FIOUT)                     */
/*  26   >   "           "    : "      (FIDEF)                    */
/*  27   >   "           "    : "      (FIERR)                    */
/*  28   >   "           "    : "      (SYSINSYM)                 */
/*  35   >   IDCAMS/"LISTC CAT.....  " ERROR                      */
/*  36   >   SYMBOLIC NAMES DON'T EXIST                           */
/*  37   >   ICF CATALOG HAS NO NONVSAM ENTRIES                   */
/*  38   >   NO CHANGES DONE FOR NONVSAM ENTRY                    */
/* ------------------------------------------------------------- */
PARSE UPPER ARG DEBUG
/* ------------------------------------------------------------- */
/* START                                                         */
/* ------------------------------------------------------------- */
IF DEBUG = 'YES'
   THEN DO
   TRACE ALL
   MSG_STATUS = MSG("ON")
        END
       ELSE DO
         MSG_STATUS = MSG("OFF")
         TRACE OFF
            END
 " PROF NOPREF "
W_CTRWRK = 0                              /*WORK ITEM          */
W_CTRSYM = 0                              /*SYSINSYM COUNTER   */
```

```
W_RECDSN = Ø                                /*NVSAM FILE COUNTER  */
W_RECSYM = Ø                                /*SYMBOLIC VOL COUNTER*/
W_RECERR = Ø                                /*ERROR  MSG COUNTER  */
/* ---------------------------------------------------------- */
/* ALLOCATION/WORK_DATASET                                    */
/* ---------------------------------------------------------- */
FIDEF    =  SYSVAR('SYSUID').SYSINDEF     /* FILE FOR "DEFINE"  */
FIERR    =  SYSVAR('SYSUID').ERRORDSN     /* ERROR FILE         */
FIOUT    =  SYSVAR('SYSUID').OUTLISTC     /* OUTPUT "LISTC CAT" */
   " DEL '"FIDEF"' "                        /* RC <= 9 IS GOOD   */
   " DEL '"FIERR"' "                        /* RC <= 9 IS GOOD   */
   " DEL '"FIOUT"' "                        /* RC <= 9 IS GOOD   */
     " FREE FI(FIOUT) "
     "ALLOC FI(FIOUT) UNIT(VIO) RECFM(V B A) LRECL(125)
           SPACE (2 2) TRACKS NEW DELETE"  /* TEMP      FILE    */
       IF RC > Ø THEN EXIT(25)
     " FREE FI(FIDEF) "
   " ALLOC FI(FIDEF) DA('"FIDEF"') RECFM(F B) LRECL(8Ø)
     SPACE (1 1) TRACKS NEW CATALOG "      /* PERMANENT FILE    */
       IF RC > Ø THEN EXIT(26)
     " FREE FI(FIERR) "
   " ALLOC FI(FIERR) DA('"FIERR"') RECFM(F B) LRECL(8Ø)
     SPACE (1 1) TRACKS NEW CATALOG "      /* PERMANENT FILE    */
       IF RC > Ø THEN EXIT(27)
/* ---------------------------------------------------------- */
/* IDCAMS/LISTC                                               */
/* ---------------------------------------------------------- */
   "FREE  F(SYSPRINT)"
   "ALLOC F(SYSPRINT) DUMMY"
   ADDRESS  "LINKMVS" "IDCAMS"
       IF RC  <> Ø  THEN EXIT(35)
/* ---------------------------------------------------------- */
/* OPEN/WORK_DATASET                                          */
/* ---------------------------------------------------------- */
          "EXECIO Ø DISKW FIDEF    (OPEN"  /* OPEN DATASET      */
          "EXECIO Ø DISKW FIERR    (OPEN"  /* OPEN DATASET      */
          "EXECIO Ø DISKR FIOUT    (OPEN"  /* OPEN DATASET      */
/* ---------------------------------------------------------- */
/* USE SYSTEM SYMBOL                                          */
/* ---------------------------------------------------------- */
          "EXECIO Ø DISKR SYSINSYM (OPEN"  /* OPEN DATASET      */
            IF RC > Ø THEN EXIT(28)
DO FOREVER
          "EXECIO 1 DISKR SYSINSYM"        /* READ  DATASET     */
            IF RC > Ø THEN LEAVE           /* PHISICAL EOF      */
             PULL SYSINSYM
        IF INDEX(SYSINSYM,' &') <> Ø THEN
          DO
           W_CTRWRK = W_CTRWRK + 1
            SYMVOL.W_CTRWRK = SUBSTR(SYSINSYM,1Ø,Ø6)
```

```
            ESPVOL.W_CTRWRK = SUBSTR(SYSINSYM,Ø1,Ø6)
             END
END
       IF  W_CTRWRK = Ø THEN EXIT(36)
W_CTRSYM = W_CTRWRK
/* ---------------------------------------------------------- */
DO FOREVER
          "EXECIO 1 DISKR FIOUT "          /* READ  DATASET     */
            IF RC > Ø THEN LEAVE           /* PHYSICAL EOF      */
             PULL FIOUT
              TGTXXX = SUBSTR(FIOUT,18,44)
               TGTDSN = STRIP(TGTXXX,'T')
                END_LISTC = INDEX(FIOUT,' ENTRIES PROCESSED WAS:')
                 IF END_LISTC <> Ø THEN LEAVE   /* LOGICAL EOF  */
    SELECT
       WHEN INDEX(FIOUT,'ING FROM CATALOG') <> Ø THEN
             TGTICF = SUBSTR(FIOUT,55,44)
       WHEN INDEX(FIOUT,'ØNONVSAM ------')  = Ø THEN
             NOP
       WHEN INDEX(TGTDSN,'SYS1.VVDS.V')     <> Ø THEN
             NOP
       WHEN INDEX(TGTDSN,'SYS1.VTOCIX.')    <> Ø THEN
             NOP
       OTHERWISE
             CALL RTLIST
    END                                        /* END SELECT   */
END                                            /* END DO       */
MLEND:
          "EXECIO Ø DISKW FIDEF    (FINIS"  /* CLOSE DATASET    */
          "EXECIO Ø DISKW FIERR    (FINIS"  /* CLOSE DATASET    */
          "EXECIO Ø DISKW FIOUT    (FINIS"  /* CLOSE DATASET    */
          "EXECIO Ø DISKW SYSINSYM (FINIS"  /* CLOSE DATASET    */
          "FREE DATASET("FIERR")"
          "FREE DATASET("FIDEF")"
          "FREE DATASET("FIOUT")"
        W_RECDSN = RIGHT(W_RECDSN,7,'Ø')
        W_RECSYM = RIGHT(W_RECSYM,7,'Ø')
        W_RECERR = RIGHT(W_RECERR,7,'Ø')
    W_CTRWRK = Ø
  DO W_CTRSYM
    W_CTRWRK = W_CTRWRK + 1
       SAY "* ---------------------------------------- * "
       SAY "  REQUESTED VOLUME CHANGES FROM: " ESPVOL.W_CTRWRK
       SAY "            TO SYMBOLIC VARIABLE: " SYMVOL.W_CTRWRK
  END
       SAY "* ---------------------------------------- * "
       SAY "                        ICF CATALOG: " TGTICF
       SAY "  TOTAL NUMBER OF NONVSAM DATASETS: " W_RECDSN
       SAY "  DATASETS WITH NEW CATALOG ENTRY : " W_RECSYM
       SAY "  TOTAL NUMBER OF ERROR RECORDS   : " W_RECERR
```

```
       SAY "* ---------------------------------------- * "
       IF ABS(W_RECERR) = Ø THEN " DEL '"FIERR"' " /* EMPTY FILE*/
       IF ABS(W_RECDSN) = Ø THEN EXIT(37)
       IF ABS(W_RECSYM) = Ø THEN EXIT(38)
  EXIT(ØØ)
/* ------------------------------------------------------------ */
RTLIST:
 W_RECDSN = W_RECDSN + 1                  /*NVSAM FILE COUNTER  */
 LZ = OUTTRAP(LINE.)
 "LISTC ENT("TGTDSN") VOLUME"
 RC_LIST = RC
         DO I =1 TO LINE.Ø
          POS_VOL = INDEX(LINE.I,'VOLSER------------')
            IF POS_VOL <> Ø
               THEN DO
                POS_VOL = POS_VOL + 18
                 TGTVOL  = SUBSTR(LINE.I,POS_VOL,6)
                    END                          /* END IF       */
         END                                     /* END DO       */
       IF RC_LIST = Ø THEN CALL RTLDEF
          ELSE CALL RTLERR
     RETURN                                    /* RETURN RTLIST */
/* ------------------------------------------------------------ */
RTLDEF:
  W_CTRWRK = Ø
  DO W_CTRSYM
    W_CTRWRK = W_CTRWRK + 1
    IF TGTVOL = ESPVOL.W_CTRWRK THEN
      DO
       TGTVOL = SYMVOL.W_CTRWRK
         W_RECSYM = W_RECSYM + 1        /*SYMBOLIC VOL COUNTER  */
          NEWSTACK
           QUEUE " DELETE ("TGTDSN") -"
           QUEUE "  CATALOG("TGTICF") -"
           QUEUE "   NOSCRATCH "
           QUEUE "    SET MAXCC = Ø"
           QUEUE " DEFINE NVSAM(NAME("TGTDSN")  -"
           QUEUE "        DEVT(ØØØØ) VOLUMES("TGTVOL")) -"
           QUEUE "  CATALOG("TGTICF")  "
           QUEUE "               "
           "EXECIO * DISKW FIDEF "
          DELSTACK
      END
  END
     RETURN
RTLERR:
  W_RECERR = W_RECERR + 1                  /*ERROR  MSG COUNTER  */
  FIERR =    '> ERROR FOR ENTRY     : 'TGTDSN
  PUSH FIERR
  "EXECIO 1 DISKW FIERR "
```

