206

MVS

update

November 2003

In this issue

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £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 1999 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.

Editor

Trevor Eddolls

Disclaimer

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

Contributions

When Xephon is given copyright, articles published in *MVS Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) 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.

NOTIFYEXTENT catalog option

INTRODUCTION

This article describes a new CATALOG function that is available with the DFSMS APAR OW54162 (DF/SMS 1F0 – UW88212/ DFSMS 1G0 – UW88213).

This new function implements the new MODIFY CATALOG,NOTIFYEXTENT(*xx*) parameter, which notifies you about catalogs using more than *xx* per cent of their maximum number of extents (123).

This enhancement is designed to help reduce system outages resulting from catalogs reaching the maximum number of extents without warning. When this occurs, any catalog operations that need to extend the catalog again will fail, and this can cause outages in online systems or major applications.

The installation may now set a threshold value as a percentage of the maximum extents, and, for any catalog that exceeds that percentage, an immediate action message is sent to the console to warn that the threshold has been exceeded.

The installation can then take preventative action as necessary to prevent an unscheduled outage. The percentage of the maximum is calculated as the allocated extents divided by 123, multiplied by 100 and rounded to the nearest per cent.

IMPLEMENTATION

The enhancement consists of four parts:

- A new form of the MODIFY CATALOG command to establish the threshold for messages:

  ```
  MODIFY CATALOG,NOTIFYEXTENT(xxx)
  ```

  where xxx is a percentage number from 0 to 99. A value of zero means that normal monitoring will be suppressed, and is the default.

This setting is retained across catalog restarts, but not IPLs. So you should add this command to your automation package to activate this functionality. It is not possible to add this command in COMMND00 because it is processed before the CATALOG address space is active.

- New message IEC361I is issued whenever either the index or data component of a catalog exceeds the current threshold:

```
IEC361I CATALOG catname HAS REACHED xxx% OF THE MAXIMUM EXTENTS
```

  This message is issued as an immediate action (descriptor code 2) message unless the percentage value exceeds 90%, and then it is issued as a critical action message (descriptor code 11). If normal monitoring has been disabled (eg the current threshold is zero), the message will still be issued when any catalog exceeds 90%.

- Message IEC348I, resulting from the MODIFY CATALOG, ALLOCATED command, has been modified to display the current percentage allocation values for all catalogs that are not deleted or closed, and that have been referenced since the last IPL:

```
F CATALOG,ALLOCATED
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC348I ALLOCATED CATALOGS 263
*CAS*****************************************************************
* FLAGS -VOLSER-USER-CATALOG NAME                                  % *
* Y-I-R- DLB$Ø2 ØØØ1 CATALOG.ZOSR14                                4 *
* Y-I-R- SYSBA1 ØØØ1 CATALOG.MCAT.BKUPØ1                           1 *
* YSI-E- DB2KØ1 ØØØ1 CATALOG.KDB2                                 22 *
* Y-I-R- SYSBA1 ØØØ1 CATALOG.BXCF                                  5 *
* Y-I-R- SYSBA1 ØØØ1 CATALOG.BTSO                                 13 *
* Y-I-R- SYSBA1 ØØØ1 CATALOG.BPRODUIT                             11 *
* Y-I-R- SYS$A1 ØØØ1 CATALOG.BACKUP                                1 *
* Y-I-E- SYSPA2 ØØØ1 CATALOG.MCAT.PRODØ1                           1 *
*****************************************************************
* Y/N-ALLOCATED TO CAS, S-SMS, V-VLF, I-ISC, C-CLOSED, D-DELETED,    *
* R-SHARED, A-ATL, E-ECS SHARED, K-LOCKED                            *
*CAS*****************************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

- Message IEC359I, resulting from the MODIFY CATALOG, REPORT command has been modified to display the current

setting of the extent monitoring threshold, or '(NONE)' if normal monitoring is disabled.

```
F CATALOG,REPORT
IEC351I CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I CATALOG REPORT OUTPUT 423
*CAS*************************************
*  CATALOG COMPONENT LEVEL   = HDZ11G0   *
*  CATALOG ADDRESS SPACE ASN = 001D      *
*  SERVICE TASK UPPER LIMIT  =  180      *
*  SERVICE TASK LOWER LIMIT  =   60      *
*  HIGHEST # SERVICE TASKS   =   38      *
*  CURRENT # SERVICE TASKS   =   38      *
*  MAXIMUM # OPEN CATALOGS   = 1,024     *
*  ALIAS TABLE AVAILABLE     = YES       *
*  ALIAS LEVELS SPECIFIED    = 1         *
*  SYS% TO SYS1 CONVERSION   = OFF       *
*  CAS MOTHER TASK           = 009A3920  *
*  CAS MODIFY TASK           = 009A3700  *
*  CAS ANALYSIS TASK         = 009A0E88  *
*  CAS ALLOCATION TASK       = 009A33D8  *
*  VOLCAT HI-LEVEL QUALIFIER = SYS1      *
*  NOTIFY EXTENT             =   10%     *
*  DELETE UCAT/VVDS WARNING  = ON        *
*  DATASET SYNTAX CHECKING  = ENABLED   *
*CAS*************************************
```

It should be understood that this support does not provide any protection or warnings for catalogs that have no secondary space allocation.

Records with duplicate fields in a file

I was asked several times by our production people for a solution to a problem they often encountered. They receive from customers files containing identical fields in several records, and what they need is a file in which those duplicate records are eliminated. The field, identified by a length and an offset, can be in any position within the record. For this elimination of duplicates to take place, the file should be sorted by field (however, this is not always the case because there can be higher-level sort fields, like date, customer number, etc).

So I developed a program that reads a file and creates an output file, eliminating contiguous records with equal field contents and writing out just the first record of each duplicate. The program needs, as parameters, the length and offset of the field to consider. The input file can be a VSAM or a fixed-length sequential. The output file is a sequential. Any sort needed for the input file should be done within the JCL, before running the program.

At the end, the program writes out a message to sysprint stating how many records were read and how many were written.

As an example, here is a piece of JCL, examining a field with length 12 and offset 154:

```
//STEP4    EXEC PGM=DUPLICAT,PARM='12,154'
//INFILE   DD DISP=SHR,DSN=input_file
//OUTFILE  DD DISP=SHR,DSN=output_file
//SYSPRINT DD SYSOUT=*
```

DUPLICAT SOURCE CODE

```
*=================================================================*
* DUPLICAT - Find duplicate fields in contiguous records of a file   *
*            Parameters: length (max 250) and offset of field.       *
*            DDnames: Infile, Outfile, Sysprint                      *
* This program reads an input file and copies it to an output file.  *
* If more than one contiguous record has identical fields, only the  *
* first record is written to the output file.                        *
```

```
* The input file should be sorted by field. It can be a sequential   *
* or a VSAM. The output file must be sequential.                     *
*====================================================================*
&PROGRAM SETC  'DUPLICAT'
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     GETPARMS
         DC    CL16' &PROGRAM 1.2'
         DC    CL8'&SYSDATE'
*====================================================================*
* Separate input parameter by comma into its components, convert     *
* them to binary form and store them in fields PARM1 and PARM2.      *
* If any of the parms do not exist, the program terminates.          *
*====================================================================*
GETPARMS DS    ØH
         LR    R2,R1               Copy parm pointer to R2.
         L     R2,Ø(Ø,R2)          Load parm address
         LH    R3,Ø(R2)            Load parm length in R3
         OPEN  (SYSPRINT,OUTPUT)   Open sysprint (for error msgs)
         LTR   R3,R3               Any parm entered?
         BZ    ERRMSGØ             No, error
*
         LR    R6,R2
         AR    R6,R3               R6: point after end of parms
         LA    R6,2(Ø,R6)          Skip 2 bytes of parmlength
         LA    R2,2(Ø,R2)
         LR    R4,R2               R4: Current char to ckeck
         LA    R11,PARM1           Area to keep parms
         XR    R9,R9               Clear length counter
*
LOOPARMS EQU   *
         CR    R4,R6               End of parms?
         BNL   CONVERT             Yes, go convert the last one
         CLI   Ø(R4),C','          Comma (separator) found?
         BE    CONVERT             Yes, go convert parm
         LA    R9,1(Ø,R9)          Inc index (char counter)
         LA    R4,1(Ø,R4)          Inc pointer (current char)
         B     LOOPARMS            And continue
*
CONVERT  EQU   *
         LTR   R9,R9               Any chars in current parm?
```