```
FIERR =    '> IDCAMS RETURN CODE  : 'RC_LIST
PUSH FIERR
"EXECIO 1 DISKW FIERR "
   RETURN
```


DB2 Information Integrator Classic Federation for z/OS

IBM acquired the assets of CrossAccess (mainframe data integration specialists) in October 2003, and later announced a new product – DB2 Information Integrator Classic Federation for z/OS. The first question that comes to mind is, how different is this from the DB2 Information Integrator? And the answer is DB2 II Classic Federation allows easy access to non-relational data sources like VSAM on a mainframe, with no additional programming effort. The impact of this can be better realized when we consider the fact that most mainframe data is still stored in non-relational file structures, and it is estimated that about 70% of the legacy data resides in VSAM files.

DB2 II Classic Federation for z/OS is worth considering as an alternative to traditional approaches used to expose mainframe data, for example:

- Custom programming enables mainframe data to be accessed from open systems, which involves additional personnel effort as well as maintenance issues.
- Mainframe ETL projects to populate your data marts, ODS, and data warehouse involve additional hardware, software, personnel costs, as well as maintenance issues.
- Maintaining a copy of the mainframe data on other platforms involves data integrity issues in addition to extra costs in resources.

- Migrating the legacy applications to relational databases on mainframes or other platforms is an expensive as well as a risky proposition.

The aim is to be able to access the data – relational or non-relational, structured and unstructured, private and public, mainframe and distributed – as if it is a single database. IBM DB2 II Classic Federation along with DB2 II lets you do just that, thereby expanding IBM's Enterprise Information Integration (EII) solution capability. It is worth noting that this tool can either be used independently or as an extension to the DB2 Information Integrator.

Now you do not have to migrate the legacy data from non-relational data sources like VSAM to a relational database just to expose them to the Web. Using DB2 II Classic Federation you can easily access the legacy data from VSAM, Software AG Adabas, CA-IDMS, and CA-Datacom. What is interesting is that you can do all this using simple direct SQL statements – SELECT, INSERT, UPDATE, and DELETE – and no additional programming is required on the mainframe.

DB2 II Classic Federation for z/OS uses a meta-data approach, mapping the physical data sources to appropriate relational structures. The applications, ranging from Web clients to a data warehouse, can access mainframe data from any type of source using standard approaches, such as JDBC, ODBC, and CLI. These applications or tools can be on any platform – Unix, Linux, Windows, and mainframes – and access the mainframe data without bothering about the source data nuances.

The two main concerns that come up with this type of access to mainframe data from anywhere are security and performance.

DB2 II Classic Federation works well with any of the security tools on your mainframe – RACF from IBM, TopSecret or ACF2 from CA – and hence the data is as secure as your current set-up.

Though the external view is the same irrespective of the data source, DB2 II Classic Federation promises good performance because internally it uses the native access for each of these data source types and exploits the optimization opportunities offered by each (basically a best-of-breed approach rather than a one-size-fits-all). In addition, techniques like multi-threading and pre-fetching of data are used to improve the performance and scalability of the solution.

DB2 II Classic Federation provides a simple UI to do the mapping of legacy data source to relational tables and views. Once this meta-data is available, any SQL-literate application or programmer can access the mainframe data as if it were on a local RDBMS.

Let us see a broad outline of how the whole thing works:

1 A client application or product issues a standard JDBC/ ODBC call – only the data can be mainframe ADABAS, VSAM, or DATACOM.

2 DB2 II Classic Federation takes this request and maps it to the native data source, and, using the native data access, retrieves the appropriate data after performing the security checks. Note that the SQL requests are transformed dynamically to optimized native calls.

3 The DB2 II Classic Federation automatically handles translation and re-formatting of data from the source format to relational rows and columns, based on the meta-data.

DB2 II Classic Federation is seamlessly integrated with the WebSphere family of products. For example, the WebSphere Business Integrator workflow can now include mainframe ADABAS data through the DB2 II Classic Federation JDBC client. This actually removes a lot of hassles when designing a workflow and the typical additional overheads in terms of replicating the mainframe data, including an additional step for getting the data from a mainframe source, can be dispensed with.

DB2 II Classic Federation for z/OS allows you to see the mainframe legacy data, in whatever format, as relational data that is available right where you are – with little effort and no knowledge about the source data type.

If your organization's key business is on legacy platforms, and also uses non-relational data sources, and you are currently looking for ways to seamlessly integrate with your new suite of applications, DB2 II Classic Federation for z/OS could be the solution.

One way to look at it is that DB2 II Classic Federation for z/OS is yet another way of integrating your mainframe. But this one is simple and elegant and is expected to have a profound impact on how real-time the data in your Web site is, how your data warehouse gets populated with legacy input, and how seamlessly the integration with the mainframe takes place.

Sasirekha Cota
Tata Consultancy Services (India)

Storage class performance reporter

As new technologies emerge and business requirements change, your storage system environment continually evolves. As it does, one of your primary challenges as a storage administrator is to effectively and proactively manage these changes. Whether you have data that is managed by DFSMS or non-SMS data, you need detailed information to analyse all the elements of a data management change. We all know that the days of long batch windows, extra processor cycles, and under-utilized storage devices are over. Thus, managing a z/OS Sysplex installation in a dynamic business environment presents many new challenges, most of which centre around reducing data processing costs while increasing the efficiency of data storage and management.

In its *MVS Performance Notebook*, IBM stated that, 'over 75% of the problems reported to the IBM Washington System Center can be traced to some kind of I/O contention. Channel loading, control unit or device contention, dataset placement, paging configurations, and shared DASD are the major culprits...' A large computer installation often has hundreds of DASD devices that must be managed. Not all devices experience performance problems, and storage administrators or performance analysts have time to address only the most serious problems. Thus, the goal of investigating I/O resources is to minimize delays in satisfying I/O requests. A high level of contention within your I/O subsystem can be a concern for responsiveness and throughput. The issue is whether application I/Os are being delayed. The key to finding and fixing I/O-related performance problems is I/O response time (that is, the length of time it takes to complete an I/O operation). I/O response time can have a dramatic effect on performance, particularly with on-line and interactive subsystems, such as TSO, IMS, DB2, and CICS. I/O response time is also the most critical element in batch throughput.

STORAGE CLASS BASICS

System-managed storage enables you to improve DASD I/O performance across the installation and at the same time reduce the need for manual tuning by defining performance goals for each class of data. Prior to DFSMS, critical and important datasets that required improved performance were allocated to specific volumes manually. Datasets that required low response times were placed on low activity volumes, where cacheing was available. It is DFSMSdfp that now provides performance management by selectively managing the use of cache controller resources and selection devices that can meet performance requirements at allocation. Dynamic Cache Management Extended (DCME) dynamically determines the use of cache in a storage subsystem for each dataset when it is accessed, and it allows the separation of performance and service levels of datasets by using the

storage class. As we all know, the cache is the area within the DASD controller where data for read and write access is stored. This data would otherwise be accessed directly from a DASD volume. Reading data from a cache control unit significantly reduces the I/O time over data access from a DASD volume. Data found in cache is called a read hit. Hit I/O operations do not disconnect from the channel, so there are no RPS delays and the entire mechanical motion is eliminated, because there is no seek or latency. When the data is not found in cache, a read miss occurs. The optimum situation is to have a read hit with every read operation. This yields a 100% hit ratio. The hit ratio is the number of read hits compared with the total number of read operations. The higher the ratio, the better the cache device is being used. A high hit ratio indicates that when data is accessed, it was found in cache, which eliminates the mechanical delays (for example, RPS, seek, and latency delays) associated with DASD access.

A storage class is a list of storage objectives and requirements. Each storage class represents a list of services that are available to datasets. A storage class does not represent any physical storage, but rather provides the criteria that DFSMS uses in determining an appropriate location to place a dataset or object. DFSMS uses storage classes to separate dataset performance objectives and availability from physical storage. In other words, a storage class construct details the intended performance characteristics required for a dataset assigned to a given class. The response times set for each storage class are target response times for the disk controller to achieve when processing an I/O request. It decides whether the volume should be chosen by the user or by DFSMS. The assignment of a storage class does not guarantee its performance objective, but DFSMS selects a volume that offers performance as close as possible. Only DFSMS-managed datasets use storage classes. Changes in a storage class apply to the datasets that are already using that class.

From a performance point of view, storage classes are used

to determine the following:

- What response time is required for the data (whether the data should be cached: always, sometimes, or never).
- Whether the data has particular performance requirements that can be satisfied only by data striping or compression.
- Whether access to the data is biased towards read or write, or towards sequential or direct processing.

Storage class performance objectives are defined by specifying the performance objectives values in the corresponding fields of the Storage Class Define panel of ISMF (Interactive Storage Management Facility). Performance objectives are attributes that determine how quickly the system responds to I/O requests for datasets in this storage class: you can request millisecond response (MSR) times and indicate the bias of both direct and sequential access datasets. If you leave all MSR and bias fields blank (direct and sequential), DFSMS ignores device performance during volume selection. There are five performance objective fields that can be used to describe which storage classes have a fast response time and which ones have a slow response time:

- Direct millisecond response – the value in this field shows the direct access response time required for datasets in this storage class. The value is the number of milliseconds required to read or write a 4-kilobyte block of data.
- Direct bias – the value in this field shows the direct access bias for datasets in this storage class. The direct access bias indicates whether the majority of input/output scheduled for the datasets in this storage class is read, write, or unknown. Transaction logs usually have a write bias. A rarely updated dataset, like a production PROCLIB, would have a read bias. If no bias has been defined to the storage class, the field will be blank.
- Sequential millisecond response – the value in this field shows the sequential access response time required for

datasets in this storage class. The value is the number of milliseconds required to read a 4-kilobyte block of data.

- Sequential bias – the value in this field shows the sequential access bias for datasets in this storage class. The sequential access bias shows whether the majority of input/output scheduled for datasets in this storage class is read, write, or unknown. Transaction logs usually have a write bias. Rarely updated datasets, like a production PROCLIB, would have a read bias.
- Initial access response seconds – the value in this field shows the time required (in seconds) to locate, mount, and prepare a piece of media for data transfer.

The MSR serves basically two purposes in DFSMS. First, it is used as the performance objective for selecting candidate volumes for new dataset placement. During new dataset allocation, DFSMS looks for a volume that meets or closely matches this objective. If no volume satisfies the objective, then DFSMS tries to find a volume that comes closest to matching it. If more than one MSR is explicitly or implicitly specified, the storage class and associated device MSRs are averaged and compared. Second, if the data is placed on a volume attached through an IBM 3990 storage controller with cache, and cache is enabled for that volume, the MSR is used to determine whether cacheing is mandatory, optional, or should be inhibited for the dataset. You can request DFSMS to ignore various device performances during volume selection by leaving all MSR and bias fields blank. This lets you spread data evenly across non-cached and cached active devices. A storage administrator or performance analyst should take a note of the fact that a given DASD can have different performance capabilities for direct access (random access, for example) and for sequential access applications. Its performance capabilities depend on whether you are reading data or writing data. Please note to that each device type and model has a predetermined MSR capability for each condition. Additionally, if the device is attached to a cache-capable

control unit, the response capabilities are improved when cacheing is active. If a device is cache capable, it must also have cacheing active at the time of allocation in order to be represented by the cacheing MSR values.

ANALYSING STORAGE CLASS STATISTICS

After defining storage class performance objectives, one should analyse their impact on response times for input and output requests for datasets assigned to a particular storage class. It should help you analyse SMS storage class policy usage. For example, with the information gained by using the storage class reporter EXEC provided below, you can get the answer to questions like these: what is the average millisecond response time (MSR) for each storage class policy? Are there inconsistencies between the attribute settings for each storage class? What is the level of I/O intensity for each storage class? In order to find out what DFSMS was doing with our storage class performance objectives, a simple report writer was written. This report writer uses SMF records produced by DFSMS. When enabled by SMFPRM*xx* TYPE parameter, SMF creates record type 42 subtype 5, which provides storage class, VTOC, and VVDS I/O statistics. It should be noted that a subtype 5 record is written when the global SMF interval expires. The global SMF interval is specified via the INTVAL parameter in the SMF parmlib member. Note: record type 30 records may also be synchronized to the global interval. See 'Performing Interval Accounting' (Chapter 3) in the *z/OSMVS System Management Facilities* manual (SA22-7630) for more information on the use of record type 30.

Two reports are provided by this reporter. The first one, storage class performance report, provides the answer to the above-mentioned questions. Interesting to observe is the I/O intensity value. It indicates the degree to which a system is accessing a storage class. Keep in mind that the level of I/O intensity – high or low – is simply a measure of activity, not necessarily a problem. I use it as an indicator to help show

where real problems might exist. I would recommend looking at system-level data during the initial phase, concentrating on I/O intensity because this measurement has an effect on everything else. Looking at the information here can identify a storage class that might be causing multiple levels of problems. The formula used to develop the I/O intensity value is: I/Os per second * millisecond response time (MSR). Sometimes we refer to I/Os per second as the I/O rate. The MSR value is the total of: CONN + PEND + DISC + IOSQ.

The MSR components are:

- CONN (connect time) – the time when data is being transferred.
- PEND (pending time) – the time spent waiting for access to a system resource. PEND time can be caused by cross-system sharing.
- DISC (disconnect time) – during an I/O operation, the time the channel is disconnected from the storage control unit. High DISC times occur when data is not cached, so operation has to wait for the device. Device latency is a prime factor.
- IOSQ (I/O system queue) – the time an I/O operation is in the I/O system queue. IOSQ time indicates a control unit busy condition for a resource shared within the same subsystem.

The second report, volume statistics report, shows VTOC I/O information for selected DASD volumes, on the selected date. Different aspects of each volume's I/O VTOC response and service times are reported, such as VTOC data/index and VVDS as well as 3990 control unit cache and I/O statistics.

CODE

In order to provide a starting point from which one can begin to analyse storage class performance I have coded a sample report writer. The code is a four-part stream: the first part

(DELETE) is a clean-up step, which deletes the files to be used in later steps. In the second step (EXT425), SMF records 42 subtype 5 are extracted from the SMF dataset to a file, which can be used as a base of archived records. In the next part (SORT425), previously extracted records (selection being defined by INCLUDE's condition) are being sorted and copied to a file, which is the input to SMSPERF EXEC invoked in the SMSREXX step.

```
//DELETE    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE hlq.R425.DATA
    DELETE hlq.SMFCOPY.OUT
    SET MAXCC=Ø
/*
//EXT425   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,(6Ø,15),RLSE),
//         DCB=(your.smf.dataset)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
       DATE(yyyyddd,yyyyddd)
       START(Ø9ØØ)
       END(17ØØ)
       INDD(INDA1,OPTIONS(DUMP))
       OUTDD(OUTDA,TYPE(42(5)))
/*
//SORT425  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=hlq.SMFCOPY.OUT,DISP=SHR
//SMF42    DD DSN=hlq.R425.DATA,
//         SPACE=(CYL,(3Ø,15)),UNIT=SYSDA,DISP=(NEW,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=3276Ø)
//TOOLIN   DD *
  SORT FROM(RAWSMF) TO(SMF42) USING(SMF4)
//SMF4CNTL DD *
*
* Eliminate Header and Trailer records
* Sort by date and time
*
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,42)
  SORT FIELDS=(11,4,PD,A,7,4,BI,A)
/*
//SMSREXX  EXEC PGM=IKJEFTØ1,REGION=ØM
```

```
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.lib
//SMF      DD DISP=SHR,DSN=hlq.R425.DATA
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
prof nopref
%SMSPERF
/*
```

SMSPERF EXEC

```
/* REXX EXEC to read and format SMF 42.5 records                  */
/* Note: This particular subtype has internal offset variables,  */
/*       and the logic of this EXEC is looping on the count      */
/*    field, because these internal segments have no count field.*/

signal ON ERROR
ADDRESS TSO
 userid=SYSVAR(SYSUID)
 sms = userid||'.sms.sclass.rep'
 vto = userid||'.sms.vtoc.rep'
 x = MSG('OFF')

 IF SYSDSN(sms) = 'OK'
 THEN "DELETE "sms" PURGE"
 "ALLOC FILE(SMSSC) DA("sms")",
    " UNIT(SYSALLDA) NEW TRACKS SPACE(300,150) CATALOG",
    " REUSE RELEASE LRECL(155) RECFM(F B)"

 IF SYSDSN(vto) = 'OK'
 THEN "DELETE "vto" PURGE"
 "ALLOC FILE(VTOC) DA("vto")",
    " UNIT(SYSALLDA) NEW TRACKS SPACE(300,150) CATALOG",
    " REUSE RELEASE LRECL(120) RECFM(F B)"
/*-------------------------------------------------------------*/
/* Header for Storage Class performance report                 */
/*-------------------------------------------------------------*/
  schd.1 = left('Storage Class performance report',50)
  schd.2 = left(' ',45) left(' ----  Average time (ms)   ----',37),
           left(' ',19,' '),
           left('- 3990 Control unit cache & I/O statistics --',44)
  schd.3 = left('Date',11)   left('Time',10)   left('Class',7) ,
           left('# I/Os',9)  left('Resp ',9)   left('Conn',9)  ,
           left('Pend',9)    left('Disc',8)    left('IOSQ',4)  ,
           left('IO Rate',7) left('Intens.',7)                 ,
           left('Ca.C',4)    left('Ca.h',4)    right('Hit%',5) ,
           right('Wr.C',5)   right('Wr.h',5)   right('Seq',4)  ,
           right('RLC',4)    right('ILC',3)    left('Dev.time',9 )
  schd.4 = left('-',150,'-')
 "EXECIO * DISKW SMSSC (STEM schd.)"
```

```
/*-------------------------------------------------------------*/
/* Header for Volume Statistics report                         */
/*-------------------------------------------------------------*/
  vold.1 = left('Volume Statistics report',5Ø)
  voll.1 = left(' ',39) left(' ----  Average time (ms)   ----',32),
           left('- 399Ø Control unit cache & I/O statistics --',44)
  voll.2 = left('VOLSER',7)  left('Component',1Ø)             ,
           left('# I/Os',9)  left('Resp ',9)   left('Conn',9) ,
           left('Pend',9)    left('Disc',8)    left('IOSQ',4) ,
           left('Ca.C',4)    left('Ca.h',4)    left('Wr.C',4) ,
           left('Wr.h',4)    left('Seq',4)     left('RLC',4)  ,
           left('ILC',4)     left('Dev.time',9 )
  voll.3 = left('-',117,'-')
 "EXECIO * DISKW VTOC (STEM vold.)"

 'EXECIO *  DISKR SMF ( STEM x. FINIS'
    do i = 1 to x.Ø
/*-------------------------------------------------------------------*/
/*    Header for SMF record type 42                                  */
/*-------------------------------------------------------------------*/
   smftype  = c2d(SUBSTR(x.i,2,1))              /*  SMF record type */
   smfdate = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)   /*  Unpack SMF date */
   smftime = smf(c2d(SUBSTR(x.i,3,4)))          /*  Decode SMF time */
   sysid   = SUBSTR(x.i,11,4)              /* System identification */
   sywid   = SUBSTR(x.i,15,4)                   /*     Subsystem id */
   smfstype = c2d(SUBSTR(x.i,19,2))             /*   Record subtype */
   smf42nt  = c2d(SUBSTR(x.i,21,2))          /* No.of triplets in rec.*/
   smf42ops = c2d(SUBSTR(x.i,25,4))       /*Offset to product section */
   smf42lps = c2d(SUBSTR(x.i,29,2))       /*Length to product section */
   smf42nps = c2d(SUBSTR(x.i,31,2))       /*Number to product sections*/
   smf42sro = c2d(SUBSTR(x.i,33,4))  /*Offset to SC response time sec*/
   smf42srl = c2d(SUBSTR(x.i,37,2))  /*Length of SC response time sec*/
   smf42srn = c2d(SUBSTR(x.i,39,2))  /*Number of SC response time sec*/
   smf42vho = c2d(SUBSTR(x.i,41,4))     /*Offset to volume header sec*/
   smf42vhl = c2d(SUBSTR(x.i,45,2))     /*Length of volume header sec*/
   smf42vhn = c2d(SUBSTR(x.i,47,2))     /*Number of volume header sec*/
/*-------------------------------------------------------------------*/
/*   Storage Class Response Time Section (SMF42 subtype 5)           */
/*   I/O response and service time components are recorded in        */
/*   multiples of 128 micro-seconds. Converted to milliseconds.      */
/*-------------------------------------------------------------------*/
smssb.1 = left(' ',3)
   IF (smf42srn > Ø) Then do
    do pp = Ø to (smf42srn -1)
    sro  = (smf42sro + (pp*smf42srl))- 3
S42SCRNL = c2d(SUBSTR(x.i,sro,2))       /* Storage class name length */
S42SCRNN =     SUBSTR(x.i,sro+2,3Ø)       /* Storage class name      */
S42SCIOR = c2d(SUBSTR(x.i,sro+32,4))      /* Response time          */
    CIOR = S42SCIOR*128E-3                /* Converted to millisecond*/
S42SCIOC = c2d(SUBSTR(x.i,sro+36,4))      /* Avg I/O connect time    */
```

```
    CIOC = S42SCIOC*128E-3                /* Converted to millisecond*/
S42SCIOP = c2d(SUBSTR(x.i,sro+40,4))      /* Avg I/O pending time    */
    CIOP = S42SCIOP*128E-3                /* Converted to millisecond*/
S42SCIOD = c2d(SUBSTR(x.i,sro+44,4))      /* Avg I/O disconnect time */
    CIOD = S42SCIOD*128E-3                /* Converted to millisecond*/
S42SCIOQ = c2d(SUBSTR(x.i,sro+48,4))      /* Avg cntl unit queue time*/
    CIOQ = S42SCIOQ*128E-3                /* Converted to millisecond*/
S42SCION = c2d(SUBSTR(x.i,sro+52,4))      /* Total number of I/Os    */
iorate = format(S42SCION/18000,8,4)       /* I/O rate in seconds     */
ioint  = format(iorate*cior,8,4)          /* I/O intensity           */
/*-------------------------------------------------------------------*/
/*       3990 Control unit cache statistics                          */
/*-------------------------------------------------------------------*/
S42SCCND = c2d(SUBSTR(x.i,sro+56,4))      /* No. of cache candidates */
S42SCHIT = c2d(SUBSTR(x.i,sro+60,4))      /* No. of cache hits       */
S42SCWCN = c2d(SUBSTR(x.i,sro+64,4))      /* No. of write candidates */
S42SCWHI = c2d(SUBSTR(x.i,sro+68,4))      /* No. of write hits       */
S42SCSEQ = c2d(SUBSTR(x.i,sro+72,4))      /* No. of sequential I/Os  */
S42SCRLC = c2d(SUBSTR(x.i,sro+76,4))      /* No. of record level     */
                                          /* cache I/O operations:RLC*/
S42SCICL = c2d(SUBSTR(x.i,sro+80,4))      /* No.of inhibit cache     */
                                          /* load I/O operations :ILC*/
S42SCDAO = c2d(SUBSTR(x.i,sro+84,4)) /*Avg I/O device-active-only time*/
    CDAO = S42SCDAO*128E-3                /* Converted to millisecond*/