```
         BZ    CONVERT2           No, skip pack and cvb instructions
         S     R9,=F'1'           Sub one for ex
         EX    R9,EXPACK          Execute pack
         LA    R9,1(Ø,R9)         Increment again
         CVB   R7,PARMPACK        Convert to binary into R7
         ST    R7,Ø(R11)          And store it
*
CONVERT2 EQU   *
         CR    R4,R6              End of all parms?
         BNL   CHECKPRM           Yes, jump ahead
         AR    R2,R9              Add length to base pointer
         LA    R2,1(Ø,R2)         And skip comma
         XR    R9,R9              Reset length
         LR    R4,R2              R4: Current char
         LA    R11,4(Ø,R11)       Point next storearea
         B     LOOPARMS
*
CHECKPRM CLC   PARM1,=F'-1'       Parm1 specified?
         BE    ERRMSGØ            No, error
         CLC   PARM2,=F'-1'       Same for parm2
         BE    ERRMSGØ
         CLC   PARM1,=F'25Ø'      length greater than 25Ø?
         BH    ERRMSG1
         L     R7,PARM1           R7: length - 1 (for compare)
         S     R7,=F'1'
*=====================================================================*
* Check what input file we have. First open it as VSAM.               *
* If Error, assume non-VSAM file. If VSAM, test ACB for ESDS.         *
* If ESDS, modify RPL accordingly.                                    *
* Output file must be a sequential.                                   *
*=====================================================================*
OPENACB1 EQU   *                  Open ACB for VSAM input file
         OPEN  INFILEA            If error, go open sequential
         LTR   R15,R15
         BNZ   OPENDCB1
         TESTCB ACB=INFILEA,                                        X
               ATRB=ESDS          Check if VSAM ESDS
         BNE   OPENDCB2           No, go open output file
*
ESDSFIL1 EQU   *
         MODCB RPL=INFILER,                                         X
               OPTCD=ADR          Modify RPL for ESDS
         B     OPENDCB2
*
OPENDCB1 EQU   *                  Open sequential input file
         OPEN  (INFILED,INPUT)
         LTR   R15,R15
         BNZ   ERRMSG2
         MVI   FILETYP1,C'S'      Set flag sequential (nonvsam)
```

```
         LA    R2,INFILED          Address IHADCB of input file with R2
         USING IHADCB,R2
*
OPENDCB2 EQU   *                   Open output file
         OPEN  (OUTFILE,OUTPUT)
         LTR   R15,R15
         BNZ   ERRMSG3*
*=================================================================*
* Read and compare loop                                          *
*=================================================================**
READFILE EQU   *
         XR    R8,R8               Record count for input file
         XR    R9,R9               Record count for output file
         L     R3,PARM2            R3 is last position of string to
         A     R3,PARM1            compare. Lrecl cannot be smaller
*
READLOOP EQU   *
         CLI   FILETYP1,C'V'       VSAM file?
         BNE   READSEQ1            No, go to sequential
*
READVSA1 EQU   *
         GET   RPL=INFILER         Read VSAM file
         LTR   R15,R15             End of file?
         BNZ   EXITØ
         L     R4,VAREA1           Get address of data in R4.
         SHOWCB RPL=INFILER,                                      X
               AREA=LRECL1,                                       X
               LENGTH=4,                                          X
               FIELDS=RECLEN
         L     R5,LRECL1           Get record length in R5
         B     COMPARE             and jump to compare
*
READSEQ1 EQU   *
         GET   INFILED             Read sequential (locate method)
         LR    R4,R1               copy address of data to R4.
         LH    R5,DCBLRECL         Load R5 with record length.
*
COMPARE  EQU   *
         LA    R8,1(Ø,R8)          Increment input record counter
         CR    R5,R3               Record smaller than last position?
         BL    ERRMSG4             Yes, error.
         LR    R6,R4
         A     R6,PARM2            Add offset to record address
         C     R8,=F'1'            First record has no previous
         BE    COMPARE1            string to compare with, so skip
         EX    R7,EXCOMPAR         Execute compare
         BE    COMPARE2            If strings equal, do not write
*
COMPARE1 EQU    *
```

```
         PUT   OUTFILE,(R4)        Write sequential
         LA    R9,1(Ø,R9)          Increment output counter
*
COMPARE2 EQU   *
         EX    R7,EXMOVE           Move to string
         B     READLOOP            and read next
*=====================================================================*
* Send final messages, close files, and exit                          *
*=====================================================================*
EXITØ    EQU   *
         LR    RØ,R8
         BAL   R1Ø,UNPACK
         MVC   ENDNUM1,OUT1Ø
         LR    RØ,R9
         BAL   R1Ø,UNPACK
         MVC   ENDNUM2,OUT1Ø
         PUT   SYSPRINT,ENDMSG1
         PUT   SYSPRINT,ENDMSG2
*
EXIT1    EQU   *
         CLOSE INFILED
         CLOSE INFILEA
         CLOSE OUTFILE
         CLOSE SYSPRINT
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         XR    R15,R15
         BR    R14
*=====================================================================*
* Subroutines and work areas                                          *
*=====================================================================*
EXCOMPAR EQU   *
         CLC   Ø(Ø,R6),STRING
*
EXMOVE   EQU   *
         MVC   STRING,Ø(R6)
*
EXPACK   EQU   *
         PACK  PARMPACK,Ø(Ø,R2)
*
UNPACK   EQU   *
         CVD   RØ,REGDECIM
         UNPK  OUT12,REGDECIM
         BR    R1Ø
*
ERRMSGØ  EQU   *
         PUT   SYSPRINT,=CL8Ø'> Parameters missing'
         B     EXIT1
ERRMSG1  EQU   *
```

```
         PUT   SYSPRINT,=CL8Ø'> Parm1 (length) exceeds limit of 25Ø'
         B     EXIT1
ERRMSG2  EQU   *
         PUT   SYSPRINT,=CL8Ø'> Error opening input file'
         B     EXIT1
ERRMSG3  EQU   *
         PUT   SYSPRINT,=CL8Ø'> Error opening output file'
         B     EXIT1
ERRMSG4  EQU   *
         PUT   SYSPRINT,=CL8Ø'Record smaller than compare position'
         PUT   SYSPRINT,=CL8Ø'at last read record. Program terminated'
         B     EXITØ
*
ENDMSG1  DC    C'Number of records read from INFILE . . :'
ENDNUM1  DS    CL1Ø
         DC    CL4Ø' '
ENDMSG2  DC    C'Number of records written to OUTFILE . :'
ENDNUM2  DS    CL1Ø
         DC    CL4Ø' '
STRING   DS    ØC
STRING1  DS    CL25Ø
*
INFILEA  ACB   DDNAME=INFILE
INFILER  RPL   ACB=INFILEA,                                          X
               OPTCD=LOC,                                            X
               AREA=VAREA1,                                          X
               ARG=CHAVE1
*
INFILED  DCB   DSORG=PS,MACRF=(GL),                                  X
               EODAD=EXITØ,                                          X
               DDNAME=INFILE
*
OUTFILE  DCB   DSORG=PS,MACRF=(PM),                                  X
               DDNAME=OUTFILE
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                  X
               LRECL=8Ø,                                             X
               DDNAME=SYSPRINT
*
SAVEA    DS    18F
VAREA1   DS    F
CHAVE1   DS    F
LRECL1   DS    F
FILETYP1 DC    C'V'
PARMPACK DS    D
PARM1    DC    F'-1'
PARM2    DC    F'-1'
         DS    ØD
REGDECIM DS    CL9
         DS    ØF
```

```
OUT12    DS    ØCL12
OUT1Ø    DS    CL1Ø
         DS    CL2
*
         LTORG
         DCBD  DSORG=PS
         YREGS
         END
```


A peek into SMF30 data

Many performance problems in an application can be quickly solved once the bottlenecks are identified. Performance monitoring tools help us identify bottlenecks, but what if a site doesn't have those expensive tools? What if a site is planning to get rid of its performance tools to cut down costs? Sometimes it is difficult to justify the investment in performance monitoring tools because many sites using them do not use a lot of the information that they provide.

Here is a small utility program that would interpret the SMF 30 records (address space-level accounting record) and produce a report that could help identify candidates for performance tuning. This program can be customized according to your requirements – such as populating a database for performance analysis or producing customized reports. We have utilized this in the past by populating an Access database. The information in the reports is limited to analysing batch workloads.

SOFTWARE DEPENDENCIES

This program works well with the SMF30 records produced with Version 2 Release 10 of OS/390.

Compile this program after concatenating the SYS.MACLIB provided with your release of OS/390. SMF type 30 layout will be

expanded using the IFASMFR macro and the program is set to produce the report using SMF records as input.

GETTING STARTED

Although SMF data given as is to this program can contain any of the SMF record types, it will utilize only the type 30 records with sub-types 4 and 5. It is therefore preferable to filter out those records using IFASMFDP, and SORT these in order of JOBNAME/ STEP name so that the step level reports generated are more readable. JCL to extract the SMF 30 (subtypes 4 and 5) and subsequently SORT the SMF 30 records is given below:

```
//**** Place Job Card Here
//STEP01Ø EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DUMPIN   DD DISP=SHR,DSN=MY.SMF.DATASET
//DUMPOUT  DD DSN=&&OUTSMF,DISP=(NEW,PASS),
//         DCB=(LRECL=32760,RECFM=VBS),SPACE=(CYL,(50,20),RLSE)
//SYSIN    DD *,DCB=BLKSIZE=80
 INDD(DUMPIN,OPTIONS(DUMP))
 OUTDD(DUMPOUT,TYPE(30(4,5)))
 START(0600)
 END(2200)
/*
//STPE020     EXEC PGM=SORT
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DISP=(OLD,DELETE),DSN=&&OUTSMF
//SORTOUT  DD  DSN=MY.SORTED.SMF.DATASET,DISP=(NEW,CATLG),
//         DCB=(LRECL=32760,RECFM=VBS),SPACE=(CYL,(50,20),RLSE)
//SYSIN    DD  *
     SORT FIELDS=(247,8,CH,A,215,8,CH,A,255,2,BI,A)
/*
```

As one may want to focus on a particular set of jobs, this utility allows the user to limit the report output by using masks at the job level. The include criterion is specified on the SYSIN card and can contain the exact job name; alternatively, it can include wildcard characters such as '*' to indicate 0 or more character match, or '?' to indicate exactly one character match. Multiple criteria can be given in the SYSIN card. All jobs satisfying the criteria on the SYSIN cards will be included in the generated reports.

The JCL to run this utility program is shown below:

```
//RUN      EXEC PGM=SMF3ØPG
//STEPLIB  DD  DISP=SHR,DSN=<loadlib containing the utility program>
//SYSIN    DD  *
JOBNAME=PRD*
//SMFIN    DD  DISP=SHR,DSN=MY.SORTED.SMF.DATASET
//STEPSU   DD  SYSOUT=*
//JOBSU    DD  SYSOUT=*
//STEPTIME DD  SYSOUT=*
//JOBTIME  DD  SYSOUT=*
//STEPSTOR DD  SYSOUT=*
```

The reports produced by the sample JCL show only the jobs that start with PRD (by providing the JOBNAME=PRD* in SYSIN DD card).

REPORTS

This utility produces five reports that can be used to analyse the address space accounting information – DD cards JOBSU, STEPSU, JOBTIME, STEPTIME, and STEPSTOR.

A report specified by the DD card JOBTIME contains the following information at the job level:

- Job name and job number.
- Job start date (format YYYY-MM-DD) and start time (format HH:MM:SS).
- Job end date (format YYYY-MM-DD) and end time (format HH:MM:SS).
- Total CPU time taken by the job (format HH:MM:SS).
- SRB time taken by the job (format HH:MM:SS).
- Total EXCP count for the job.
- WLM class and WLM service name.

A report specified by the DD card JOBSU contains similar information to JOBTIME except that total CPU time and SRB time is expressed in terms of service units. EXCP count is expressed in terms of I/O service units.

A report specified by the DD card STEPTIME contains the following information:

- Job name and job number.
- Step start date (format YYYY-MM-DD) and start time (format HH:MM:SS).
- Step end date (format YYYY-MM-DD) and end time (format HH:MM:SS).
- Step name.
- Program name.
- CPU time taken by the step (format HH:MM:SS).
- SRB time taken by the step (format HH:MM:SS).
- EXCP count for the step.

Report specified by the DD card STEPSU contains similar information to STEPTIME except that CPU time and SRB time are expressed in terms of service units. Additionally I/O service units are also reported.

A report specified by STEPSTOR contains information about memory usage at the step level. The following information is contained in the report:

- Job name and job number.
- Step name.
- Program name.
- Step start date (format YYYY-MM-DD) and start time (format HH:MM:SS).
- Step end date (format YYYY-MM-DD) and end time (format HH:MM:SS).
- User storage below the 16MB line (in KB).
- User storage above the 16MB line (in KB).

- System storage below the 16MB line (in KB).
- System storage above the 16MB line (in KB).

The output format generated by this utility is easily portable to a spreadsheet or a database like Microsoft Access. Once the data is imported to an office tool like Excel or Access, various in-built functions (like ORDERING or SORTING, Data/Time functions) can be used to quickly get to valuable information like identifying the top consumers of CPU time, identifying long-running jobs, charting the trends over a period of time, or capacity planning. With good job naming conventions, the information can also be used for billing or charge-back purposes. Looking at resource usage in terms of service units may also be useful when moving jobs from one system to other. The STEPSTOR report can be used to see whether the program is using too much storage below the line. If yes, it can be re-compiled with DATA (31) option (if COBOL) to relieve storage below the line. In heavily paging environments, one may want to look at long-running programs consuming high storage to see whether there is potential to reduce their storage requirements. On the other hand, if the paging activity is negligible (or UIC stays at 255), one may want to look at exploiting the storage to reduce elapsed/CPU time for the programs showing high EXCP counts. Depending on what those programs do, it may be possible to use various buffering techniques like batch LSR or System Managed Buffering (SMB).

SMF30

```
SMF30PG  TITLE 'A SAMPLE SMF30 REPORT WRITER PROGRAM'
SMF30PG  CSECT
SMF30PG  AMODE 31
SMF30PG  RMODE ANY
         MACRO
         $STRGACQ &TYPE,&LEN=,&LOC=
         AIF   ('&TYPE' EQ 'R').L0001
         L     R0,&LEN
         AGO   .L0002
.L0001   ANOP
         LR    R0,&LEN
.L0002   ANOP
         A     R0,=F'4'
```

```
         STORAGE OBTAIN,LENGTH=(Ø),LOC=&LOC
         ST    RØ,Ø(,R1)
         LA    R1,4(,R1)
         MEND
         MACRO
         $STRGREL &TYPE,&ADDR=
         AIF   ('&TYPE' EQ 'R').LØØØ1
         L     R1,&ADDR
         AGO   .LØØØ2
.LØØØ1   ANOP
         LR    R1,&ADDR
.LØØØ2   ANOP
         S     R1,=F'4'
         L     RØ,Ø(,R1)
         STORAGE RELEASE,ADDR=(1),LENGTH=(Ø)
         MEND
*
         YREGS
         USING SMF3ØPG,R12
$CONS    DS    ØH
         B     $CONSL(R15)
         DC    C'SMF3ØPG &SYSDATE. &SYSTIME.'
         DS    ØH
$CONSL   EQU   *-$CONS
         STM   R14,R12,12(R13)          Store caller information
         LR    R12,R15                  R12 Base
         LR    R2,R1                    Save R1 for later
         $STRGACQ A,LEN==A($DYNL),LOC=31   Acq Dyn Stor
         ST    R13,4(,R1)               Save Back pointer
         ST    R1,8(,R13)               Save forward pointer
         LR    R13,R1                   My save area
         USING $DYN,R13                 Let Assembler know
         BAS   R14,OPEN#ØØØ             Open all the files.
         LTR   R15,R15                  Mandatory files opened ?
         BNZ   RETURN4                  N: bail out
         LA    R1,SYSIN                 point to SYSIN offset
         M     RØ,=A(FILESIZ)           Get the real offset
         A     R1,FILEDATA              Add start
         MVI   INCL,INCLALL             Default to include all
         CLI   FILEIND-FILEINFO(R1),X'ØØ' Was the file opened ?
         BE    SMFP#Ø2Ø                 N:
         L     R2,FILEREC-FILEINFO(,R1) Get the record address
         L     R3,FILEDCB-FILEINFO(,R1) Get the SYSIN DCB Addr
SMFP#ØØ5 DS    ØH
         LR    RØ,R2                    Move record ADDR to RØ
         GET   (3)                      Get a record
         CLC   Ø(Ø8,R2),=C'JOBNAME='    Is it a mask ?
         BNE   SMFP#ØØ5                 N:
         CLI   Ø8(R2),C' '              Is there a mask ?
         BE    SMFP#ØØ5                 N:
```