Select
 when S42SCCND > 0 then  rhit = (S42SCHIT/S42SCCND)*100
 otherwise               rhit= '000000'        /*Cache read hit ratio*/
END
/*-------------------------------------------------------------------*/
/*   Printed Storage Class performance variables:                    */
/*-------------------------------------------------------------------*/
smssc.1 = left(date('n',smfdate,'j'),11) left(smftime,10) ,
          left(S42SCRNN,8) ,                 /*   Storage class name */
          right(S42SCION,4) ,                /* Total number of I/Os */
          right(CIOR,9) ,                    /*   Response time (ms) */
          right(CIOC,9),            /*     Avg I/O connect time (ms) */
          right(CIOP,9),            /*     Avg I/O pending time (ms) */
          right(CIOD,9),            /*  Avg I/O disconnect time (ms) */
          right(CIOQ,6),            /* Avg cntl unit queue time (ms) */
          right(IORATE,7,4),                 /*  I/O rate in seconds */
          right(IOINT,7,4),                  /*        I/O intensity */
          right(S42SCCND,4),                 /*     Cache candidates */
          right(S42SCHIT,4),                 /*           Cache hits */
          format(RHIT,3,2),                  /*      Cache hit ratio */
          right(S42SCWCN,4),                 /*     Write candidates */
          right(S42SCWHI,6),                 /*           Write hits */
          right(S42SCSEQ,4),                 /*      Sequential I/Os */
          right(S42SCRLC,3),                 /*             RLC I/Os */
          right(S42SCICL,3),                 /*             ILC I/Os */
```

```
          right(CDAO,8)             /*Avg I/O device-active-only(ms) */
 "EXECIO * DISKW SMSSC (STEM smssc.)"
end
 "EXECIO * DISKW SMSSC (STEM smssb.)"
  end
/*-------------------------------------------------------------------*/
/*   Volume Statistics Header section.                               */
/*   Note that offsets to component sections are zero                */
/*   if there are no statistics for the component.                   */
/*-------------------------------------------------------------------*/
vtssc.1 = left(' ',1)
vtssc.2 = left('INTERVAL TOD:',13) ,
          left(date('n',smfdate,'j'),11) left(smftime,1Ø)
 "EXECIO * DISKW VTOC (STEM vtssc.)"
 "EXECIO * DISKW VTOC (STEM voll.)"
Select
 When smf42vho >Ø then call VSH  smf42vho
 otherwise nop
END
 end
/* Close & free all allocated files  */
 "EXECIO Ø DISKW SMSSC(FINIS "
 "EXECIO Ø DISKW  VTOC(FINIS "
 say
 say 'Storage Class performance report dsn ...:'sms
 say 'VTOC Statistics report dsn  ............:'vto
 say
 "free FILE(SMSSC VTOC)"
exit
/*-------------------------------------------------------------------*/
/*        Error exit routine                                         */
/*-------------------------------------------------------------------*/

ERROR: say 'The following command produced non-zero RC =' RC
       say SOURCELINE(SIGL)
       exit

SMF:
/* REXX - convert  SMF time  */
arg time
    time1    = time % 1ØØ
    hh       = time1 % 36ØØ
    hh       = RIGHT("Ø"||hh,2)
    mm       = (time1 % 6Ø) - (hh * 6Ø)
    mm       = RIGHT("Ø"||mm,2)
    ss       = time1 - (hh * 36ØØ) - (mm * 6Ø)
    ss       = RIGHT("Ø"||ss,2)
    otime  = hh||":"||mm||":"||ss           /* Compose SMF timestamp */
    return otime
```

```
VSH:
/*-------------------------------------------------------------------*/
/*   Volume Statistics section.                                      */
/*-------------------------------------------------------------------*/
parse arg smf42vho
    nxof = smf42vho- 3
     Do while nxof > Ø
S42VTNXT = c2d(SUBSTR(x.i,nxof,4))          /* Offset to next volume */
S42VTSER =     SUBSTR(x.i,nxof+4,6)         /*         Volume serial */
S42VTADR = c2d(SUBSTR(x.i,nxof+1Ø,2))       /*        Device address */
S42VTFL1 = c2d(SUBSTR(x.i,nxof+12,1))       /*                 Flags */
 Select
  when S42VTFL1 ='192' then flag = 'SMS'    /* Device is SMS managed */
  otherwise                 flag = 'ONL'    /*      Device is online */
 End
S42VTUNC = c2d(SUBSTR(x.i,nxof+2Ø,4))     /*   I/O count, unknown ds */
S42VTVDO = c2d(SUBSTR(x.i,nxof+24,4)) /*Offset to VTOC Data comp. sec*/
S42VTVDL = c2d(SUBSTR(x.i,nxof+28,2)) /*Length of VTOC Data comp. sec*/
Select
 When S42VTVDL >Ø then call VDC  S42VTVDO
 otherwise do
       VDIOR = '' ;    VDIOC = '' ;    VDIOP = '' ;
       VDIOD = '' ;    VDIOQ = '' ; S42VDION = '' ;
    S42VDCND = '' ; S42VDHIT = '' ; S42VDWCN = '' ;
    S42VDWHI = '' ; S42VDSEQ = '' ; S42VDRLC = '';
    S42VDICL = '' ; VDDAO = ''
  end
 End
S42VTVXO = c2d(SUBSTR(x.i,nxof+32,4))   /* Offset to VTOC Index sec. */
S42VTVXL = c2d(SUBSTR(x.i,nxof+36,2))   /* Length of VTOC Index sec. */
Select
 When S42VTVXL >Ø then call VDI  S42VTVXO
 otherwise do
      VXIOR = '' ;    VXIOC = '' ;    VXIOP = '' ;
      VXIOD = '' ;    VXIOQ = '' ; S42VXION = '' ;
   S42VXCND = '' ; S42VXHIT = '' ; S42VXWCN = '' ;
   S42VXWHI = '' ; S42VXSEQ = '' ; S42VXRLC = '' ;
   S42VXICL = '' ;    VXDAO = ''
  end
 End
S42VTVVO = c2d(SUBSTR(x.i,nxof+4Ø,4))         /* Offset to VVDS sec. */
S42VTVVL = c2d(SUBSTR(x.i,nxof+44,2))         /* Length of VVDS sec. */
Select
 When S42VTVVL >Ø then call VVDS  S42VTVVO
 otherwise do
  VVIOR = '' ; VVIOC = '' ; VVIOP = '' ;
  VVIOD = '' ; VVIOQ = '' ; S42VVION = '' ;
  S42VVCND = '' ; S42VVHIT = '' ; S42VVWCN = '' ;
  S42VVWHI = '' ; S42VVSEQ = '' ; S42VVRLC = '' ;
  S42VVICL = '' ; VVDAO = ''
```

```
 end
End
  nxof = S42VTNXT- 3
end
Return

VDC:
/*-------------------------------------------------------------------*/
/*   VTOC data component                                             */
/*   I/O response and service time components are recorded in        */
/*   multiples of 128 micro-seconds. Converted to milliseconds.      */
/*-------------------------------------------------------------------*/
parse arg off
  vdof =  off -3
  S42VDIOR = c2d(SUBSTR(x.i,vdof,4))    /* Response time             */
     VDIOR = S42VDIOR*128E-3            /* Response time (ms)        */
  S42VDIOC = c2d(SUBSTR(x.i,vdof+4,4))  /* Avg I/O connect time      */
     VDIOC = S42VDIOC*128E-3            /* Avg I/O connect time (ms) */
  S42VDIOP = c2d(SUBSTR(x.i,vdof+8,4))  /* Avg I/O pending time      */
     VDIOP = S42VDIOP*128E-3            /* Avg I/O pending time (ms) */
  S42VDIOD = c2d(SUBSTR(x.i,vdof+12,4)) /* Avg I/O disconnect time   */
     VDIOD = S42VDIOD*128E-3            /* Avg I/O disconnect sec.   */
  S42VDIOQ = c2d(SUBSTR(x.i,vdof+16,4)) /* Avg cntl unit queue time  */
     VDIOQ = S42VDIOQ*128E-3            /* Avg cntl unit queue (ms)  */
  S42VDION = c2d(SUBSTR(x.i,vdof+20,4)) /* Total number of I/Os      */
/*-------------------------------------------------------------------*/
/* VTOC data component / 3990 Control unit cache statistics          */
/*-------------------------------------------------------------------*/
  S42VDCND = c2d(SUBSTR(x.i,vdof+24,4)) /* No. of cache candidates   */
  S42VDHIT = c2d(SUBSTR(x.i,vdof+28,4)) /* No. of cache hits         */
  S42VDWCN = c2d(SUBSTR(x.i,vdof+32,4)) /* No. of write candidates   */
  S42VDWHI = c2d(SUBSTR(x.i,vdof+36,4)) /* No. of write hits         */
  S42VDSEQ = c2d(SUBSTR(x.i,vdof+40,4)) /* No. of sequential I/Os    */
  S42VDRLC = c2d(SUBSTR(x.i,vdof+44,4))  /* No. of RLC I/Os          */
  S42VDICL = c2d(SUBSTR(x.i,vdof+48,4))  /* No. of ICL I/Os          */
  S42VDDAO = c2d(SUBSTR(x.i,vdof+52,4)) /* Avg I/O device-active-only*/
     VDDAO = S42VDDAO*128E-3            /* Converted to milliseconds */
/*-------------------------------------------------------------------*/
/*   Printed VTOC data component variables:                          */
/*-------------------------------------------------------------------*/
 vtdrec.1= left(S42VTSER,6)  left(' Vtoc Data:',12), /*Volume Serial */
           right(S42VDION,4),                /* Total number of I/Os */
           right(VDIOR,9),                   /*   Response time (ms) */
           right(VDIOC,9),                /*Avg I/O connect time (ms) */
           right(VDIOP,9),                /*Avg I/O pending time (ms) */
           right(VDIOD,9),             /*Avg I/O disconnect time (ms) */
           right(VDIOQ,6),             /*Avg cntl unit queue time (ms)*/
           right(S42VDCND,4),                     /* Cache candidates */
           right(S42VDHIT,4),                      /*      Cache hits */
           right(S42VDWCN,4),                     /* Write candidates */
```

```
           right(S42VDWHI,4),                    /*      Write hits */
           right(S42VDSEQ,4),                    /* Sequential I/Os */
           right(S42VDRLC,4),                    /*        RLC I/Os */
           right(S42VDICL,4),                    /*        ILC I/Os */
           right(VDDAO,9)           /*Avg I/O device-active-only(ms) */
  "EXECIO * DISKW VTOC (STEM vtdrec.)"
Return

VDI:
/*-------------------------------------------------------------------*/
/*   VTOC index component                                            */
/*   I/O response and service time components are recorded in        */
/*   multiples of 128 micro-seconds. Converted to seconds.           */
/*-------------------------------------------------------------------*/
parse arg off
  viof =  off -3
S42VXIOR = c2d(SUBSTR(x.i,viof,4))            /* Response time       */
   VXIOR = S42VXIOR*128E-3               /* Converted to milliseconds*/
S42VXIOC = c2d(SUBSTR(x.i,viof+4,4))        /* Avg I/O connect time  */
   VXIOC = S42VXIOC*128E-3               /* Converted to milliseconds*/
S42VXIOP = c2d(SUBSTR(x.i,viof+8,4))         /* Avg I/O pending time */
   VXIOP = S42VXIOP*128E-3               /* Converted to milliseconds*/
S42VXIOD = c2d(SUBSTR(x.i,viof+12,4))    /* Avg I/O disconnect time  */
   VXIOD = S42VXIOD*128E-3               /* Converted to milliseconds*/
S42VXIOQ = c2d(SUBSTR(x.i,viof+16,4))    /* Avg cntl unit queue time */
   VXIOQ = S42VXIOQ*128E-3               /* Converted to milliseconds*/
S42VXION = c2d(SUBSTR(x.i,viof+20,4))       /* Total number of I/Os  */
/*-------------------------------------------------------------------*/
/*  VTOC index component / 3990 Control unit cache stat.             */
/*-------------------------------------------------------------------*/
S42VXCND = c2d(SUBSTR(x.i,viof+24,4))     /* No. of cache candidates */
S42VXHIT = c2d(SUBSTR(x.i,viof+28,4))     /* No. of cache hits       */
S42VXWCN = c2d(SUBSTR(x.i,viof+32,4))     /* No. of write candidates */
S42VXWHI = c2d(SUBSTR(x.i,viof+36,4))     /* No. of write hits       */
S42VXSEQ = c2d(SUBSTR(x.i,viof+40,4))     /* No. of sequential I/Os  */
S42VXRLC = c2d(SUBSTR(x.i,viof+44,4))     /* No. of RLC I/Os         */
S42VXICL = c2d(SUBSTR(x.i,viof+48,4))     /* No. of ICL I/Os         */
S42VXDAO = c2d(SUBSTR(x.i,viof+52,4))     /*Avg I/O device-active    */
                                                      /*-only time*/
   VXDAO = S42VXDAO*128E-3               /* Converted to milliseconds*/
/*-------------------------------------------------------------------*/
/*   Printed VTOC index component variables:                         */
/*-------------------------------------------------------------------*/
 vtirec.1= left(S42VTSER,6)  left(' Vtoc Indx:',12), /*Volume Serial */
           right(S42VXION,4),                /* Total number of I/Os */
           right(VXIOR,9),                   /*   Response time (ms) */
           right(VXIOC,9),               /* Avg I/O connect time (ms) */
           right(VXIOP,9),               /* Avg I/O pending time (ms) */
           right(VXIOD,9),            /* Avg I/O disconnect time (ms) */
           right(VXIOQ,6),           /* Avg cntl unit queue time (ms) */
```

```
           right(S42VXCND,4),                    /* Cache candidates */
           right(S42VXHIT,4),                     /*      Cache hits */
           right(S42VXWCN,4),                    /* Write candidates */
           right(S42VXWHI,4),                     /*      Write hits */
           right(S42VXSEQ,4),                     /* Sequential I/Os */
           right(S42VXRLC,4),                     /*        RLC I/Os */
           right(S42VXICL,4),                     /*        ILC I/Os */
           right(VXDAO,9)                /*Avg I/O device-active-
only(ms) */
  "EXECIO * DISKW VTOC (STEM vtirec.)"
Return

VVDS:
/*-------------------------------------------------------------------*/
/*   VVDS component                                                  */
/*   I/O response and service time components are recorded in        */
/*   multiples of 128 micro-seconds.  Converted to milliseconds.     */
/*-------------------------------------------------------------------*/
parse arg off
  vvof =  off -3
S42VVIOR = c2d(SUBSTR(x.i,vvof,4))              /* Response time   */
   VVIOR = S42VVIOR*128E-3                   /* Converted to millisec*/
S42VVIOC = c2d(SUBSTR(x.i,vvof+4,4))         /* Avg I/O connect time */
   VVIOC = S42VVIOC*128E-3                   /* Converted to millisec*/
S42VVIOP = c2d(SUBSTR(x.i,vvof+8,4))         /* Avg I/O pending time */
   VVIOP = S42VVIOP*128E-3                   /* Converted to millisec*/
S42VVIOD = c2d(SUBSTR(x.i,vvof+12,4))        /* Avg I/O disc. time   */
   VVIOD = S42VVIOD*128E-3                   /* Converted to millisec*/
S42VVIOQ = c2d(SUBSTR(x.i,vvof+16,4))        /* Avg cntl unit queue  */
   VVIOQ = S42VVIOQ*128E-3                   /* Converted to millisec*/
S42VVION = c2d(SUBSTR(x.i,vvof+20,4))        /* Total number of I/Os */
/*-------------------------------------------------------------------*/
/*  VVDS component / 3990 Control unit cache statistics              */
/*-------------------------------------------------------------------*/
S42VVCND = c2d(SUBSTR(x.i,vvof+24,4))     /* No. of cache candidates */
S42VVHIT = c2d(SUBSTR(x.i,vvof+28,4))     /* No. of cache hits       */
S42VVWCN = c2d(SUBSTR(x.i,vvof+32,4))     /* No. of write candidates */
S42VVWHI = c2d(SUBSTR(x.i,vvof+36,4))     /* No. of write hits       */
S42VVSEQ = c2d(SUBSTR(x.i,vvof+40,4))     /* No. of sequential I/Os  */
S42VVRLC = c2d(SUBSTR(x.i,vvof+44,4))     /* No. of RLC I/Os         */
S42VVICL = c2d(SUBSTR(x.i,vvof+48,4))     /* No. of ICL I/Os         */
S42VVDAO = c2d(SUBSTR(x.i,vvof+52,4))     /* Avg I/O device-active   */
                                                        /*-only time*/
   VVDAO = S42VVDAO*128E-3                /* Converted to millisecond*/
/*-------------------------------------------------------------------*/
/*   Printed VVDS component variables:                               */
/*-------------------------------------------------------------------*/
 vvdrec.1= left(S42VTSER,6)  left(' VVDS comp:',12),  /*Volume Serial*/
           right(S42VVION,4),                /* Total number of I/Os */
           right(VVIOR,9),                   /*   Response time (ms) */
```

```
            right(VVIOC,9),              /* Avg I/O connect time (ms) */
            right(VVIOP,9),              /* Avg I/O pending time (ms) */
            right(VVIOD,9),            /*Avg I/O disconnect time (ms) */
            right(VVIOQ,6),          /* Avg cntl unit queue time (ms)*/
            right(S42VVCND,4),                   /* Cache candidates */
            right(S42VVHIT,4),                    /*      Cache hits */
            right(S42VVWCN,4),                   /* Write candidates */
            right(S42VVWHI,4),                    /*      Write hits */
            right(S42VVSEQ,4),                    /* Sequential I/Os */
            right(S42VVRLC,4),                    /*        RLC I/Os */
            right(S42VVICL,4),                    /*        ILC I/Os */
            right(VVDAO,9)           /*Avg I/O device-active-only(ms) */
  "EXECIO * DISKW VTOC (STEM vvdrec.)"
Return
```

CONCLUSION

It may seem that a lot of the argument for or against the DFSMS performance options is rooted in assumptions about old technology. When DFSMS arrived in the late 1980s, cache was relatively rare and expensive and there wasn't a software mechanism to control access to it. DFSMS allowed you to 'control' that, albeit at a fairly gross level. The MSR concept itself depends on the assumption that the performance of a dataset meant the same thing all the time. Focusing system resources on a dataset to make sure it looks good all the time, so that it will not cause delay some of the time, may be considered as an example of the 'old school' of performance management. However, not all datasets that are associated with critical applications need ultimate performance at all times. That's all changed. With WLM (and SHARK supports the idea), the performance of a dataset is critical at a particular time if it is used by an important unit of work and if an I/O delay is causing the service class of that unit of work to miss its goal. On the other hand, the fact is that there are a lot of installations that don't have a SHARK subsystem or any FICON channels, so all of this wonderful new technology won't help them at all. DFSMS cache management is still important to them.

Mile Pekic
Systems Programmer (Serbia and Montenegro)

CBPDO Internet delivery

INTRODUCTION

CBPDO Internet delivery is a new way for IBM to send CBPDO product and service-only orders over the Internet.

CBPDO Internet delivery is part of a larger picture, where IBM is providing interfaces and functions that help customers to deal more directly with IBM. Customers can receive their order more quickly, simplify their installation process, and reduce tape handling.

You should order corrective and preventive service orders for Internet delivery using ShopzSeries. IBM intends to position ShopzSeries as the primary ordering and delivery method for software service on z/OS platforms.

This article describes how to use this new IBM service to order, transfer, receive, and apply a CBPDO Internet delivery (in our case study, SDK for z/OS 1.04 – FMID: HJVA140).

CBPDO INTERNET DELIVERY PACKAGE

IBM creates the CBPDO package using the SMP/E GIMZIP service routine. It creates an archive for each of the product files (SMPMCS, RELFILES, DOCLIB, README, etc).

GIMZIP also creates a package attribute file reflecting all of the archive files associated with the package.

Two methods are available to transfer the CBPDO package to your host:

- Using a direct download to host – in this case, your host must have Internet access and must run ICSF (Integrated Cryptographic Services Facility, which provides data integrity checks using hash values) in order to be able to use the RECEIVE FROMNETWORK SMP/E command.

Figure 1: ShopzSeries Web site

- Using an intermediate node workstation – in this case, you will first download the CBPDO package to your PC workstation using the Download Director Java applet provided by IBM. And then you will have to upload the package to your host SMPNTS HFS dataset using an FTP

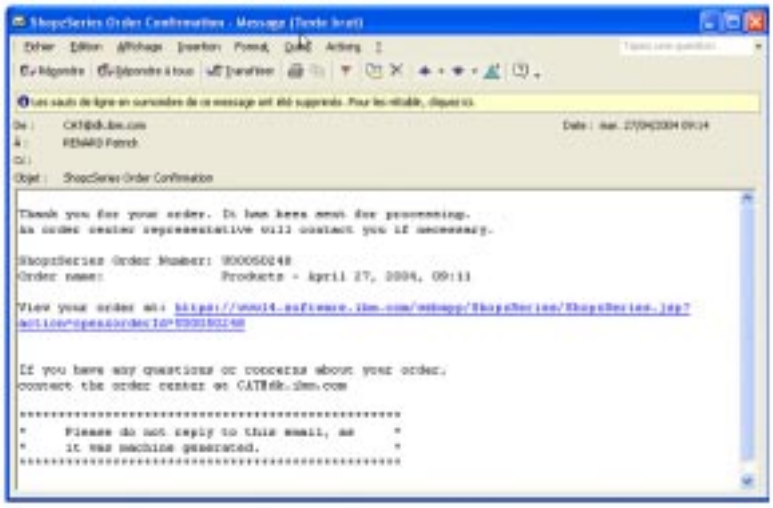

Figure 2: Order confirmation

transfer from your PC. Finally, you will use the RECEIVE FROMNTS SMP/E.

THE WHOLE CBPDO ORDER AND INSTALLATION PROCESS

Ordering the CBPDO

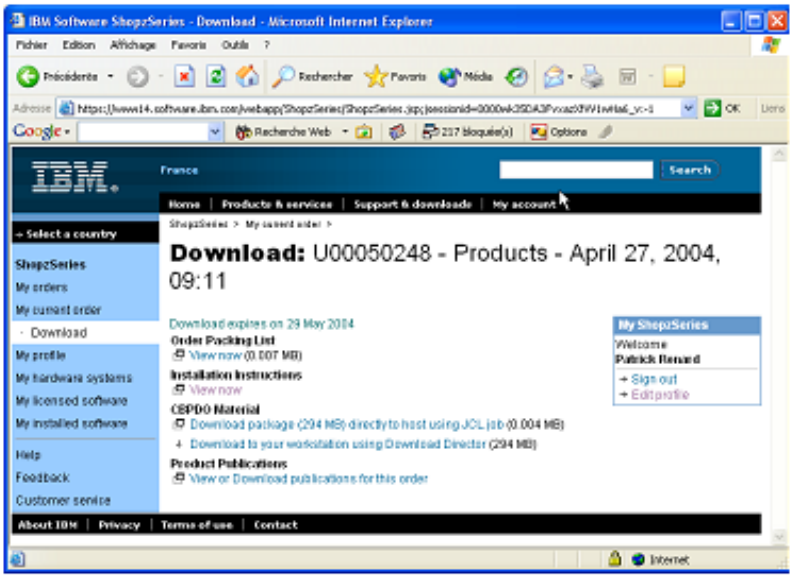

Figure 3: Order download page

First, you should connect to the ShopzSeries Web site to create your CBPDO order using the URL https://

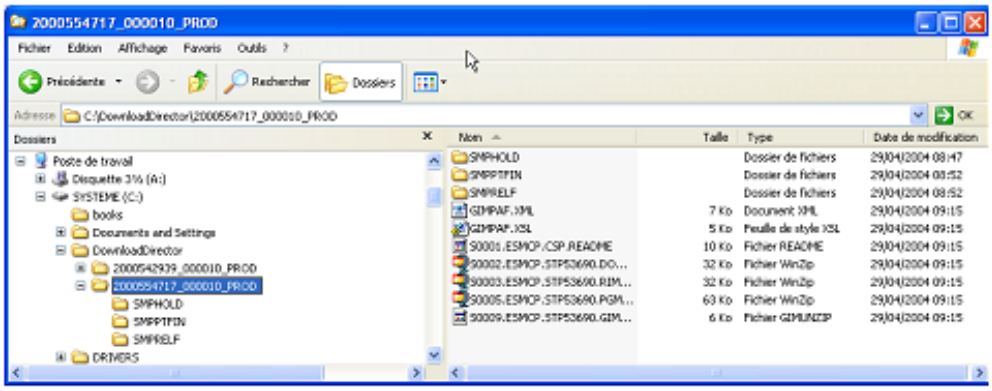

Figure 4: Disk contents

www14.software.ibm.com/webapp/ShopzSeries/ShopzSeries.jsp – see Figure 1.

When you fill in your order form, you will have to specify 'Preferred media' = Internet.

When the order is completed, you will receive a 'ShopzSeries Order Confirmation' by e-mail – see Figure 2.

Downloading the CBPDO package from the Internet using an intermediate node

A few days later, when the order is fulfilled by IBM, you will receive an e-mail to warn you that the 'Order is ready for download'.

When you click on this link, you are transferred to the order download page – see Figure 3.

In our case, we will use the Download Director method because ICSF is not running on my host.

When you click on the Download Director link, you automatically start an IBM Java applet. If you use a proxy server, you will be prompted for its configuration (IP address, port, userid, password, etc).

When the transfer is complete, you will get on your PC disk the contents of the CBPDO package – see Figure 4.

Making the package available to the host

After the package is downloaded to your workstation, it must be made available to the host.

First you should create an HFS dataset on your host to contain the SMPNTS (SMP/E Network Temporary Store) directory. This directory will be used to temporarily store the package files. Its minimum size should be three times the size of the compressed package provided on the ShopzSeries HTML page (3 * 294 = 882MB in our case).

Then you should use FTP to upload the package to the host.

The following sample FTP commands can be used to do this.

```
OPEN your_host_IP_@
uid
password

prompt
mkdir /local/appli/smpe/smpnts/STP53690
cd /local/appli/smpe/smpnts/STP53690
lcd C:\DownloadDirector\2000554717_000010_PROD
bin
mput *.XSL
mput *.XML
mput *.README
mput *.Z
mput *.GIMUNZIP
mkdir /local/appli/smpe/smpnts/STP53690/SMPHOLD
cd /local/appli/smpe/smpnts/STP53690/SMPHOLD
lcd C:\DownloadDirector\2000554717_000010_PROD\SMPHOLD
bin
mput *.Z
mkdir /local/appli/smpe/smpnts/STP53690/SMPPTFIN
cd /local/appli/smpe/smpnts/STP53690/SMPPTFIN
lcd C:\DownloadDirector\2000554717_000010_PROD\SMPPTFIN
bin
mput *.Z
mkdir /local/appli/smpe/smpnts/STP53690/SMPRELF
cd /local/appli/smpe/smpnts/STP53690/SMPRELF
lcd C:\DownloadDirector\2000554717_000010_PROD\SMPRELF
bin
mput *.Z
quit
```

When the FTP transfer is completed, you get the following data in your SMPNTS directory:

```
   File  Directory  Special_file  Commands  Help
 ------------------------------------------------------------------------
                                Directory List

 Select one or more files with / or action codes. If / is used also
 select an action from the action bar otherwise your default action
 will be used.  Select with S to use your default action.  Cursor
 select can also be used for quick navigation.  See help for details.
 EUID=0   /local/appli/smpe/smpnts/STP53690/
   Type  Perm  Changed-EST-1    -Size  Filename       Row 1 of 12
 _ Dir    750  2004-05-04 10:18   736  .
 _ Dir    755  2004-05-04 09:33   288  ..
 _ File   640  2004-05-04 09:34  7040  GIMPAF.XML
 _ File   640  2004-05-04 09:33  4800  GIMPAF.XSL
```

```
 _ Dir     750  2004-05-04 09:34    320  SMPHOLD
 _ Dir     750  2004-05-04 09:34    320  SMPPTFIN
 _ Dir     750  2004-05-04 11:35    384  SMPRELF
 _ File    640  2004-05-04 09:34   9882  S0001.ESMCP.CSP.README
 _ File    640  2004-05-04 09:34  32256  S0002.ESMCP.STP53690.DOCLIB.pax.Z
 _ File    640  2004-05-04 09:34  32256  S0003.ESMCP.STP53690.RIMLIB.pax.Z
 _ File    640  2004-05-04 09:34  64512  S0005.ESMCP.STP53690.PGMDIR.pax.Z
 _ File    640  2004-05-04 09:34   5589  S0009.ESMCP.STP53690.GIMUNZIP
```

At this point, you should edit and submit the GIMUNZIP JCL to extract the RIMLIB dataset from the package.

```
//UNZIP    EXEC PGM=GIMUNZIP,PARM='HASH=NO'
//SYSUT3   DD UNIT=SYSALLDA,SPACE=(CYL,(50,10))
//SYSUT4   DD UNIT=SYSALLDA,SPACE=(CYL,(25,5))
//SMPOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SMPDIR   DD PATH='/local/appli/smpe/smpnts/STP53690',
//            PATHDISP=KEEP
//SYSIN    DD *
<GIMUNZIP>
<ARCHDEF
name="S0002.ESMCP.STP53690.DOCLIB.pax.Z"
volume="MNT$01"
newname="SMAINT.STP53690.DOCLIB">
</ARCHDEF>
<ARCHDEF
name="S0003.ESMCP.STP53690.RIMLIB.pax.Z"
volume="MNT$01"
newname="SMAINT.STP53690.RIMLIB">
</ARCHDEF>
<ARCHDEF
name="S0005.ESMCP.STP53690.PGMDIR.pax.Z"
volume="MNT$01"
newname="SMAINT.STP53690.PGMDIR">
</ARCHDEF>
</GIMUNZIP>
/*
```

The GIMUNZIP utility creates three PDSs, which will be used to continue the CBPDO installation.

```
  Menu  Options  View  Utilities  Compilers  Help
 ------------------------------------------------------------------------
 DSLIST - Data Sets Matching SMAINT.STP53690                 Row 1 of 3
 Command ===>                                             Scroll ===> PAGE

 Command - Enter "/" to select action          Message           Volume
 ------------------------------------------------------------------------
          SMAINT.STP53690.DOCLIB                                 MNT$08
```

```
          SMAINT.STP53690.PGMDIR                                MNT$06
          SMAINT.STP53690.RIMLIB                                MNT$01
 ************************ End of Data Set list ************************
```

Receiving the package using SMP/E

After the RIMLIB dataset has been created, you can receive your order using the SMP/E RECEIVE FROMNTS command:

```
//SMPER1  EXEC PGM=GIMSMP,REGION=0M,
//             PARM='PROCESS=WAIT',
//             DYNAMNBR=120
//SMPCSI    DD  DISP=SHR,DSN=ZOSR14.GLOBAL.CSI
//SMPNTS    DD  PATHDISP=KEEP,
//             PATH='/local/appli/smpe/smpnts/'
//SMPOUT    DD  SYSOUT=*
//SMPRPT    DD  SYSOUT=*
//SMPLIST   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//*************************************************************
//* AS SHIPPED BY IBM, RCVPDO IS SET UP TO RECEIVE
//* THE FMIDS, PTFS AND HOLDDATA
//*************************************************************
//SMPCNTL  DD  *
  SET      BOUNDARY (GLOBAL) .
  RECEIVE
   FROMNTS(STP53690)
   DELETEPKG
   .

/*
```

You can optionally specify the DELETEPKG parameter to delete the package after a successful RECEIVE.

Because ICSF is not active on the system, GIM23411I messages are issued to inform you that data integrity using hash values will not be used.

```
1PAGE 0001  - NOW SET TO GLOBAL ZONE          DATE 05/04/04  TIME
19:28:36  SMP/E 32.05   SMPOUT   OUTPUT

 GIM42401I    THE FOLLOWING PARAMETERS WERE SPECIFIED ON THE EXEC
STATEMENT FOR GIMSMP: 'PROCESS=WAIT'.
   SET      BOUNDARY (GLOBAL) .
 GIM20501I    SET PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS
00.
```

```
   RECEIVE
    FROMNTS(STP53690)
    DELETEPKG
    .

 GIM23411I    ICSF IS NOT AVAILABLE. DATA INTEGRITY VERIFICATION WILL
NOT BE PERFORMED ON PACKAGE STP53690.
 GIM23411I    ICSF IS NOT AVAILABLE. DATA INTEGRITY VERIFICATION WILL
NOT BE PERFORMED ON PACKAGE STP53690.

 GIM23411I    ICSF IS NOT AVAILABLE. DATA INTEGRITY VERIFICATION WILL
NOT BE PERFORMED ON PACKAGE STP53690.

 GIM23411I    ICSF IS NOT AVAILABLE. DATA INTEGRITY VERIFICATION WILL
NOT BE PERFORMED ON PACKAGE STP53690.