```
         XC    INCL,INCL                 Indicate mask present
         $STRGACQ A,LEN==A(MASKL),LOC=31 Acq stor for mask
         MVC   MASKNEXT-MASKDSCT(4,R1),MASKLIST Store next adr
         ST    R1,MASKLIST               Store anchor
         MVC   MASKNAME-MASKDSCT(8,R1),Ø8(R2) Store Mask
         LA    R4,Ø8(,R2)                Bump past literal
         LA    R5,Ø8(,R2)                Bump past literal
         LA    R6,8                      length of jobname
SMFP#ØØ6 DS    ØH
         CLI   Ø(R4),C' '                is this char blank ?
         BE    SMFP#ØØ7                  Y:
         LA    R4,1(,R4)                 bump past this char
         BCT   R6,SMFP#ØØ6               loop back
SMFP#ØØ7 DS    ØH
         SR    R4,R5                     subtract start address
         LTR   R4,R4                     is it zero ?
         BZ    SMFP#ØØ5                  y: back to read
         ST    R4,MASKNAML-MASKDSCT(R1)  store length
         B     SMFP#ØØ5                  read next record
SMFP#Ø1Ø DS    ØH
SMFP#Ø15 DS    ØH
SMFP#Ø2Ø DS    ØH
         LA    R1,SMFIN                  process smf file
         M     RØ,=A(FILESIZ)
         A     R1,FILEDATA
         CLI   FILEIND-FILEINFO(R1),X'ØØ' is the file open ?
         BE    RETURN4                   n: go back
         L     R2,FILEREC-FILEINFO(,R1)  get the record address
         L     R3,FILEDCB-FILEINFO(,R1)  get the dcb address
SMFP#Ø25 DS    ØH
         LR    RØ,R2                     store record address in RØ
         GET   (3)                       get a record
         MVI   STEP,ISSTEP               indicate this is a step rec
         CLI   SMF3ØRTY-SMFRCD3Ø(R2),X'1E' is this rec type 3Ø ?
         BNE   SMFP#Ø25
         CLC   SMF3ØSTP-SMFRCD3Ø(2,R2),=H'Ø4' is this a subtype 4 ?
         BL    SMFP#Ø25
         CLC   SMF3ØSTP-SMFRCD3Ø(2,R2),=H'Ø5' is this  asubtype 5 ?
         BH    SMFP#Ø25
         CLC   SMF3ØSTP-SMFRCD3Ø(2,R2),=H'Ø4'
         BE    SMFP#251
         MVI   STEP,ISJOB
SMFP#251 DS    ØH
         LA    R1,SMF3ØDTE-SMFRCD3Ø(R2)  get the end   date
         BAS   R14,CNVD#ØØØ              convert date
         MVC   ENDDATE,WORKDATE+2        save date for later
         ICM   R1,15,SMF3ØTME-SMFRCD3Ø(R2) get the end time
         BAS   R14,CNVT#ØØØ              convert time
         MVC   ENDTIME,WORKTIME+2        save time for later
         CLC   SMF3ØIOF-SMFRCD3Ø(4,R2),=F'Ø' Iden section present ?
```

```
          BE    SMFP#Ø26
          CLC   SMF3ØILN-SMFRCD3Ø(2,R2),=H'Ø' Iden section present ?
          BE    SMFP#Ø26
          CLC   SMF3ØION-SMFRCD3Ø(2,R2),=H'Ø' Iden section present ?
          BE    SMFP#Ø26
          L     R4,SMF3ØIOF-SMFRCD3Ø(R2)  map the iden section
          AR    R4,R2
          MVC   JOBNAME,SMF3ØJBN-SMF3ØID(R4) get jobname
          CLI   INCL,INCLALL              is this include all ?
          BE    SMFP#254                  n:
          L     R5,MASKLIST               point to mask list
SMFP#253  DS    ØH
          LTR   R5,R5                     is one present ?
          BZ    SMFP#Ø25                  n: get another record
          BAS   R14,CHEK#ØØØ              check the mask
          LTR   R15,R15                   check good ?
          BZ    SMFP#254                  y: process record
          L     R5,MASKNEXT-MASKDSCT(,R5) get next mask
          B     SMFP#253                  check again
SMFP#254  DS    ØH
          MVC   JOBNUM,SMF3ØJNM-SMF3ØID(R4) get the job number
          MVC   STEPNAME,SMF3ØSTM-SMF3ØID(R4) get the step name
          MVC   PROGRAM,SMF3ØPGM-SMF3ØID(R4)  get the program name
          LA    R1,SMF3ØSTD-SMF3ØID(R4)   Get start date
          BAS   R14,CNVD#ØØØ              Convert date
          MVC   STRTDTE,WORKDATE+2        Save for later
          ICM   R1,15,SMF3ØSIT-SMF3ØID(R4) Get start time
          BAS   R14,CNVT#ØØØ              Convert Time
          MVC   STRTTME,WORKTIME+2        Save time for later
SMFP#Ø26  DS    ØH
          BAS   R14,HEDR#ØØØ              Build healder info
          BAS   R14,SUDT#ØØØ              Build SU details
          BAS   R14,TMDT#ØØØ              Build Time details
          BAS   R14,STDT#ØØØ              Build Storage details
          B     SMFP#Ø25                  Get next record
SMFP#Ø3Ø  DS    ØH
SMFP#Ø35  DS    ØH
SMFP#Ø4Ø  DS    ØH
          B     RETURNØ
RETURNØ   DS    ØH
          LA    R5,Ø                      Process OK. RC=Ø
          B     RETURN
RETURN4   DS    ØH
          LA    R5,4                      Process not OK. RC=4
RETURN    DS    ØH
          BAS   R14,CLOS#ØØØ              Close all files.
          LR    R1,R13                    Save current dyn area
          L     R13,4(,R13)               Restore old dyn area
          $STRGREL R,ADDR=1               Release curr dyn area
          L     R14,12(,R13)              Restore R14
```

```
         LR    R15,R5                   R15->RC
         LM    RØ,R12,2Ø(R13)           Restore other regs
         BSM   Ø,R14                    goback
*        Check if the pattern mask matches the jobname
CHEK#ØØØ BAKR  R14,Ø
         LA    R2,MASKNAME-MASKDSCT(R5) Get address of Mask
         LA    R3,MASKNAML-MASKDSCT(R5) Point to length of mask
         LA    R4,JOBNAME               Point to current jobname
         ASAXWC    PATTERNSTR=(2),                                  -
               PATTERNSTRLEN=(3),                                   -
               STRING=(4),                                          -
               STRINGLEN==A(8),                                     -
               ZEROORMORE=Z,                                        -
               ONECHAR=O,                                           -
               RETCODE=RETCODE,                                     -
               WORKAREA=WORKAREA,                                   -
               MF=(E,MYLIST)
         L     R15,RETCODE              RC=Ø, match, else nomatch
CHEK#999 PR                             Go back to caller
*        Open all files.
OPEN#ØØØ BAKR  R14,Ø
         SLR   RØ,RØ                    Initialize RØ
         LA    R1,FILESIZ               Get the size of 1 file rec
         LA    R3,DDINFOS               Get no. of DDs
         MR    RØ,R3                    Get size required
         LR    RØ,R1
         $STRGACQ R,LEN=Ø,LOC=31        Acq Dyn Stor
         ST    R1,FILEDATA              Store returned addr
         SLR   R2,R2
         LA    R3,DDINFO                Point to DDINFO
         LR    R4,R1
         LA    R5,DDINFOS               Get no. of DDs
         USING FILEINFO,R4
OPEN#ØØ5 DS    ØH
         LA    RØ,MODOPENL              Get length of OPEN MODEL
         $STRGACQ R,LEN=Ø,LOC=31        ACQ dyn area
         ST    R1,FILEOPEN              Store returned addr
         MVC   Ø(MODOPENL,R1),MODOPEN   Move OPEN MODEL
         LA    RØ,MODCLOSL              Get Length of CLOSE MODEL
         $STRGACQ R,LEN=Ø,LOC=31        ACQ dyn area
         ST    R1,FILECLOS              Store returned addr
         MVC   Ø(MODCLOSL,R1),MODCLOS   Move close model
         LA    RØ,MODDCBEL              Get length of DCBE model
         $STRGACQ R,LEN=Ø,LOC=31        ACQ dyn area
         ST    R1,FILEDCBE              Store returned addr
         MVC   Ø(MODDCBEL,R1),MODDCBE   Move DCBE model
         USING DCBE,R1
         MVC   DCBEEODA,16(R3)          Save EOD address
         MVC   DCBESYNA,2Ø(R3)          Save SYNAD address
```