 GIM35201I    SMPTLIB ZOSR14.HJVA140.F1 WAS ALLOCATED AND CATALOGED ON
VOLUME DLB$12.

 GIM23411I    ICSF IS NOT AVAILABLE. DATA INTEGRITY VERIFICATION WILL
NOT BE PERFORMED ON PACKAGE STP53690.

 GIM35201I    SMPTLIB ZOSR14.HJVA140.F2 WAS ALLOCATED AND CATALOGED ON
VOLUME DLB$12.

 GIM39401I    SMPTLIB DATA SETS WERE LOADED FOR SYSMOD HJVA140.

 GIM22701I    RECEIVE PROCESSING WAS SUCCESSFUL FOR SYSMOD HJVA140.

 GIM20501I    RECEIVE PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE
WAS 00.


 GIM20502I    SMP/E PROCESSING IS COMPLETE. THE HIGHEST RETURN CODE WAS
00. SMP/E IS AT LEVEL 32.05.
```

You can now complete the installation using the standard APPLY/ACCEPT SMP/E process.

Alexandre Goupil
Systems Programmer (France)

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

INTRODUCTION

As I reported in the June 2004 issue (number 213) of *MVS Update* (see *High resource users – accumulated statistics suite based on SMF records*) we have been developing a tool/suite to help our users to select batch jobs for performance monitoring. The development of this tool/suite is an on-going task, and we have now reached the second phase.

This phase is to incorporate the generated statistics within an ISPF suite to allow user-friendly selection of job steps to monitor.

Parallel to the development of this new phase, we have also noticed a couple of items to correct/improve.

CORRECTIONS/ENHANCEMENTS

Step name empty

By pure coincidence we had an RMF record that didn't have a step name. This caused an error in the processing of the REXX routine because the number and position of words in a parameter list then changed. To avoid this problem I now check whether the stepname is empty and, when it is, overwrite it with '########' within the REXX routine BTCHSFOR:

```
SMF30STM = Left(SMF30STM,8,' ')
if SMF30STM = '        ' then
  do
    SMF30STM = '########'
  end
```

Calculation of yearly usage

Previously I calculated the frequency of job/steps by comparing the run date of the first recorded and last recorded occurrence of a job step and the number of runs in that period. This method produced a statistical imbalance because regularly-occurring jobs over a short period of time were calculated upwards for their yearly usage.

I have now introduced a new method to somewhat counteract this discrepancy.

A new parameter for the date of the SMF dataset is now used as the period end date to calculate a more realistic time period. The start date remains the first recorded date in the sample, because occurrences over 20 are discarded and new job steps require a start date from their first occurrence.

If a job step is recorded only once, it will be assumed, as in the original code, that it occurs just once a year.

Changes have been made in JCL BTCHSTAT:

```
...
//SET1     SET SMFAL10=smf.dataset.D2004063.SMFDATA
//SET2     SET SMFDATE=2004063
...
```

Changes have been made in REXX BTCHSDB2:

```
...

trace r
 parse upper arg smfdate
 jdate = substr(smfdate,3,5)
 daysmf  = date('S',jdate,'J')
 daylast = date('B',daysmf,'S')
trace o
 numeric digits 12                  /* numeric increase from 9 to 12 */

...



/* resource averages */
runtimeav = trunc(runtimetot / count)
totsrvav = trunc(totsrvtot / count)
```

```
cpusrvav = trunc(cpusrvtot / count)
srbsrvav = trunc(srbsrvtot / count)
iosrvav  = trunc(iosrvtot / count)
msosrvav = trunc(msosrvtot / count)
/* sample period in days */
dayfirst = word(dbwork.count,15)
dayfirst = date('B',dayfirst,'S')
days = (daylast - dayfirst) + 1
/* frequency of job/jobstep per year */
if count = 1 then
  yearfactor = 1
else
  do
    frequency = trunc(count/days,8)
    yearfactor = frequency * 365
  end
/* calculated yearly resource use */
runtimeyy = trunc(runtimeav * yearfactor)
totsrvyy = trunc(totsrvav * yearfactor)
cpusrvyy = trunc(cpusrvav * yearfactor)
srbsrvyy = trunc(srbsrvav * yearfactor)
iosrvyy  = trunc(iosrvav * yearfactor)
msosrvyy = trunc(msosrvav * yearfactor)

...
```

ISPF INTERFACE EXTENSION

Introduction

In the first phase (see *MVS Update* June 2004) we had a suite of REXX/JCL to extract information from our daily SMF record files, accumulate this information, exclude the irrelevant parts, analyse the remaining parts, and report on the resource usage of batch jobs.

The daily updated statistics dataset, hlq.BTCHSTAT.DBST, which was used as the input for the report-generating step, is now used as input to an ISPF user interface. So, the users can read an on-line report and then directly select jobs to be monitored in the future.

The output from this interface is a separate dataset that can be used in conjunction with automatic and/or manual techniques to provide input for performance monitoring tools.

In our shop we have Compuware's Strobe, but I believe it won't be a problem, with minor adaptation, to work with other performance monitors. The only Strobe-specific parameter that we knowingly generate is GOMIN (expected duration of the job in minutes).

Features

Our ISPF interface is crammed with simple features to make the lives of our users as trouble-free as possible.

These features include:

- On-line help.
- Concise representation of resource usage.
- Sorts on:
  - yearly estimated usage
  - average usage
  - job name.
- Find 'string' feature.
- Selection filtering – with display/update of filters in use.
- High *nnn* feature, where a bulk selection of the first *nnn* jobs can be made.
- Individual select/unselect with display of selected jobs.
- Display of jobs selected, but filtered out.
- Filters based on job name or program name (with wildcards).
- Summary of selections made, with the ability to remove unwanted selections.
- Accumulation of selections in a dataset.
- Automatic removal of duplicate selections.

Detail

The general concept was well covered in my previous article and this forms the basis for this second phase.

Principally the second phase is just an on-line representation of the reports (hardcopy) generated in the first phase with the added ability to select job steps and collect them into one dataset for further processing.

A few interesting points that arose as I developed the routine are as follows.

The timer that I use to show the loading of the table was introduced because relatively quickly over 5,000 entries were extracted from our system and the loading seamed to hang. To display the percentage, I checked the total records and divided by 10 to obtain a value for 10%. Each time 10% was loaded I used the elapsed time feature of REXX to calculate the elapsed time in seconds and then displayed this together with the accumulated percentage. There is nothing worse than looking at an empty screen when you're waiting for something to happen.

I used the GO command simply so that the selections remained visible at first so that the user could change their mind about their selections directly at the point of selection without having to re-enter the routine. This I found particularly useful when using the High command because it allowed more control over mass selection. I later coupled the GO command with the PF3 key, because I myself often forgot to use the GO command and with PF3 I exited and found I had lost the selections I wanted. To exit without making any new selections simply use the PF4 key.

The High command is a useful one, particularly at the start of selecting job steps for performance monitoring when the user is not sure what to monitor. This command allows a quick selection of the first *nnn* job steps in one command. I have restricted this at the moment to 150, but this value can be overwritten in the code. The command functions with the top

positioned records currently displayed and can be used to select the top estimated yearly total service unit usage and/or the top average total service unit usage. It can also be used when sorted on job name, but this is not very useful (I don't recommend it).

When the High or the normal Select command is used, selected entries will be displayed with an 'S' at the front, or, if they have been filtered out, with a '-'. The filter technique is useful and necessary to prevent unwanted selections being made. In our experience, unwanted selections come mainly in three categories:

1 Standard programs from IBM or other vendors, which are already highly tuned and would cause unnecessary resource usage to monitor.

2 User jobs that are run just once or are constantly changing, even though using similar job and step names.

3 Programs from third-party vendors that are not compatible with the monitoring program, often leading to abends.

To make life easier when creating and using the filters, I have combined two wild characters, the first for one position and the second for continuation, therefore allowing many filter possibilities to be defined as one filter. Each filter can be defined to compare either the job name or the program name.

The 'find' feature is also very useful when browsing the report. The report is updated each day on our site and the find function then allows quick reference to a job or program that may have been modified after monitoring, so that the users can (if they've made notes from previous days) check to see whether improvements have been made. This is also possible when browsing the daily (hardcopy) reports, and it makes this routine a little more convenient. The method behind the sort is a REXX routine, not that supplied with the ISPF services; the REXX routine is more flexible and searches every line for a match, not just a particular column.

The RANK value on the right-hand side of the display stays with each record when the records are sorted in another order. This can be used together with the SJ (sort on job name) function and the 'find' function to group similar-named jobs together and see whether the jobs for which a user is responsible have a high ranking and may be a candidate for monitoring.

Samples

Call to routine

```
  Menu  List  Mode  Functions  Utilities  Help
------------------------------------------------------------------------------
ISRTSO                          ISPF Command Shell
Enter TSO or Workstation commands below:

===> btchsdsp

7412 rows
loading table
10% in 0   seconds
20% in 2   seconds
30% in 3   seconds
40% in 5   seconds
50% in 7   seconds
60% in 9   seconds
70% in 11  seconds
80% in 14  seconds
90% in 16  seconds
100% in 19  seconds
press <ENTER>
***
```

Main Panel

```
BTCH01T ----------------Batch Job Statistics ------ Row 1 to 32 of 7,412
Command==>                                                  Scroll=> CSR
AL13745          SORTS SJ Jobname SY Yearly SA Average        15.07.2004
19:55

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 !
-!-!--------!--------!--------!--!------------!----------!--!--!--!--!--
----!
    PALLF011 Z0MWBP00 DB2INITR 3   250796891050   68711477  7  0  0 92 1
    PMIBL512 Z0MWBP00 DB2INITR 3   173591786140 1109719203 10  0  0 89 2
    PMIBL412 Z0MWBP00 DB2INITR 3   133485934787 1280002140  5  0  0 94 3
    PMIBL312 Z0MWBP00 DB2INITR 3   132698969566 1272455898  5  0  0 94 4
    PALLFA11 Z0MWBP00 DB2INITR 3    61308268696 1368938127 58  0  0 88 5
    PALBL180 Z0MWBP00 DB2INITR 3    50631496139  166459714  7  0  0 92 6
    PMIBL212 Z0MWBP00 DB2INITR 3    49823243530 1941362305  8  0  0 91 7
    PALVT002 Z0MWBP00 DB2INITR 3    48639458020 1220369989  7  0  0 11 8
```

Filter List Input

```
BTCH03T
---------------------- Batch Job Statistics FILTER List --------------
Command==> FI                                              Scroll= CSR
AL13745                                                     14.07.2004

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
```

Filter List Display

```
BTCH02T                                                 Row 1 to 22 of 22
------------------- Batch Job Statistics FILTER List -------------------
Command==>                                                 Scroll=> CSR
AL13745                                                  15.07.2004 20:01

            command  FIlter  (new Filters)
D (Delete)
  JOBNAME  PGMNAME  COMMENTS
-!--------!--------!-------------------------------------------------!
           BTCUTILB SE PGM TO CALL DSNUTILB
(AL13745)
           DFH*     STANDARD PROGRAM                    (AL13745)
           DSNUTILB STANDARD PROGRAM                    (AL13745)
           ICEGENER STANDARD PROGRAM                    (AL13745)
           IDCAMS   STANDARD PROGRAM                    (AL13745)
           IEB*     STANDARD PROGRAM                    (AL13745)
```

Selection List

```
BTCH04T                                                  Row 1 to 19 of 19
------------------- SELECTions to be processed List --------------------
Command==>                                                     Scroll=> CSR
AL13745                                                  15.07.2004 20:02

D (Delete)                  DSN =  'SYS4.STROBE.BTCHSTAT.SELECT'
-!--------!-------!--------!--------!----!---------------------!
?!JOBNAME !STEPNUM!STEPNAME!PGMNAME !GOMIN!REQUESTOR & TIME    !
-!--------!-------!--------!--------!----!---------------------!
  PALBLSTA 2       *OMVSEX  BPXPRECP 644   AL13745  150704 1809
  PALBL056 3       Z0MWBP00 DB2INITR 98    AL13745  150704 1809
  PALBL080 3       Z0MWBP00 DB2INITR 22    AL13745  150704 1809
  PALBL095 3       Z0MWBP00 DB2INITR 103   AL13745  150704 1809
  PALBL156 3       Z0MWBP00 DB2INITR 164   AL13745  150704 1809
  PALBL169 3       Z0MWBP00 DB2INITR 429   AL13745  150704 1809
  PALBL180 3       Z0MWBP00 DB2INITR 20    AL13745  150704 1809
  PALBL256 3       Z0MWBP00 DB2INITR 212   AL13745  150704 1809
  PALBL356 3       Z0MWBP00 DB2INITR 192   AL13745  150704 1809
  PALLFA11 3       Z0MWBP00 DB2INITR 447   AL13745  150704 1808
  PALLF005 3       DB2UNLD  IKJEFT1A 2     AL13745  150704 1809
```

Limitations

As mentioned earlier, our interface has been developed with Compuware's Strobe in mind. This should not pose any problems for other performance monitoring products because the source is all there from the SMF record extraction through to the end selections and, with relatively minor modifications, any other required fields can be built into the suite.

Future planning

A further extension that I commented on in the June 2004 issue is that we may write a routine to compare the run statistics of the latest runs against our accumulated averages.

If a large discrepancy (outside of set percentage limits) is recorded, we would then report this and trigger manual or automatic monitoring for such jobs.

We've not started on this phase so far, and don't expect to have anything until the fourth quarter 2004 at the earliest. Naturally, I'll keep you all posted on any developments as and when they occur.

The output from this routine is used by us in another routine where we manage all our requests to and from Strobe. To allow others to use this output I will write a simple routine to take the generated output and insert it into a batch job for submission to Strobe. This I will make available in the very near future.

SUMMARY

The original suite produces daily updated statistics for up to the latest 20 runs of a jobstep and reports the greatest resource (total service units) users:

1 Sorted on estimated yearly usage.

2 Average usage per run.

The second phase provides a comfortable user interface to be able to select jobs/steps for future performance monitoring.

The input for the suite is the IBM standard SMF records, and the selection output is produced primarily as an interface to Compuware's Strobe product.

SUPPLIED CODE

The supplied code is:

- REXX – BTCHSDSP, batch statistic display and selection routine.
- PANELS – BTCH01T, main panel; BTCH02T, filter display; BTCH03T, new filters; BTCH04T, selection summary.
- Messages – BTCH00.
- Help panels – BTCH01H, main panel help; BTCH01H1; BTCH01H2; BTCH01H3; BTCH01H4; BTCH01H5; BTCH01H6; BTCH02H, filter display help; BTCH03H, new filters help; BTCH03H1; BTCH04H, selection summary help.

BTCHSDSP

```
/* REXX *  BTCHSDSP ************************************************ */
/* Description:  Display for BTCHSTAT DB                             */
/* ----------------------------------------------------------------- */
/* Created ....: 24.02.2004      Rolf Parker                         */
/* ----------------------------------------------------------------- */
/* Updates ....: 24.02.2004 RP Creation                              */
/* ***************************************************************** */
/* initialize variables                                              */
/* ***************************************************************** */
trace o
pfkey = ''
DD = substr(date('S'),7,2)
MM = substr(date('S'),5,2)
YY = substr(date('S'),3,2)
ddmmyy = DD || MM || YY
HH = substr(time('N'),1,2)
MM = substr(time('N'),4,2)
hhmm = HH || MM
IDS = userid() ddmmyy hhmm
wildch = '_'
wildco = '*'
highlimit = 150
numeric digits 12                  /* numeric increase from 9 to 12  */
outpos = 0
stor2 = 'AA01'
mgmt2 = 'MSYST000'
vol2  = 'SYS004'
unit2 = '3390'
spc2  = '1,1'
seldd = 'BTCHSSEl'
seldsn = "'hlq.BTCHSTAT.SELECT'"
dsnname = seldsn
dd1 = 'BTCHSTAT'
ds1 = "'hlq.BTCHSTAT.DBST'"
stor3 = 'AA01'
mgmt3 = 'MSYST000'
vol3  = 'SYS004'
unit3 = '3390'
spc3  = '1,1'
dd3 = 'BTCHSFIL'
ds3 = "'hlq.BTCHSTAT.FILTER'"
/* ***************************************************************** */
/* read FILTER(r/o) and BTCHSTAT                                     */
/* ***************************************************************** */
call open_filter_read
"ALLOC FI("dd1") DA("ds1") SHR REUSE "
"EXECIO * DISKR "dd1" (STEM dbstat. FINIS"
"ISPEXEC TBOPEN btchst WRITE"
```