```
         DROP  R1
         CLI   8(R3),C'I'               Is this open for Input
         BNE   OPEN#Ø1Ø                 y:
         LA    RØ,MODDCBIL              Get input DCB length
         $STRGACQ R,LEN=Ø,LOC=24        ACQ dyn area
         ST    R1,FILEDCB               Store returned addr
         MVC   Ø(MODDCBIL,R1),MODDCBI   Move input DCB model
         B     OPEN#Ø15
OPEN#Ø1Ø DS    ØH
         LA    RØ,MODDCBOL              Get length of output DCB
         $STRGACQ R,LEN=Ø,LOC=24        ACQ dyn area
         ST    R1,FILEDCB               Store returned addr
         MVC   Ø(MODDCBOL,R1),MODDCBO   Move output DCB model
OPEN#Ø15 DS    ØH
         USING IHADCB,R1
         MVC   DCBDDNAM,Ø(R3)           Move DD name
         MVC   DCBDCBE,FILEDCBE         Move DCBE address
         DROP  R1
         L     RØ,12(,R3)               Move length of record
         $STRGACQ R,LEN=Ø,LOC=31        ACQ dyn area
         ST    R1,FILEREC               Store returned addr
         MVC   FILERECL,12(R3)          save lrecl
         XC    FILEIND,FILEIND          initialize indicator
         MVC   FILEHEDR,24(R3)          move header address
         L     R6,FILEDCB               point to file dcb
         L     R7,FILEOPEN              point to open area
         CLI   8(R3),C'I'               is this open for input ?
         BNE   OPEN#Ø2Ø                 n:
         OPEN  ((6),INPUT),MF=(E,(7)),MODE=31 open file
         B     OPEN#Ø25
OPEN#Ø2Ø DS    ØH
         OPEN  ((6),OUTPUT),MF=(E,(7)),MODE=31 open file
OPEN#Ø25 DS    ØH
         LTR   R15,R15                  Open success ?
         BZ    OPEN#Ø35                 Y:
         CLI   9(R3),C'M'               Was this a mandatory file ?
         BNE   OPEN#Ø4Ø                 n:
         LA    R2,4                     indicate RC=4
         B     OPEN#Ø4Ø
OPEN#Ø35 DS    ØH
         MVI   FILEIND,X'Ø1'            Indicate file open
         CLI   8(R3),C'I'               Was this input ?
         BE    OPEN#Ø4Ø                 y:
         LR    R1,R4                    point to offset
         BAS   R14,HDRW#ØØØ             Write header record
OPEN#Ø4Ø DS    ØH
         LA    R3,DDINFOL(,R3)          Bump to next dd
         LA    R4,FILESIZ(,R4)          Bump to next file
         BCT   R5,OPEN#ØØ5              open them
```

```
         LR    R15,R2                    R15->RC
OPEN#999 PR
         DROP  R4
*        Write header records to output file
HDRW#ØØØ BAKR  R14,Ø
         LR    R4,R1                     Point to offset
         L     RØ,FILEREC-FILEINFO(R4)   Get record addr
         L     R3,FILEDCB-FILEINFO(R4)   Get DCB addr
         L     R1,FILERECL-FILEINFO(R4)  Get record length
         L     R14,FILEHEDR-FILEINFO(R4) Get header rec addr
         LR    R15,R1                    length in R15
         ICM   R15,B'1ØØØ',BLANK         Default to ' '
         LTR   R14,R14                   Is header present ?
         BZ    HDRW#999                  n: bail out
         MVCL  RØ,R14                    Move record to output
         L     RØ,FILEREC-FILEINFO(R4)   point to record addr
         PUT   (3)                       WRite output
HDRW#999 PR                              goback
*        Close all files.
CLOS#ØØØ BAKR  R14,Ø
         L     R2,FILEDATA               Point to file information.
         LA    R3,DDINFOS                Get number of DDs
         USING FILEINFO,R2
CLOS#ØØ5 DS    ØH
         L     R6,FILEDCB                Get DCB addr
         L     R7,FILECLOS               Get close area
         CLI   FILEIND,X'ØØ'             Is this file open ?
         BE    CLOS#Ø1Ø                  n:
         CLOSE ((6)),MF=(E,(7)),MODE=31  Close file
CLOS#Ø1Ø DS    ØH
         L     R1,FILEOPEN               Point to open area
         $STRGREL R,ADDR=1               Release storage
         L     R1,FILECLOS               Point to close area
         $STRGREL R,ADDR=1               Release storage
         L     R1,FILEDCB                Point to dcb area
         $STRGREL R,ADDR=1               Release storage
         L     R1,FILEDCBE               Point to dcbe area
         $STRGREL R,ADDR=1               Release storage
         L     R1,FILEREC                Point to rec  area
         $STRGREL R,ADDR=1               Release storage
         MVI   FILEIND,X'ØØ'             Indicate file closed
         LA    R2,FILESIZ(,R2)           Bump to next
         BCT   R3,CLOS#ØØ5               Close them
CLOS#999 PR                              go back
         DROP  R2
PRTF#ØØØ BAKR  R14,Ø                     Write output to files
         M     RØ,=A(FILESIZ)            R1 points to offset
         A     R1,FILEDATA               add start offset
         CLI   FILEIND-FILEINFO(R1),X'ØØ' is this file open ?
```

```
         BE    PRTF#999                  n: goback
         L     R2,FILEREC-FILEINFO(R1)   Point to record addr
         L     R3,FILEDCB-FILEINFO(R1)   Point to DCB
         L     R4,FILERECL-FILEINFO(R1)  Get record length
         LR    RØ,R2
         LR    R1,R4
         LA    R14,PRTLINE               Point to PRTLINE
         LA    R15,L'PRTLINE             Get PRTLINE address
         ICM   R15,B'1ØØØ',BLANK         Default to ' '
         MVCL  RØ,R14                    Move record to output
         LR    RØ,R2                     RØ-> record
         PUT   (3)                       Put to record to file/.
PRTF#999 PR                              goback
HEDR#ØØØ BAKR  R14,Ø                     write header part of record
         MVC   PRTLINE(L'BLANK),BLANK    initialize prtline
         MVC   PRTLINE+1(L'PRTLINE-1),PRTLINE
         LA    R1,PRTLINE                point to prtline
         USING HEDR,R1                   let assembler know
         MVC   HEDRJNME,JOBNAME          move jobname
         MVC   HEDRJNUM,JOBNUM           move jobnumber
         MVC   HEDRSTDT,STRTDTE          move start date
         MVC   HEDRSTTM,STRTTME          move start time
         MVC   HEDRENDT,ENDDATE          move end date
         MVC   HEDRENTM,ENDTIME          move end time
         CLI   STEP,ISJOB                is this step info ?
         BE    HEDR#999                  n: goback
         MVC   HEDRSTNM,STEPNAME         move stepname
         MVC   HEDRPGNM,PROGRAM          move program name
HEDR#999 PR                              goback
SUDT#ØØØ BAKR  R14,Ø                     write SU details
         BAS   R14,HEDR#ØØØ              write header part of rec
         USING SMFRCD3Ø,R2               let assembler know
         CLC   SMF3ØPOF,=F'Ø'            is there info ?
         BE    SUDT#999                  n: goback
         CLC   SMF3ØPLN,=H'Ø'            is there info ?
         BE    SUDT#999                  n: goback
         CLC   SMF3ØPON,=H'Ø'            is there info ?
         BE    SUDT#999                  n: goback
         A     R2,SMF3ØPOF               Add offset to section.
         USING SMF3ØPRF,R2               Let assembler know
         LA    R1,PRTLINE                Point to prtline
         CLI   STEP,ISJOB                is this jobinformation ?
         BE    SUDT#ØØ5                  y:
         LA    R1,HEDRSTEP(,R1)          Bump step header info
         LA    R3,STEPSU                 indicate stepsu
         B     SUDT#Ø1Ø
SUDT#ØØ5 DS    ØH
         LA    R1,HEDRJOB(,R1)           Bump past job header info
         LA    R3,JOBSU                  indicate jobsu
```

```
SUDT#Ø1Ø DS    ØH
         USING SUDSECT,R1               Let assembler know
         MVC   SUTOT,EDMASKN            Move mask to su totals
         MVC   SUCPU,EDMASKN            Move mask to su cpu
         MVC   SUSRB,EDMASKN            Move mask to su srb
         MVC   SUIO,EDMASKN             Move mask to su io
         ICM   R14,15,SMF3ØSRV          get total su
         CVD   R14,DWORD                conver to decimal
         ED    SUTOT,DWORD+3            edit to su totals
         ICM   R14,15,SMF3ØCSU          get cpu   su
         CVD   R14,DWORD                conver to decimal
         ED    SUCPU,DWORD+3            edit to su cpu
         ICM   R14,15,SMF3ØSRB          get srb   su
         CVD   R14,DWORD                conver to decimal
         ED    SUSRB,DWORD+3            edit to su srb
         ICM   R14,15,SMF3ØIO           get io    su
         CVD   R14,DWORD                conver to decimal
         ED    SUIO,DWORD+3             edit to su io
         CLI   STEP,ISSTEP              is this step info ?
         BE    SUDT#Ø15                 y: goback
         MVC   SUWLM,SMF3ØWLM           move wlm class
         MVC   SUSRVNM,SMF3ØSCN         move wlm service name
         MVC   JOBWLMC,SMF3ØWLM         move wlm class
         MVC   JOBWLMS,SMF3ØSCN         move wlm service name
SUDT#Ø15 DS    ØH
         LR    R1,R3                    point to dd offset
         BAS   R14,PRTF#ØØØ             write this record
SUDT#999 PR                             goback
         DROP  R1,R2
TMDT#ØØØ BAKR  R14,Ø                    write TIME details
         BAS   R14,HEDR#ØØØ             write header part of rec
         USING SMFRCD3Ø,R2              let assembler know
         CLC   SMF3ØCOF,=F'Ø'           is there info ?
         BE    TMDT#999                 n: goback
         CLC   SMF3ØCLN,=H'Ø'           is there info ?
         BE    TMDT#999                 n: goback
         CLC   SMF3ØCON,=H'Ø'           is there info ?
         BE    TMDT#999                 n: goback
         ZAP   DEXCP,=P'Ø'              Initialize excp count
         CLC   SMF3ØUOF,=F'Ø'           is there info ?
         BE    TMDT#ØØ3                 n: goback
         CLC   SMF3ØULN,=H'Ø'           is there info ?
         BE    TMDT#ØØ3                 n: goback
         CLC   SMF3ØUON,=H'Ø'           is there info ?
         BE    TMDT#ØØ3                 n: goback
         LR    R3,R2
         A     R3,SMF3ØUOF
         L     R1,SMF3ØTEP-SMF3ØURA(,R3) Get excp count
         CVD   R1,DEXCP                 Convert to decimal
```

```
TMDT#ØØ3 DS    ØH
         A     R2,SMF3ØCOF              Add offset
         USING SMF3ØCAS,R2              Let assembler know
         LA    R4,PRTLINE               Point to prtline
         CLI   STEP,ISJOB               is this jobinformation ?
         BE    TMDT#ØØ5                 y:
         LA    R4,HEDRSTEP(,R4)         bump past step header
         LA    R3,STEPTIME              indicate steptime dd
         B     TMDT#Ø1Ø
TMDT#ØØ5 DS    ØH
         LA    R4,HEDRJOB(,R4)          bump past job header
         LA    R3,JOBTIME               indicate jobtime
TMDT#Ø1Ø DS    ØH
         USING TIMEDSCT,R4              let assembler know
         MVC   TIMEEXCP,EDMASKN         move edit mask to excp
         ICM   R1,15,SMF3ØCPT           Get tcb cpu time
         BAS   R14,CNVT#ØØØ             convert time
         MVC   TIMECPU,WORKTIME+2       write cpu time to output
         ICM   R1,15,SMF3ØCPS           Get tcb srb time
         BAS   R14,CNVT#ØØØ             convert time
         MVC   TIMESRB,WORKTIME+2       write srb time to output
         ED    TIMEEXCP,DEXCP+3         Edit excps
         CLI   STEP,ISSTEP              Is this step info ?
         BE    TMDT#Ø15
         MVC   TIMEWLM,JOBWLMC
         MVC   TIMESRVN,JOBWLMS
TMDT#Ø15 DS    ØH
         LR    R1,R3                    point to dd offset
         BAS   R14,PRTF#ØØØ             write this record
TMDT#999 PR                             goback
         DROP  R4,R2
STDT#ØØØ BAKR  R14,Ø                    write storage details
         BAS   R14,HEDR#ØØØ             write header part of output
         USING SMFRCD3Ø,R2              let assembler know
         CLC   SMF3ØROF,=F'Ø'           is there info ?
         BE    STDT#999                 n: goback
         CLC   SMF3ØRLN,=H'Ø'           is there info ?
         BE    STDT#999                 n: goback
         CLC   SMF3ØRON,=H'Ø'           is there info ?
         BE    STDT#999                 n: goback
         A     R2,SMF3ØROF              Add offset
         USING SMF3ØSAP,R2              Let assembler know
         LA    R4,PRTLINE               point to prtline
         CLI   STEP,ISJOB               is this job info ?
         BE    STDT#999                 y:goback
         LA    R4,HEDRSTEP(,R4)         bump step header info
         LA    R3,STEPSTOR              indicate stepstor
         USING STORDSCT,R4              let assembler know
         MVC   STORUB16,EDMASKN         Move edit mask to UStor<16M
```

```
         MVC   STORUA16,EDMASKN          Move edit mask to UStor>16M
         MVC   STORSB16,EDMASKN          Move edit mask to SStor<16M
         MVC   STORSA16,EDMASKN          Move edit mask to SStor>16M
         ICM   R14,B'1111',SMF30URB      Get Ustor<16M
         SRL   R14,10                    Convert to KB
         CVD   R14,DWORD                 Convert to decimal
         ED    STORUB16,DWORD+3          Edit to output area
         ICM   R14,B'1111',SMF30EUR      Get Ustor>16M
         SRL   R14,10                    Convert to KB
         CVD   R14,DWORD                 Convert to decimal
         ED    STORUA16,DWORD+3          Edit to output area
         ICM   R14,B'1111',SMF30ARB      Get Sstor<16M
         SRL   R14,10                    Convert to KB
         CVD   R14,DWORD                 Convert to decimal
         ED    STORSB16,DWORD+3          Edit to output area
         ICM   R14,B'1111',SMF30EAR      Get Sstor>16M
         SRL   R14,10                    Convert to KB
         CVD   R14,DWORD                 Convert to decimal
         ED    STORSA16,DWORD+3          Edit to output area
         LR    R1,R3                     Point to dd offset
         BAS   R14,PRTF#000              write this output rec
STDT#999 PR                              goback
         DROP  R4,R2
CNVT#000 BAKR  R14,0                     Convert time to HH:MM:SS
         SLR   R0,R0                     R1-> time in 100th of sec
         D     R0,=F'360000'             Get hours
         CVD   R1,DWORD                  Convert to decimal
         SRP   DWORD,4,0                 000000000HH0000C
         ZAP   WTIME,DWORD               Save it for later
         LR    R1,R0                     Get reminder
         SLR   R0,R0
         D     R0,=F'6000'               Get minutes
         CVD   R1,DWORD                  Convert to decimal
         SRP   DWORD,2,0                 0000000000MM00C
         AP    WTIME,DWORD               Add to saved time
         LR    R1,R0                     Get reminder
         SLR   R0,R0
         D     R0,=F'100'                Get seconds
         CVD   R1,DWORD                  Convert to decimal
         AP    WTIME,DWORD               Add to saved time
         MVC   WORKTIME,EDMASKT          Move edit mask for time
         ED    WORKTIME,WTIME+4          edit time
CNVT#999 PR                              goback
*        Convert date from 0nYYDDDF to CCYY-MM-DD format
CNVD#000 BAKR  R14,0
         ZAP   DWORD,0(4,R1)             R1 points to input date
         OI    DWORD+L'DWORD-1,X'0F'     Make pack!
         AP    DWORD,=P'1900000'         Add 1900000
         ZAP   WDATE,DWORD               Save for later
```