```
select
  when rc = 0 then
    do
      "ISPEXEC TBERASE btchst"
    end
  when rc = 8 then nop
  otherwise
    do
      errorfunc = 'TBOPEN'
      errortab  = 'btchst'
      call errormethod
    end
end
do
  "ISPEXEC TBCREATE btchst",
  "KEYS(row)",
  "NAMES(sel jobname stepname pgmname stepnum",
  "totsrvyy totsrvav procpu prosrb proio promso count runtimav",
  "sortsy sortsa )"
  if rc > 0 then
    do
      errorfunc = 'TBCREATE'
      errortab  = 'btchst'
      call errormethod
    end
  "ISPEXEC TBSORT btchst FIELDS(totsrvyy,C,D)"
  if rc > 0 then
    do
      errorfunc = 'TBSORT'
      errortab  = 'btchst'
      call errormethod
    end
end
/* ********************************************************************* */
/* load BTCHSTAT into TABLE                                              */
/* ********************************************************************* */
say dbstat.0 'rows'
say 'loading table'
load = 0
loadtot = 0
pro100 = dbstat.0
pro10  = trunc(pro100/10)
stage = 0
seconds = trunc(time('R'))
do row = 1 to dbstat.0
  load = load + 1
  if load = pro10 then
  do
    seconds = left(trunc(time('E')),3,' ')
    if seconds <> 1 then stext = 'seconds'
```

```
    else stext = 'second'
    stage = stage + 1
    loadtot = stage * 10
    loadtot = loadtot || '% in' seconds stext
    say loadtot
    load = 0
  end
  sel = ' '
  jobname   = word(dbstat.row,1)
  stepname  = word(dbstat.row,2)
  pgmname   = word(dbstat.row,3)
  stepnum   = word(dbstat.row,4)
  totsrvyy  = word(dbstat.row,5)
  sortsy    = right(word(dbstat.row,5),13,0)
  totsrvav  = word(dbstat.row,12)
  sortsa    = right(word(dbstat.row,12),10,0)
  count     = word(dbstat.row,17)
  cpusrvav  = word(dbstat.row,13)
  srbsrvav  = word(dbstat.row,14)
  iosrvav   = word(dbstat.row,15)
  msosrvav  = word(dbstat.row,16)
  runtimav  = word(dbstat.row,11)
  procpu    = trunc(((cpusrvav/totsrvav)*100))
  prosrb    = trunc(((srbsrvav/totsrvav)*100))
  proio     = trunc(((iosrvav/totsrvav)*100))
  promso    = trunc(((msosrvav/totsrvav)*100))
  "ISPEXEC TBADD btchst "
  if rc > 0 then
    do
      errorfunc = 'TBADD'
      errortab  = 'btchst'
      call errormethod
    end
end
say 'press <ENTER>'
"ISPEXEC TBTOP btchst"
"ISPEXEC TBSKIP btchst ROW(1)"
/* ************************************************************** */
/* process master selection/action HIGH/FILTER                    */
/* DISPLAY Table                                                  */
/* ************************************************************** */
do forever
  run1 = 0
  if substr(word(zcmd,1),1,2) = "SJ" then
    do
      "ISPEXEC TBSORT btchst FIELDS(jobname,C,A,stepnum,N,A)"
      "ISPEXEC TBTOP btchst "
    end
  if substr(word(zcmd,1),1,2) = "SA" then
    do
```

```
    "ISPEXEC TBSORT btchst FIELDS(sortsa,C,D)"
    "ISPEXEC TBTOP btchst "
  end
if substr(word(zcmd,1),1,2) = "SY" then
  do
    "ISPEXEC TBSORT btchst FIELDS(sortsy,C,D)"
    "ISPEXEC TBTOP btchst "
  end
if substr(word(zcmd,1),1,2) = "HI" then
  do
    run1 = 1
    high = word(zcmd,2)
    if high = '' then high = 5Ø
    if high > highlimit | datatype(high,W) <> 1 then
      do forever
        if high = '' then
          do
            say 'no value entered'
            say 'Enter new value'
            pull high
            iterate
          end
        if datatype(high,W) <> 1 then
          do
            say 'value not a whole number'
            say 'Enter new value'
            pull high
            iterate
          end
        if high <= highlimit then leave
        say 'HIGH feature restricted to max' highlimit
        say 'Enter new value'
        pull high
      end
    call high_select
    zcmd = ''
  end
if substr(word(zcmd,1),1,3) = "FIL" then
  do
    run1 = 1
    zcmd = ''
    call filter
  end
if substr(word(zcmd,1),1,1) = "F" then
  do
    run1 = 1
    call find_it
    zcmd = ''
  end
if substr(word(zcmd,1),1,2) = "RF" then
```

```
    do
      run1 = 1
      call find_it
      zcmd = ''
    end
  btchØ1trc = ''
  btchØ1tpf = ''
  "ISPEXEC TBDISPL btchst PANEL(BTCHØ1T)"
  btchØ1trc = rc
  btchØ1tpf = pfkey
  if btchØ1tpf = 'PFØ4' then zcmd = 'CANCEL'
  if btchØ1tpf = 'PFØ3' then zcmd = 'GO'
 /*if btchØ1trc >= 8 | zcmd = 'CANCEL' then */
   if zcmd = 'CANCEL' then
     do
       run1 = 2
       leave
     end
   else
     do
       topline = ztdtop
       call selaction
       "ISPEXEC TBTOP btchst"
       "ISPEXEC TBSKIP btchst NUMBER("topline")"
       if substr(word(zcmd,1),1,1) = "G" then
         do
           run1 = 1
           leave
         end
     end
end
/* ******************************************************************* */
/* Collect selected entries                                            */
/* ******************************************************************* */
select
   when run1 = 2 then
     do
       call end_routine
     end
   When run1 = 1 then
     do
       "ISPEXEC TBSORT btchst FIELDS(totsrvyy,C,D)"
       "ISPEXEC TBTOP btchst"
       i = Ø
       do forever
         sel = 'S'
         "ISPEXEC TBSCAN btchst ARGLIST( sel ) "
         if rc = Ø then
           do
             outpos  = outpos + 1
```

```
              gomin = trunc(runtimav/6Ø) + 1
              if gomin > 144Ø then gomin = 144Ø
              selout.outpos = jobname stepnum stepname pgmname gomin IDS
              "ISPEXEC TBDELETE btchst"
            end
          else leave
        end
        selout.Ø = outpos
      end
    otherwise
      do
         call end_routine
      end
end
/* ******************************************************************* */
/* Wrap up and write out selections                                    */
/* ******************************************************************* */
'ISPEXEC TBEND btchst'
call write_select
'ISPEXEC TBEND filter'
call rem_dups_disp_sel
exit rc
/* ******************************************************************* */
/* End of main routine                                                 */
/* ******************************************************************* */
/* ******************************************************************* */
/* Select/Unselect action                                              */
/ ******************************************************************* */
selaction:
  do forever
    select
      when fun = 'S'  | fun = 's'   then
      do
        fun = 'S'
        sb = Ø
        sbpgm = pgmname
        sbjob = jobname
        call select_block
        if sb then
          sel = '_'
        else
          sel = 'S'
        "ISPEXEC TBMOD btchst ORDER "
      end
      when fun = 'U' | fun = 'u' then
      do
        fun = 'S'
        sel = ' '
        "ISPEXEC TBMOD btchst ORDER "
      end
```

```
      otherwise nop
    end
    if ZTDSELS > 1 then
      "ISPEXEC TBDISPL btchst "
    else leave
  end
return
/* ******************************************************************* */
/* write selection info to dataset                                     */
/* ******************************************************************* */
write_select:
  if outpos > Ø then
    do
      rc = MSG('OFF')
      ADDRESS TSO
      select
        when SYSDSN(seldsn) = 'OK' then
          do
            "FREE DSNAME("seldsn")"
            "ALLOC DD("seldd") DSN("seldsn") SHR MOD "
          end
        when SYSDSN(seldsn) = 'DATASET NOT FOUND' then
          do
            "ALLOC DD("seldd") NEW CATALOG REUSE ",
            "DSN("seldsn") ",
            "LRECL(8Ø) BLKSIZE(3272Ø) RECFM(F,B) ",
            "DSORG(PS) ",
            "STORCLAS("stor2") MGMTCLAS("mgmt2")",
            "VOLUME("vol2") UNIT("unit2")",
            "SPACE("spc2") TRACKS"
          end
        otherwise
          do
            "FREE DSNAME("seldsn")"
            "DELETE "seldsn
            "ALLOC DD("seldd") NEW CATALOG REUSE ",
            "DSN("seldsn") ",
            "LRECL(8Ø) BLKSIZE(3272Ø) RECFM(F,B) ",
            "DSORG(PS) ",
            "STORCLAS("stor2") MGMTCLAS("mgmt2")",
            "VOLUME("vol2") UNIT("unit2")",
            "SPACE("spc2") TRACKS"
          end
      end
      "EXECIO "outpos" DISKW "seldd" (stem selout. FINIS"
      "FREE DSNAME("seldsn")"
      rc = MSG('ON')
    end
return
end_routine:
```

```
return
errormethod:
  say 'error in ' errortab 'action' errorfunc  ' return code = ' rc
  "ISPEXEC TBEND " errortab ""
exit
/* ********************************************************************* */
/* display FILTERs                                                       */
/* ********************************************************************* */
filter:
zcmd = ''
"ISPEXEC TBCLOSE filter"
"ISPEXEC TBOPEN filter WRITE"
do forever
  if substr(word(zcmd,1),1,2) = "FI" then
      do forever
        ztdsels = Ø
        call filter_insert
        if pfkey = 'PFØ3' | pfkey = 'PFØ4' | freecnt > Ø then leave
      end
  btchØ2trc = ''
  btchØ2tpf = ''
  "ISPEXEC TBSORT filter FIELDS(fjobname,C,A fpgmname,C,A)"
  "ISPEXEC TBDISPL filter PANEL(BTCHØ2T)"
  btchØ2trc = rc
  btchØ2tpf = pfkey
  if btchØ2trc >= 8 | zcmd = 'CANCEL' then leave
  topline = ztdtop
  do while ZTDSELS > Ø
    call processfun
  end
  "ISPEXEC TBTOP filter"
  "ISPEXEC TBSKIP filter NUMBER("topline")"
end
call write_filter
call open_filter_read
return rc
/* ********************************************************************* */
/* remove FILTERs                                                        */
/* ********************************************************************* */
processfun:
  select
    when fun = 'D' then
      do
        "ISPEXEC TBDELETE filter  "
        select
          when rc < 8 then
            do
              info = 'entry deleted'
              fjobname = ''
              fpgmname = ''
```

```
            end
          when rc = 8 then
            do
              info = 'entry not found and therefore not deleted'
              fjobname = ''
              fpgmname = ''
            end
          when rc = 12 then
            do
              info = 'table not open'
              fjobname = ''
              fpgmname = ''
            end
          otherwise
            do
              errorfunc = 'TBDELETE'
              errortab  = 'filter'
              call errormethod
            end
        end
      end
    otherwise nop
  end
  if ztdsels > 0 then
    "ISPEXEC TBDISPL filter"
  info = ''
return rc
/* ********************************************************************* */
/* insert FILTERs                                                        */
/* ********************************************************************* */
filter_insert:
  freecnt = 0
  filter01 = ''
  filter02 = ''
  filter03 = ''
  filter04 = ''
  filter05 = ''
  filter06 = ''
  filter07 = ''
  filter08 = ''
  filter09 = ''
  filter10 = ''
  filter11 = ''
  filter12 = ''
  filter13 = ''
  filter14 = ''
  filter15 = ''
  filter16 = ''
  filter17 = ''
  filter18 = ''
```

```
filter19 = ''
filter2Ø = ''
fcommØ1 = ''
fcommØ2 = ''
fcommØ3 = ''
fcommØ4 = ''
fcommØ5 = ''
fcommØ6 = ''
fcommØ7 = ''
fcommØ8 = ''
fcommØ9 = ''
fcomm1Ø = ''
fcomm11 = ''
fcomm12 = ''
fcomm13 = ''
fcomm14 = ''
fcomm15 = ''
```

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

Rolf Parker
Systems Programmer (Germany)

Get logrec on-line

When serious, unclear problems occur, systems programmers often need to check the contents of the logrec records.

In a crisis situation you have to run EREP, remember and specify the correct option parameters, find out useful information from a huge quantity of sysout lines, locate the right archive logrec files, etc.

Actually, it is a very slow diagnosis process. I've always missed having a simple condensed logrec sysout (like SYSLOG), which would provide a view of essential events as they occur.

This simple program provides a condensed view of logrec. It

uses the ENF facility, listens for code 36 (ENFREQ), then the system schedules a local SRB (the listener user exit). This SRB routine obtains storage at the location to which logrec records are moved, and posts an ECB to trigger the editing process in a sysprint file. Just run the SLOGREC program as an STC and check SYSPRINT.

If nothing appears, issue a slip action record command on 0C4 (or 0Cx) events (SLIP SET,C=0C4,ACTION=RECORD). It is quite interesting to see how many 'underground' exceptions really occur.

Restrictions:

1 Enf code 36 does not signal logrec record type '9x'.

2 Enf code 36 is not global; this means one STC per z/OS image.

Records edit:

- Some records show a single TAG, such as MCH (machine check), ETR, and DPSV because I have had neither time nor a real situation in which to test them.
- Some records are not processed, eg EOD (end of day) and IPL.

SYSPRINT example:

```
12/06/2004 11:49:25.82 2084-05550F Soft MSTJCL00 S00C1 IGC0101C IEAVTR2D
0001
12/06/2004 13:08:22.98 2084-05550F MIH  Missing intrp for SMS      01E0
12/06/2004 14:15:54.66 2084-05550F SR   PIDS/SCLOG     RIDS/IXGA1AUS
RIDS/IXGIN
                            SR   SR2110SCLOG     5752UA06217
12/06/2004 15:11:06.37 2084-05550F SLH  Channel Error for JES3     0710-
BA
12/06/2004 15:11:06.57 2084-05550F UCH  Unit Check device 000A84 RDSTEST
12/06/2004 16:08:16.11 2084-05550F CRW  Channel Recovery  IOSRACRW 0000
12/06/2004 17:08:16.17 2084-05550F LMI  Link Incident for IBM-009032-005
12/06/2004 18:11:37.25 2084-05550F VTAM Vtam device failed 000D4C NET
12/06/2004 19:52:13.46 2084-05550F DDR  Perm. Error swap From 000130 To
000133
```

CODE

```
SYLOGREC TITLE 'Logrec Event Log '
SYLOGREC CSECT
*=================================================================*
* MODULE-NAME: SYLOGREC                                           *
* DESCRIPTIVE-NAME: "Online condensed Logrec"                     *
* REQUIREMENTS    :  APF                                          *
* FUNCTION        :                                               *
*  Listen to ENF code 36, Get LOGREC records and print them in a  *
*    simple format .                                              *
* RESTRICTIONS    :                                               *
*  Some Events are not implemented (IPL EOD ) see TABCODE         *
*  Doesn't eliminate duplicate record (as erep does)              *
* ABEND           :                                               *
*  U1234 => Bas Return code on ENFREQ invocation                  *
*  U2345 => ECBLIST posted not no posted ECB found                *
* RUN             :                                               *
*    Start LOGREC where LOGREC is a procedure :                   *
*       //LOGREC EXEC PGM=SYLOGREC                                *
*       //SYSPRINT DD SYSOUT=*                                    *
*       //STEPLIB DD DSN=My.loadlib,DISP=SHR                      *
*    Stop :P LOGREC                                               *
*=================================================================*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         BAKR  R14,Ø                        Save
         LR    R12,R15                      using r12
         USING SYLOGREC,R12                 Addressability
         OPEN  (SYSPRINT,OUTPUT)            Open Sysprint
         GETMAIN RU,LV=WRKLEN               Getmain Working
         LR    R11,R1                       Use R11 for
         USING WORKING,R11                  adressability
         MVC   MSGØ1(24),=CL24'Online Logrec Started...'
         PUT   SYSPRINT,MSGØ1               Msg : Started
*=================================================================*
```

```
* Extract CIB                                                    *
*================================================================*
         MVC   XCVTEXT(XCVTEXTL),EXTRLIST    Get Extract List
         EXTRACT COMADDR,FIELDS=COMM,MF=(E,XCVTEXT)
         L     R8,COMADDR                    Comm Addr
         USING COM,R8                        adressability
         ICM   R7,15,COMCIBPT                Get CIB addr
         BZ    NOCIB                         NoCib
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R7)    Free the CIB
         B     CIB1Ø
NOCIB    EQU   *
         QEDIT ORIGIN=COMCIBPT,CIBCTR=5      Set modify limit to 2
CIB1Ø    EQU   *
         L     R1,COMECBPT                   Get Comm ECB addr
         ST    R1,COMECBA                    Save Comm ECB addr
*================================================================*
* BuildEcbList                                                   *
* Evenbuf Structure :                                            *
*                  X'ØØ' Current ASCB addr                       *
*                  X'Ø4' Addr ECB                                *
*                  X'Ø8'  Addr Event Area                        *
*                  .. So on with couple ECBaddr + Area Addr      *
*                  16 Max Couple ECB-Storage                     *
*================================================================*
BUILDECB EQU   *
         L     R1,PSAAOLD-PSA(Ø,Ø)           Current ASCB
         ST    R1,EVENTBUF                   Store it
         LA    R1,EVENTBUF+4                 Skip ASCB
         LA    R2,ECBS                       Ecb Table addr
         LA    R9,17                         16 Ecbs Max + 1 Comm
ECBLØ1   EQU   *
         ST    R1,Ø(R2)                      Store ECB addr
         LA    R1,4(R1)                      to Buffer
         MVC   Ø(4,R1),=F'Ø'                 No Buffer Addr
         LA    R1,4(R1)                      Next buffer
         LA    R2,4(R2)                      Next ECB
         BCT   R9,ECBLØ1                     LOOP
         MVC   Ø(4,R2),COMECBA               Store Comm ECB addr
         OC    Ø(4,R2),=XL4'8ØØØØØØØ'        This is the last
ENFØØ    EQU   *
         L     R5,=V(PSRB36)                 Load SRB addr
         LA    R6,36                         Enf Code 36 for logrec
         LA    R4,EVENTBUF                   Load Eventbuf addr
         MODESET KEY=ZERO,MODE=SUP           To mode sup for ENFREQ
         ENFREQ ACTION=LISTEN,SRBEXIT=(R5),PARM=(R4),CODE=(R6),EOM=YES,*
               MF=(E,ENFLISTS)
         LTR   R15,R15                       ko ,
         BNZ   ABEND                         Exit with abend 1234
         MODESET KEY=NZERO,MODE=PROB         Back to mode prob
ENF2Ø    EQU   *
```

```
*=============================================================*
*  Wait For SRB event routine to Post one of these ECBs
*=============================================================*
WAIT00   EQU   *
         LA    R2,ECBS                      ECB table Addr
         LA    R9,17                        Max number of ECB
         LA    R8,1                         Increment
         XR    R7,R7                        Clear R7  => index
         WAIT  ECBLIST=ECBS                 Wait on ECB list
CHKECB   EQU   *
         L     R1,0(R2)                     get msg ecb
         L     R3,0(R2)                     Only to check shutdown
         L     R1,0(R1)                     get ECB
         N     R1,=X'40000000'              check for post
         BZ    CHKECB10                     test  Next One
         CLM   R3,B'0111',COMECBA+1         ShutDown ECB ?
         BE    SHUTDOWN                     Then , Shutdown
         LA    R1,EVENTBUF+8                To first Buffer
         XR    R6,R6                        Compute Length
         M     R6,=F'8'                     From first entry
         LA    R5,0(R7,R1)                  buffer addr addr
         L     R4,0(R7,R1)                  buffer addr
         L     R1,0(R2)                     Get ECB addr
         MVC   0(4,R1),=F'0'                and clear it
         B     LOGREC00                     Process this record
CHKECB10 EQU   *
         LA    R2,4(R2)                     To Next ECB
         BXLE  R7,R8,CHKECB                 Loop until  Max
         ABEND 2345,DUMP                    no posted ecb found.exit
*=============================================================*
* Process a logrec record:                                   *
* Find record type in TABCODE with routine addr not NULL     *
* Branch Format header routine (Date time Machine type)      *
* Branch specific record type routine                        *
*=============================================================*
LOGREC00 EQU   *
         MVI XDEP,X'40'                     Clear dependant record
         MVC XDEP+1(99),XDEP                zone
         LA    R8,4(R4)                     To Logrec record
         TM    1(R8),X'10'                  Exclude records
         BO    CLEAR                        type x'1x'
         LA    R2,TABCODE                   Event Table Code Addr
         LA    R3,TABCODEN                  Number of Tab codes
FINDCOD  EQU   *
         CLC   0(1,R8),0(R2)                Code Match
         BNE   NEXTCOD                      No, NextCode
         L     R15,=A(FMTHDR)               Format Header routine
         BALR  R14,R15                      Go , format Header
         ICM   15,B'0111',1(R2)             Is there a routine for
         LTR   R15,R15                      This type  Code => no
```

```
         BZ    CLEAR                          Don't process this code
         BALR  R14,R15                         => yes dependant routine
         B     CLEAR                          Free and reset (ecb)
NEXTCOD  EQU   *
         LA    R2,8(R2)                       Next Row in TabCode
         BCT   R3,FINDCOD                     Till the end
NOCODE   EQU   *                              No code in Tab
*
CLEAR    EQU   *
         XC    Ø(4,R5),Ø(R5)                  Clear  buffer addr
         MODESET KEY=ZERO,MODE=SUP            mode sup kØ
         L     R5,Ø(4)                        Load Length and
         FREEMAIN RU,LV=(R5),A=(R4)           Freemain
         MODESET MODE=PROB,KEY=NZERO          mode prob
         B     WAITØØ                         Wait For next event
*=============================================================*
* FMTHDR : Format Header                                      *
*        => Date + Time + XXXX-NNNNNN + TAG (From tabcode)    *
*=============================================================*
FMTHDR   EQU   *
         BAKR  R14,Ø                          Save
         LA    R4,4(R4)                       to Logrec Record
         MVC   PL4(2),2Ø(R4)                  Get Serial number
         OI    PL4+3,X'ØF'                    in
         UNPK  CL6(5),PL4(3)                  XXXX-NNNNNN form
         MVC   XSERVER(4),CL6                 XXXX
         MVI   XSERVER+4,C'-'                 XXXX-
         MVC   PL4(3),17(R4)                  Get Model number
         UNPK  WRK16(8),PL4(4)                XXXX-NNNNNN
         TR    WRK16(8),TABHEX                Ensure printable
         MVC   XSERVER+5(6),WRK16+1           Move
         TM    2(R4),X'4Ø'                    TOD format used
         BNO   NOTOD                          No
         MVC   ZTOD(8),8(R4)                  Save TOD
         B     TOD                            Go format TOD
NOTOD    EQU   *
         LA    R5,8(R4)                       To Time
         MVC   WRK12(4),4(R5)                 Move Time in Dec Format
         MVC   WRK12+4,=XL4'ØØ'               Add up X'ØØ'
         MVC   WRK12+8(4),Ø(R5)               Move Date ØØYYDDF
         CONVTOD  CONVVAL=WRK12,TODVAL=ZTOD,TIMETYPE=DEC,             *
               DATETYPE=YYDDD
TOD      EQU   *
         STCKCONV STCKVAL=ZTOD,CONVVAL=ZDATE,                        *
               TIMETYPE=DEC,DATETYPE=DDMMYYYY
         MVC   TIME_MASK(16),TIME_MASK_FIX    Refresh mask
         ED    TIME_MASK(16),ZDATE            Format Date dd/mm/yyyy
         MVC   DATE_MASK(16),DATE_MASK_FIX    Refresh mask
         ED    DATE_MASK(16),ZDATE+8          Format Time hh:mm:ss:hh
         MVC   XDATE(1Ø),DATE_MASK+1          To edit zone
```

```
         MVC   XTIME(11),TIME_MASK+1          To edit Zone
         MVC   XTYPE(Ø4),4(R2)                Short type form tabcode
         PR
         CNOP  Ø,4
*=================================================================*
* MCH    : Machine Check Handler                                  *
* Logrec : On  CP, Storage, Timer failures                        *
* Process: Print a simple "MCH" Tag (Type from Tabcode)           *
*=================================================================*
MCH      EQU   *
         BAKR  R14,Ø                          Save
         PUT   SYSPRINT,XDATE                 Print Header with TAG
         PR
*=================================================================*
* SLH    : SubChannel Logout Handler Record                       *
* Logrec : On Channel detected errors                             *
* Process: Print  "Channel error for Jobname dev-chpid"           *
*=================================================================*
SLH      EQU   *
         BAKR  R14,Ø                           Save
         LA    R4,4(R4)                        To Logrec record
         MVC   XFILLER(18),=CL18'Channel Error for '
         MVC   XFILLER+18(8),24(R4)            Jobname
         UNPK  WRK16(5),12Ø(3,R4)              Device number
         TR    WRK16(5),TABHEX                 Ensure printable
         MVC   XFILLER+27(4),WRK16             Move
         UNPK  WRK16(5),135(3,R4)              Chpid number
         TR    WRK16(5),TABHEX                 Ensure printable
         MVI   XFILLER+31,C'-'                 Dev-
         MVC   XFILLER+32(2),WRK16             Dev-chpid
         PUT   SYSPRINT,XDATE                  Print
         PR
*=================================================================*
* CRW    : Channel Report Work                                    *
* Logrec : On Path recovery (Soft) or Malfunction (Hardware)      *
* Process  Print "Channel recovery CSECT device"                  *
*=================================================================*
CRW      EQU   *
         BAKR  R14,Ø                           Save
         LA    R4,4(R4)                        to logrec record
         MVC   XFILLER(18),=CL18'Channel Recovery  '
         MVC   XFILLER+18(8),24(R4)            Csect Name
         UNPK  WRK16(5),44(3,R4)               Unpack device number
         TR    WRK16(5),TABHEX                 Ensure printable
         MVC   XFILLER+27(4),WRK16             Move
         PUT   SYSPRINT,XDATE                  Print
         PR
*=================================================================*
* OBR    : Outboard Record                                        *
* Logrec : Error Threshold reached in Device stats table          *
```

```
* Process: Select Only long OBR and Type X'3Ø' => Unit Check    *
*        : Select Only long OBR and Type X'36' => Vtam Device   *
*        : Print  "UCH  Unit Check Device xxxx Jobname"         *
*        : Print  "VTAM Device failed  xxxx Jobname"            *
*===============================================================*
OBR      EQU   *
         BAKR  R14,Ø                          Save
         LA    R4,4(R4)                       To logrec record
         TM    2(R4),X'2Ø'                    Short or long OBR
         BO    OBRSHORT                       Don't process Short
         MVC   XFILLER(18),=CL18'Unit Check device '
         CLI   Ø(R4),X'36'                    Is VTAM OBR
         BNE   OBR1Ø                          So it's Unit Check
         MVC   XFILLER(18),=CL18'Vtam device failed'
OBR1Ø    EQU   *
         UNPK  WRK16(7),57(4,R4)              Unpack device number
         TR    WRK16(7),TABHEX                Ensure printable
         MVC   XFILLER+18(8),WRK16            Move
         MVC   XFILLER+25(4),24(R4)           Associated Jobname
         PUT   SYSPRINT,XDATE                 Print
         PR
OBRSHORT EQU   *
         PR
*===============================================================*
* SWDA   Record                                                 *
* Logrec : SDWA records when a Software errors occurs           *
*        : Entered when X'4Ø' Soft-Detected Soft Error          *
*        :          when X'42' Hard-Detected Soft Error         *
*        :          when X'44' Oper-Detected Error              *
*        :          when X'48' Hard-Detected Hard Error         *
* Process: Print "Jobname Abend Load Csect Asid "               *
*===============================================================*
SDWAREC  EQU   *
         BAKR  R14,Ø                      Save
         LA    R4,4(R4)                   To logrec Record
         LA    R4,24(R4)                  Skip Header
         MVC   XJOBNAME(8),Ø(R4)          Move Jobname
         LA    R4,8(R4)                   To SDWA Base
         USING SDWA,R4                    addr
         L     R6,SDWAABCC                SDWA abend Code
         N     R6,HI                      High byte off
         XR    R7,R7                      Clear R1 for SRDL
         SRDL  R6,12                      R7= user code
         LTR   R6,R6                      Is User or System
         BNZ   SYSABEND                   Branch if system
         MVI   XCODE,C'U'                 It is a User Code
         SRL   R7,2Ø                      Convert CMP code
         CVD   R7,DW                      in decimal
         L     R6,DW+4                    get last four dec
         SRL   R6,12                      keep three bytes
```

```
         ST    R6,WORD                    Save
         B     COD00
SYSABEND EQU   *
         MVI   XCODE,C'S'                 Indicate System Code
         STCM  R6,3,WORD                  Store System code
COD00    EQU   *
         UNPK  XCODE+1(5),WORD(3)         Make completion
         TR    XCODE+1(4),TABHEX          printable
NORC     EQU   *
         MVC   XLOAD(8),SDWAMODN          Load Module
         MVC   XCSECT(8),SDWACSCT         CSect Name
         UNPK  WRK16(5),SDWAASID(3)       Unpack ASID
         TR    WRK16(4),TABHEX            Ensure printable
         MVI   WRK16+4,C' '               Clear
         MVC   XASID(4),WRK16             Move Asid
         PUT   SYSPRINT,XDATE             Print
         PR
*=================================================================*
* SR Symptom Record                                               *
* Logrec : Symptom Record built by program that detected error    *
* Process: Select only primary symptom record                     *
*        : Print "100 first bytes of primary symptom record"      *
*        :    +  "100 first bytes of ADRCMPS section              *
*=================================================================*
SR       EQU   *
         BAKR  R14,0                      Save
         LA    R4,4(R4)                   To logrec Record
         LA    R4,24(R4)                  Skip Header
         USING ADSR,R4                    To ADSR
         LR    R3,R4                      Save Addr
         AH    R3,ADSRDBO                 Offset to Primary Sympt
         LH    R5,ADSRDBL                 Primary symptoms length
         C     R5,=F'100'                 A 100 bytes Maximum length
         BNH   SR00                       to move primary record
         LA    R5,100                     Force 100 bytes  if > 100
SR00     EQU  *
         BCTR  R5,0                       -1
         EX    R5,MVCSYMP                 Move
         PUT   SYSPRINT,XDATE             Print
         MVI   XSYMP,X'40'                Clear
         MVC   XSYMP+1(99),XSYMP          Clear
*
         LR    R3,R4                       Save origin
         AH    R3,ADSRCSO                  Offset to CMPS section
         LH    R1,ADSRCSL                  CMPS length
         C     R1,=F'100'                  Is > 100
         BNH   SR10                        No
         LA    R1,100                      Else , force 100 length
SR10     EQU  *
         BCTR  R1,0                        -1
```

```
         EX    R1,MVCSYMP                    Move 1ØØ first CMPS bytes
         MVC   XDATE(35),=CL35' '            Clear Header
         PUT   SYSPRINT,XDATE                Print
         PR
         CNOP  Ø,4
MVCSYMP  MVC   XSYMP(Ø),Ø(R3)
*=================================================================*
* DDR    : Dynamic Device Reconfiguration                         *
* Logrec : On Operator or system initiated swap                   *
* Process: select only Swap permanent condition                   *
*        : Print  "Perm. Error swap from xxxx to xxxx             *
*=================================================================*
DDR      EQU   *
         BAKR  R14,Ø                       Save
         LA    R4,4(R4)                    To logrec Record
         TM    3(R4),X'1Ø'                 is permanent error swap
         BNO   DDR1Ø                       no, exit
         MVC   XFILLER(18),=CL18'Perm. Error swap  '
         MVC   XFILLER+18(5),=CL5'From '
         UNPK  WRK16(7),45(4,R4)           Unpack device number
         TR    WRK16(7),TABHEX             Ensure printable
         MVC   XFILLER+24(6),WRK16         move=> xxxx
         MVC   XFILLER+31(3),=CL3'To '     xxxx to
         UNPK  WRK16(7),53(4,R4)           Unpack device number
         TR    WRK16(7),TABHEX             Ensure printable
         MVC   XFILLER+34(6),WRK16         xxxx to xxx
         PUT   SYSPRINT,XDATE              Print
DDR1Ø    EQU   *
         PR
*=================================================================*
* MIH    : Missing Interruption Handler Record                    *
* Logrec : On Missing interuption for a device                    *
* Process: Print  "Missing intrp for xxxx Jobname "               *
*=================================================================*
MIH      EQU   *
         BAKR  R14,Ø                       Save
         LA    R4,4(R4)                    To logrec Record
         MVC   XFILLER(18),=CL18'Missing intrp for '
         MVC   XFILLER+18(8),24(R4)        JOBNAME
         UNPK  WRK16(5),122(3,R4)          Unpack device number
         TR    WRK16(5),TABHEX             Ensure printable
         MVC   XFILLER+27(4),WRK16         Move
         PUT   SYSPRINT,XDATE              Print
         PR
*=================================================================*
* ETR    : ETR                                                    *
* Logrec : External Timer event                                   *
* Process: Print a simple "ETR" Tag (Type from Tabcode)           *
*=================================================================*
ETR      EQU   *
```