```
         SRP   DWORD,64-3,Ø             Remove days
         CVB   R1,DWORD                 Convert to binary
         SLR   RØ,RØ
         D     RØ,=F'4'                 Divide by 4
         C     RØ,=F'Ø'                 Any reminder ?
         BE    CNVD#ØØ5                 y: is leap year
         L     R1,=A(NLEAPYR)           not a leap year
         B     CNVD#Ø1Ø
CNVD#ØØ5 DS    ØH
         L     R1,=A(LEAPYR)            is a leap year
CNVD#Ø1Ø DS    ØH
         ZAP   DWORD,WDATE              restore date
         SRP   WDATE,64-3,Ø             remove days
         SRP   WDATE,4,Ø                ØØØØØØØCCYYØØØØF
         SRP   DWORD,12,Ø               DDDØØØØØØØØØØØØF
         SRP   DWORD,64-12,Ø            ØØØØØØØØØØØØDDDF
         AP    WDATE,=P'1ØØ'            Indicate 1st month
CNVD#Ø15 DS    ØH
         CP    DWORD,Ø(2,R1)            is the days < table
         BNH   CNVD#Ø2Ø                 n:
         AP    WDATE,=P'1ØØ'            Add 1 month
         SP    DWORD,Ø(2,R1)            Subtract date in month
         LA    R1,2(,R1)                Bump to next table entry
         B     CNVD#Ø15                 loop back
CNVD#Ø2Ø DS    ØH
         AP    WDATE,DWORD              Add date
         MVC   WORKDATE,EDMASKD         Move mask for date
         ED    WORKDATE,WDATE+3         Edit date
CNVD#999 PR                            goback
*        Literals
Z        DC    CL1'*'                   zero or more char
O        DC    CL1'?'                   exactly 1 char
LEAPYR   DC    P'31,29,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø,31,3Ø,31' leap table
NLEAPYR  DC    P'31,28,31,3Ø,31,3Ø,31,31,3Ø,31,3Ø,31,3Ø,31'
EDMASKN  DC    X'4020202020202020212Ø'  Edit mask for number
EDMASKT  DC    X'402120207A20207A2020'  Edit mask for time
EDMASKD  DC    X'4021202020206020206020Ø' Edit mask for date
BLANK    DC    C'       '               Blanks
         DS    ØF                       DD table
DDINFO   DC    C'SYSIN   ',C'I',C'O',A(8Ø),A(SMFP#Ø1Ø),A(SMFP#Ø15)
         DC    A(Ø)
SYSIN    EQU   Ø
DDINFOL  EQU   *-DDINFO
         DC    C'SMFIN   ',C'I',C'M',A(6ØØØØ),A(SMFP#Ø3Ø),A(SMFP#Ø35)
         DC    A(Ø)
SMFIN    EQU   1
         DC    C'STEPSU  ',C'O',C'O',A(133),A(Ø),A(Ø)
         DC    A(SUSHDR)
STEPSU   EQU   2
```

```
         DC    C'JOBSU   ',C'O',C'O',A(133),A(Ø),A(Ø)
         DC    A(SUJHDR)
JOBSU    EQU   3
         DC    C'STEPTIME',C'O',C'O',A(133),A(Ø),A(Ø)
         DC    A(TMSHDR)
STEPTIME EQU   4
         DC    C'JOBTIME ',C'O',C'O',A(133),A(Ø),A(Ø)
         DC    A(TMJHDR)
JOBTIME  EQU   5
         DC    C'STEPSTOR',C'O',C'O',A(133),A(Ø),A(Ø)
         DC    A(STSHDR)
STEPSTOR EQU   6
DDINFOS  EQU   (*-DDINFO)/DDINFOL
         DC    X'FFFFFFFF'
MODDCBO  DCB   DCBE=MODDCBE,MACRF=(PM),DSORG=PS,LRECL=133 output DCB
MODDCBOL EQU   *-MODDCBO
MODDCBI  DCB   DCBE=MODDCBE,DDNAME=Ø,MACRF=(GM),DSORG=PS  Input DCB
MODDCBIL EQU   *-MODDCBI
MODDCBE  DCBE  RMODE31=BUFF,EODAD=Ø,SYNAD=Ø   Model DCBE
MODDCBEL EQU   *-MODDCBE
MODOPEN  OPEN  (,),MF=L,MODE=31               Model OPEN
MODOPENL EQU   *-MODOPEN
MODCLOS  CLOSE (,),MF=L,MODE=31               Model CLOSE
MODCLOSL EQU   *-MODCLOS
*
         LTORG
SMF3ØPG  CSECT
SUSHDR   DS    ØF                    Header information
         DC    C' JobName  JobNum   Start Date St Time  End D'
         DC    C'ate   End Time StepName Program    Total SU '
         DC    C'    CPU SU     SRB SU      IO SU             '
SUJHDR   DS    ØF
         DC    C' JobName  JobNum   Start Date St Time  End D'
         DC    C'ate   End Time   Total SU     CPU SU     SRB'
         DC    C' SU      IO SU WLM Cls  Serv Name            '
TMSHDR   DS    ØF
         DC    C' JobName  JobNum   Start Date St Time  End D'
         DC    C'ate   End Time StepName Program  TCB CPU  SR'
         DC    C'B CPU        EXCP                           '
TMJHDR   DS    ØF
         DC    C' JobName  JobNum   Start Date St Time  End D'
         DC    C'ate   End Time TCB CPU  SRB CPU        EXCP '
         DC    C'WLM Cls  Serv Name                          '
STSHDR   DS    ØF
         DC    C' JobName  JobNum   Start Date St Time  End D'
         DC    C'ate   End Time StepName Program    USR<16(K)'
         DC    C'  USR>16(K) SYS<16(K)  SYS>16(K)            '
$DYN     DSECT                       Dynamic area
SAVEAREA DS    18F                   Save area
```

```
FILEDATA DS    F                    Pointer to file information
MASKLIST DS    F                    Pointer to mask list
RETCODE  DS    F                    Return code
DWORD    DS    D                    work
DEXCP    DS    D                    work excp count
WDATE    DS    D                    work date
WTIME    DS    D                    work time
WORKTIME DS    CL(L'EDMASKT)        work area
WORKDATE DS    CL(L'EDMASKD)        work area
JOBWLMC  DS    CLØ8                 wlm class
JOBWLMS  DS    CLØ8                 wlm serice name
STRTDTE  DS    CL1Ø                 Start date
ENDDATE  DS    CL1Ø                 End  date
STRTTME  DS    CLØ8                 Start Time
ENDTIME  DS    CLØ8                 End time
JOBNAME  DS    CLØ8                 Jobname
JOBNUM   DS    CLØ8                 Job Number
STEPNAME DS    CLØ8                 Step Name
PROGRAM  DS    CLØ8                 Program name
STEP     DS    X                    Step indicator
ISSTEP   EQU   X'ØØ'                ...Is step information
ISJOB    EQU   X'Ø1'                ...Is job information
INCL     DS    X                    Include indicator
INCLALL  EQU   X'Ø1'                ...Include all
PRTLINE  DS    CL133                Print line
         ASAXWC    MF=(L,MYLIST)
WORKAREA DS     CL256               WorkArea
$DYNL    EQU   *-$DYN               Length of Dyn area
FILEINFO DSECT                     DSECT for file area
FILEOPEN DS    F                    Open area addr
FILECLOS DS    F                    Close area addr
FILEDCB  DS    F                    DCB addr
FILEDCBE DS    F                    DCBE addr
FILEREC  DS    F                    rec addr
FILERECL DS    F                    rec length
FILEHEDR DS    F                    header addr
FILEIND  DS    X                    indicator
         DS    CL3                  filler
FILESIZ  EQU   *-FILEINFO           length of file area
HEDR     DSECT                     Heeader information for rec
         DS    CL1
HEDRJNME DS    CL8                  Jobname
         DS    CL1
HEDRJNUM DS    CL8                  Jobnumber
         DS    CL1
HEDRSTDT DS    CL1Ø                 Start date
         DS    CL1
HEDRSTTM DS    CLØ8                 Start Time
         DS    CL1
```

```
HEDRENDT DS    CL1Ø                End date
         DS    CL1
HEDRENTM DS    CLØ8                End time
         DS    CL1
HEDRJOB  EQU   *-HEDR              Length of job header
HEDRSTNM DS    CLØ8                Step Name
         DS    CL1
HEDRPGNM DS    CLØ8                Program Name
         DS    CL1
HEDRSTEP EQU   *-HEDR              Length of Step Header
HEDRL    EQU   *-HEDR              Total length of header
SUDSECT  DSECT                     Dsect for SU details
SUTOT    DS    CL1Ø                Total SU units
         DS    CL1
SUCPU    DS    CL1Ø                CPU SU units
         DS    CL1
SUSRB    DS    CL1Ø                SRB SU units
         DS    CL1
SUIO     DS    CL1Ø                IO su units
         DS    CL1
SUWLM    DS    CLØ8                WLM class name
         DS    CL1
SUSRVNM  DS    CLØ8                WLM serv class
SUL      EQU   *-SUDSECT           Length
TIMEDSCT DSECT                     Dsect for TIME details
TIMECPU  DS    CLØ8                TCB CPU time
         DS    CL1
TIMESRB  DS    CLØ8                TCP SRB time
         DS    CL1
TIMEEXCP DS    CL1Ø                Total EXCP
         DS    CL1
TIMEWLM  DS    CLØ8                WLM class name
         DS    CL1
TIMESRVN DS    CLØ8                WLM serv class
TIMEL    EQU   *-TIMEDSCT          Length
STORDSCT DSECT                     Dsect for Storage section
STORUB16 DS    CL1Ø                User stor < 16M
         DS    CL1
STORUA16 DS    CL1Ø                User stor > 16M
         DS    CL1
STORSB16 DS    CL1Ø                Sys stor < 16M
         DS    CL1
STORSA16 DS    CL1Ø                Sys Stor > 16M
STORL    EQU   *-STORDSCT          Length
MASKDSCT DSECT                     Dsect for Mask details
MASKNEXT DS    F                   Ptr to Next mask info elem
MASKNAML DS    F                   Length of Mask
MASKNAME DS    CL8                 Name of Mask
MASKL    EQU   *-MASKDSCT          Length
         DCBD DSORG=PS,DEVD=DA
```