```
         BAKR  R14,Ø                                    Save
         PUT   SYSPRINT,XDATE
         PR
*===============================================================*
* LMI    : LInk Maintenance Information record                  *
* Logrec : On Link incident                                     *
* Process: Print "Link incident for MMM-XXXXX-NNN "             *
*===============================================================*
LMI      EQU   *
         BAKR  R14,Ø                      Save
         LA    R4,4(R4)                   To Logrec record
         MVC   XFILLER(18),=CL18'Link Incident for '
         MVC   XFILLER+18(3),41(R4)       Manufacturer
         MVI   XFILLER+21,C'-'            IBM-
         MVC   XFILLER+22(6),32(R4)       Type IBM-ØØ9Ø32
         MVI   XFILLER+28,C'-'            IBM-ØØ9Ø32-
         MVC   XFILLER+29(3),38(R4)       IBM-ØØ9Ø32-ØØ5
         PUT   SYSPRINT,XDATE             Print
         PR
*===============================================================*
* SIM    : Service information Message                          *
* Logrec : MAintenance Informations on Failure events           *
* Process: Print a simple "SiM" Tag (Type from Tabcode)         *
*===============================================================*
SIM      EQU   *
         BAKR  R14,Ø                      Save
         PUT   SYSPRINT,XDATE             Print
         PR
*===============================================================*
* DPSV   : Dynamic Pathing Services Validation                  *
* Logrec : DPSV recovery actions                                *
* Process: Print a simple "DPSV" Tag (Type from Tabcode)        *
*===============================================================*
DPSV     EQU   *
         BAKR  R14,Ø                      Save
         PUT   SYSPRINT,XDATE             Print
         PR
*===============================================================*
* Shutdown requested by operator                                *
*===============================================================*
SHUTDOWN EQU   *
         MVC   MSG1ØØ(22),=CL22'Online Logrec Ended.'
         PUT   SYSPRINT,MSG1ØØ            Msg : Ended
*===============================================================*
* Return                                                        *
*===============================================================*
RETURN   EQU   *
         XR    R15,R15                    Clear
         PR                               Return
         CNOP  Ø,4
```

```
ABEND    EQU   *
         ABEND 1234,DUMP                    User Abend
         PR                                 return
         CNOP  Ø,4
         LTORG
SYSPRINT DCB  DDNAME=SYSPRINT,MACRF=PM,DSORG=PS,LRECL=14Ø
         CNOP  Ø,4
ENFLISTS ENFREQ ACTION=LISTEN,CODE=ØØ,SRBEXIT=(R5),PARM=(R4),EOM=YES,   *
               EOT=YES,MF=L
         CNOP  Ø,4
*=================================================================*
* Constants                                                       *
*=================================================================*
TIME_MASK_FIX  DC    XL16'FØ21207A20207A20204B202020202020'
DATE_MASK_FIX  DC    XL16'FØ212061202061202020202020202020'
HI       DC    X'ØØFFFFFF'
TABHEX   DC    15XL16'ØØ'
         DC    C'Ø123456789ABCDEF'
*=================================================================*
*              Code  Rtn addr     Tag
*=================================================================*
TABCODE  DC    X'1Ø',AL3(MCH),CL4'MCH'          ** MCH message only **
         DC    X'13',AL3(MCH),CL4'MCH'          ** MCH message only **
         DC    X'23',AL3(SLH),CL4'SLH '
         DC    X'25',AL3(CRW),CL4'CRW'
         DC    X'3Ø',AL3(OBR),CL4'UCH'
         DC    X'34',AL3(OBR),CL4'TCAM'         ** TCAM message only **
         DC    X'36',AL3(OBR),CL4'VTAM'
         DC    X'3A',AL3(OBR),CL4'DPA'          ** DPA Message  only **
         DC    X'4Ø',AL3(SDWAREC),CL4'Soft'
         DC    X'42',AL3(SDWAREC),CL4'Soft'
         DC    X'44',AL3(SDWAREC),CL4'Soft'
         DC    X'48',AL3(SDWAREC),CL4'Hard'
         DC    X'4C',AL3(SR),CL4'SR'
***      DC    X'4F',AL3(SDWAREC),CL4'Lost'     ** not processed  **
***      DC    X'5Ø',AL3(IPL),CL4'IPL'          ** not processed  **
         DC    X'6Ø',AL3(DDR),CL4'DDR'
         DC    X'71',AL3(MIH),CL4'MIH'
**       DC    X'8Ø',AL3(EOD),CL4'EOD'          ** not processed  **
**       DC    X'81',AL3(EOD),CL4'EOD'          ** not processed  **
**       DC    X'84',AL3(EOD),CL4'EOD'          ** not processed  **
**       DC    X'9Ø',AL3(MDR),CL4'MDR'          ** Not implemented**
**       DC    X'91',AL3(MDR),CL4'MDR'          ** Not implemented**
         DC    X'A1',AL3(ETR),CL4'ETR'          ** ETR message Only **
         DC    X'A2',AL3(LMI),CL4'LMI'
         DC    X'A3',AL3(SIM),CL4'SIM'          ** SIM message Only **
         DC    X'C2',AL3(DPSV),CL4'DPSV'        ** DPSV mesasge only **
TABCODEN EQU   (*-TABCODE)/8
EXTRLIST EXTRACT MF=L
*=================================================================*
```

```
* Dynamic Working Storage                                              *
*=====================================================================*
WORKING  DSECT
XCVTEXT  EXTRACT MF=L                        Extract List
XCVTEXTL EQU   *-XCVTEXT                     Extract List length
COMADDR  DS    F                             Comm Area addr
COMECBA  DS    F                             Comm ECB  addr
EVENTBUF DS    64F                           Event Buffer
ECBS     DS    18F                           List ECB
*===========Header=================================================
XDATE    DS    CL10,CL1                      Date
XTIME    DS    CL11,CL1                      Time
XSERVER  DS    CL11,CL1                      Server
XTYPE    DS    CL04,CL1                      Type (TAG TABCODE)
XDEP     DS    CL100                         Dependant zone
*===========SDWA  Record============================================
         ORG   XDEP
XJOBNAME DS    CL8,CL1                       Jobname
XCODE    DS    CL5,CL1                       Completion code
XLOAD    DS    CL8,CL1                       Load module name
XCSECT   DS    CL8,CL1                       CSect name
XASID    DS    CL8,CL1                       ASID
         DS    CL63
*===========SR   Record============================================
         ORG   XDEP
XSYMP    DS    CL100,CL1                     Symptom record
*===========Others  ================================================
         ORG   XDEP
XFILLER  DS    CL100,CL1                     All other records
*
WRK16    DS    16C                           Working Zone
WRK12    DS    16C                           ..
WORD     DS    F                             ..
DW       DS    D                             ..
PL4      DS    PL4                           ..
ZDATE    DS    CL20                          ..
ZTOD     DS    CL20                          ..
CL6      DS    CL6                           ..
MSG01    DS    CL140                         ..
MSG100   DS    CL140                         ..
ENFPTR   DS    F                             ..
TIME_MASK      DC    XL16'F021207A20207A20204B202020202020'
DATE_MASK      DC    XL16'F021206120206120202020202020202020'
WRKLEN   EQU   *-WORKING
*=====================================================================*
* Extra Dsect                                                         *
*=====================================================================*
COM      DSECT
         IEZCOM ,
CIB      DSECT
```

```
         IEZCIB ,
         LTORG
         IHAPSA
         CVT DSECT=YES
         IXGENF
         IEFENFCT
         IEFENFSG
         IEFENFPM
         IHASDWA
         ADSR
         END
```

PSRB36

```
*=================================================================*
* PROGRAM    : PSRB36                                             *
* DESCRIPTIVE-NAME: "Enf listener exit routine"                   *
* FUNCTION        :                                               *
*  Define by ENFREQ in Main program (SYLOGREC) and schedule by ENF*
*  Get logrec record, allocation storage to move it in, and       *
*      post  the associated ECB (see EVENTBUF)  to trigegr the    *
*      print process of this particular record.                   *
* AT INPUT        :                                               *
* Six Words structures :  +ØØ Plist supplied by system =>IFBENF36 *
*                      :  +Ø8 PARM addr pass by ENFREQ =>Eventbuf *
* RECOVERY                                                        *
*  Enf stops scheduling this SRB at first failure                 *
*  FRR only percolate .
*=================================================================*
PSRB36 CSECT
PSRB36 AMODE 31
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         BAKR  R14,Ø
         LR    R12,R15                    Entry
```

```
         USING PSRB36,R12                  Addressability
         LR    R11,R1                      Save 6 word structure addr
         LA    R5,FRR                      FRR routine Addr
         SETFRR A,FRRAD=(R5),PARMAD=(R1Ø),WRKREGS=(R2,R3)
         LA    R2,BADSRBØØ                 Load retry routine addr
         ST    R2,Ø(R1Ø)                   Store
         L     R8,8(,R1)                   PARM addr (passed by ENFREQ)
         L     R5,Ø(,R1)                   IFBENF36 addr
         USING IFBENF36,R5                 Addressability
         SETLOCK OBTAIN,TYPE=LOCAL,MODE=UNCOND,REGS=USE  Obtain local
         USING PSA,Ø                                     lock
         L     R7,PSAAOLD                  Current ASCB for GetM Brnch
         LA    R4,Ø                        Input TCB as TCB current
         L     R6,IFBENF36_RECORD_LENGTH   Get Logrec record length
         A     R6,=F'4'                    Add 4 for length prefix
         GETMAIN RU,LV=(R6),BRANCH=YES,BNDRY=DBLWD
         LR    R6,RØ                       Length returned
         LR    R2,R1                       Save Storage add returned
         ST    R6,Ø(,R2)                   Store Len at Offset Ø
         LR    R3,R8                       PARM Addr ENFREQ
         LA    R3,4(R3)                    Skip ASCB
         LA    R8,Ø                        Zero
         LA    R9,16                       Max rows to check
TRY      EQU   *
         CS    R8,R2,4(R3)                 Free entries in List
         BE    OK                          Ok
         LA    R8,Ø                        Not try
         LA    R3,8(R3)                    Next Entrie
         BCT   R9,TRY                      Loop
         SETLOCK RELEASE,TYPE=LOCAL        Something weird happened
         B     EXIT                        Forget it
OK       EQU   *
         LA    R2,4(R2)
         LA    RØ,Ø(R2)                    Output Buffer Skip Length
         S     R6,=F'4'                    Move length = Total - 4
         LR    R1,R6                       Length
         LR    R7,R6                       length
         L     R6,Ø(,R11)                  Load Parm List
         LA    R6,IFBENF36_RECORD_START    To Record Start
         MVCL  RØ,R6                       move
         SETLOCK RELEASE,TYPE=LOCAL        Release Local lock
         POST  (R3),LINKAGE=SYSTEM         Post ECB
EXIT     XR    R15,R15
         PR                                   return to MVS
*=================================================================*
* Retry FRR routine                                               *
*=================================================================*
BADSRBØØ EQU   *
         B     BADSRBØ1
         DC    CL8'BADSRBØØ'
```

```
BADSRBØ1 EQU   *
         SETFRR D,WRKREGS=(R2,R3)          Delete FRR
         SETLOCK RELEASE,TYPE=LOCAL        Release Local Lock
         PR
         CNOP  Ø,4
         LTORG
*=================================================================*
FRR      EQU   *
         USING FRR,R5
         LR    R5,R15
         LR    R4,R14
         LR    R7,R2                        Load PARMAD== SDWAPARM
         L     R8,Ø(R2)                     Get Retry Routine Adr
         USING SDWA,R3
         LR    R3,R1
******   CLC   4(8,R8),=CL8'BADSRBØØ'       Retry addr ok ?
**       BNE   FRRPERC                      No  , so percolate
**       MVC   SDWASRØ1(4),SDWACMPF
**        SETRP WKAREA=(R3),RC=4,RETREGS=YES,RETADDR=(R8),RECORD=NO
******   BR    R4
FRRPERC  EQU   *
         SETRP WKAREA=(R3),RC=Ø,DUMP=NO     Percolate
         BR    R4
*=================================================================*
         LTORG
         IHAFRRS
         IHASDWA
         LTORG
         IHAPSA
         IXGENF
         IFBENF36
         END
```

David Harou
Systems Programmer (France)

Programming tip

Since I've been programming in IBM Assembler for over 35 years, I don't need to refer to the POO (Principles of Operation) very often. However, today this old dog learned a new trick by checking the z/Architecture POO (SA22-7832).

Dividing a binary doubleword by a fullword has always been a pain since the quotient is only a fullword. If the doubleword value uses its high word, then you can get a Decimal Divide exception if the fullword divisor is too small. Historically, many extra lines of code have been needed to handle this.

While writing a program that calculates an average time, I decided to check to see whether any of the new-fangled instructions could help avoid the extra coding. I found Divide Single. Divide Single divides a binary 64-bit dividend by a 32-bit or 64-bit divisor. Unlike the old Divide instruction, though, it produces a 64-bit quotient and remainder. Exactly what I needed!

Here's the code fragment I used:

```
TIME     DS    D
AVERAGE  DS    D
COUNT    DS    F
         ICM   R2,15,COUNT             IS COUNT ZERO ?
         BZ    SKIP                    IF SO, SKIP
         LG    R1,TIME                 GET TIME (64 BITS)
         DSGFR RØ,R2                   DIVIDE BY COUNT (64/32 BITS)
         STG   R1,AVERAGE              SAVE AVERAGE TIME (64 BITS)
SKIP     EQU   *
```

Note that even though an even-odd pair of registers must be specified for the Divide Single instruction, the contents of the even register are not used so it doesn't need to be cleared first. It holds the 64-bit remainder. Of course, this code won't work on an old 32-bit system, but if you've gone 100% 'z', you're safe in using it.

Bob Juch

Mainstar Software has released its Backup & Recovery Manager Suite. They have identified a problem with data centers often having diverse back-up processes, which can tie up resources and increase the likelihood of error. Their solution is the Backup & Recovery Manager Suite, a group of tools that sites can choose from to meet their need for a back-up utility (including ABARS, DFSMSdss, FDR, IDCAMS, DFSMShsm incremental and AUTODUMP, IEBGENER, ICEGENER, and SORT).

The Backup & Recovery Manager Suite provides ABARS Manager (formerly Backup & Recovery Manager), which enhances and extends ABARS processing and availability. The ABARS Manager feature, Incremental ABARS, provides all the extended control information and tracking needed to make incremental back-ups simple to implement and manage. All/Star tracks and manages the other back-up tools.

For further information contact:
Mainstar Software, PO 4132, Bellevue, WA 98009-4132, USA.
Tel: (425) 455 3589.
URL: http://www.mainstar.com/products/dr/brmste/index.asp.

* * *

Cybermation has announced its Web services interface for enterprise job schedulers, which allows companies to integrate Web services with z/OS and distributed platforms.

With the ESP Web services interface, customers can allow Web users access to mainframe, Unix, or Windows batch jobs as a Web service. The job scheduler can then optimize the running of those batch jobs.

The reason sites might be interested in the Web services interface is because they can use their existing code when producing new Web services – hence making them quicker and easier to produce.

For further information contact:
Cybermation, 125 Commerce Valley Drive, West, 8th Floor, Markham, ON L3T 7W4, Canada
Tel: (905) 707 4400.
URL: http://www.cybermation.com/products/jobscheduling/overview.html.

* * *

BMC Software has announced the availability of PATROL for PeopleSoft 2.0, which provides an automated and centralized end-to-end PeopleSoft management solution. It offers improved application performance management, root-cause analysis, and automatic problem resolution to help companies lower their operating costs. Focusing on the end-user experience, PATROL for PeopleSoft optimizes resource utilization and helps improve service levels, enabling continuous availability of important business processes, says the company.

For further information contact:
BMC Softwarem, 2101 City West Blvd, Houston, TX 77042, USA.
Tel: (713) 918 8800.
URL: http://www.bmc.com/products/proddocview/0,2832,19052_19429_23113_7038,00.html.

* * *

IBM has released Version 1.6 of z/OS. The new version contains enhancements for integrating workloads, including software support for the zSeries Application Assist Processor (zAAP).

For further information contact your local IBM representative.