```
IHADCBE
IFASMFR 3Ø
END    SMF3ØPG
```

Ganesh Rao and Pranav Sampat
Consultants (USA)

Analysing data-in-virtual statistics

INTRODUCTION

Since its introduction a long time ago (with MVS/XA!), when it received some attention, DIV (Data-In-Virtual) seems to have fallen into oblivion. The main reason for that is the fact that DIV is somewhat difficult to use because the Assembly-language primitive functions one must use are not readily available in high-level languages. However, DIV, which is a set of primitive functions, enables an application program to load and manage substantial amounts of data into memory from a VSAM Linear DataSet (LDS). The LDS itself can grow to 4GB and the program can map up to (almost) 2GB of it at a time in central memory . Applications can create, read, and update data without the I/O buffer, blocksize, and record considerations that the traditional GET and PUT types of access method require. An application written for data-in-virtual views its permanent storage data as a seamless body of data without internal record boundaries. Among the applications that can be considered for a data-in-virtual implementation are applications that process large arrays, VSAM relative record applications, and BDAM fixed-length record applications. The potential benefits may be realized eventually as DIV merges with hiperspaces (a related concept) and as subsystems, languages, and application packages exploit the DIV benefits. For example, DIV is used by DFSMS when I/O to SMS control datasets is needed.

The data-in-virtual services process the application data in 4096-byte (4KB) units on 4KB boundaries called blocks. The application data resides in what is called a data-in-virtual object, a data object, or simply an object. The data-in-virtual object is a continuous string of uninterrupted data. When one writes an application using the techniques of data-in-virtual, the I/O takes place only for the data referenced and saved. Only the referenced pages of such an object are brought into virtual storage. Bytes of the mapped pages can be accessed and changed in normal program execution without regard to the need for updating. On request, or at the time the connection to the DIV object is terminated, only the changed pages are written back to the linear dataset. If one runs an application using conventional access methods, and then runs it again using data-in-virtual, one will notice a difference in performance. This difference depends on both the size of the dataset and its access pattern. To gain the right to view or update the object, an application must use the ACCESS service. ACCESS is similar to the OPEN macro of VSAM. It has a mode parameter of READ or UPDATE, and it gives your application the right to read or update the object. If the application has finished processing the object, it uses UNACCESS to relinquish access to the object.

Before using the DIV macro to process a linear dataset object (or a hiperspace object), one must create the dataset (or the hiperspace). *OS/390 MVS Programming:Authorized Assembler Services Guide (*GC28-1763) explains how to use DIV macro functions. The 'how to' reference for hiperspaces is the *Extended Addressability Guide,* (GC28-1468). Also, it is worth consulting *An Introduction to Data-in-Virtual* (GG66-0259), which may be a bit old but provides a few Assembler, Fortran, and PL/I examples.

COLLECTING DIV DATA

When enabled by the SMFPRM*xx* TYPE parameter, SMF creates record type 41, which provides resource usage information regarding data-in-virtual. There are two subtypes of this SMF record: subtype 1 is an ACCESS record – the ACCESS data

section is written when a DIV object is accessed; subtype 2 is an UNACCESS record – the counts for the I/O activity section are accumulated by data-in-virtual while the object is in use and are reported at the time of the UNACCESS request. The subtype 2 record is written whenever a data-in-virtual object is unaccessed.

A detailed description of layout of SMF type 41 record and its subtypes can be obtained from the *MVS System Management Facilities (SMF)* (SA22-7630-03) manual. You can also find the subtype descriptions in the macro ITVSMF41 in SYS1. MACLIB.

Based on record descriptions obtained from above mentioned manual a simple DIV report writer was written.

CODE

The code is a two part stream. In the first part (COPY412), selected SMF records (selection being defined by INCLUDEs condition) are copied from an SMF dataset to a VB file, which is the input file for the second step.

In the second step (DIV412), the captured records are formatted by invoking REXX EXEC (DIV412), and a report produced. The report shows the users performing access/unaccess of an object, along with the timestamp of when the object was accessed/ unaccessed, the size of the object, and its read/write/re-read count data. Elapsed time during which the object was read or updated is calculated from two TOD timestamps.

JCL

```
//DIVJOB    JOB ACCT#,
//              MSGLEVEL=(1,1),
//              MSGCLASS=R,
//              NOTIFY=&SYSUID
//COPY412  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DISP=SHR,DISP=hlq.SMFDUMPW
//SMF412   DD DSN=your.copied.by.sort.to.VB.smf.dataset,
//         SPACE=(CYL,(1)),UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
```

```
//TOOLIN   DD *
  COPY FROM(RAWSMF) TO(SMF412) USING(SMFI)
//SMFICNTL DD *
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,41,AND,23,2,BI,EQ,2)
/*
//DIV412  EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50,PARM='%DIV412'
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.library
//SMF      DD DISP=(SHR,PASS),DSN=your.copied.by.sort.to.VB.smf.dataset,
//DIV41    DD DSN=your.div.report.dataset,
//         SPACE=(CYL,(1,1)),UNIT=SYSDA,
//         DISP=(NEW,KEEP),DCB=(RECFM=FB,LRECL=150)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
/*
```

DIV412 EXEC

```
/* REXX EXEC to read and format SMF record 41.2*/
ADDRESS TSO
/*-------------------------------------------------------------*/
/* Print report header and labels                              */
/*-------------------------------------------------------------*/
 Out.1 = left(' ',30,' '),
         ||center('Data in Virtual Report ',22,),
         ||left(' ',15,' ')
 Out.2 = left(' ',20,' '),
         ||center('Report produced on',18,),
         ||left(' ',1,' ')||left(date(),11),
         ||left(' ',1,' ')||left('at ',3,' '),
         ||left(time(),10)
 Out.3 = '  '
 Out.4 = left('SMF date',11) left('SMF time',9),
         left('SID',4)       left('DFP lvl.',7),
         left('Job name',9)  left('DD name',9),
         left('Access Time',15) left('Unaccess Time',15),
         left('A.Size',6)    left('U.Size',7),
         left('Mode',5)      left('Elaps.sec.',10),
         left('Block',5)     left('Block',6),
         left('Block',6)     left('Read',5),
         left('Write',5)
 Out.5 = left(' ',118)       left('read',5),
         left('write',6)     left('rread',6),
         left('I/O',5)       left('I/O',3)
 Out.6 = LEFT('-',149,'-')
 "EXECIO * DISKW DIV41 (STEM Out.)"
 'EXECIO * DISKR SMF ( STEM x. FINIS'
```

```
  do i = 1 to x.Ø
/*-------------------------------------------------------------*/
/*           Header/Self-defining Section                      */
/*-------------------------------------------------------------*/
smftype  = c2d(SUBSTR(x.i,2,1))             /* SMF record type */
smfstype = c2d(SUBSTR(x.i,19,2))            /* Record subtype  */
IF smftype = '41'  Then
Do
 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 */
   trp   = c2d(SUBSTR(x.i,21,2))   /*        number of triplets*/
   opd   = c2d(SUBSTR(x.i,25,4))   /* offset to product section*/
   lpd   = c2d(SUBSTR(x.i,29,2))   /* length of product section*/
   npd   = c2d(SUBSTR(x.i,31,2))   /*number of product sections*/
   od1   = c2d(SUBSTR(x.i,33,4))   /*  offset to access section*/
   ld1   = c2d(SUBSTR(x.i,37,2))   /*  length of access section*/
   nd1   = c2d(SUBSTR(x.i,39,2))   /* number of access sections*/
   od2   = c2d(SUBSTR(x.i,41,4))   /*offset to unaccess section*/
   ld2   = c2d(SUBSTR(x.i,45,2))   /*length of unaccess section*/
   nd2   = c2d(SUBSTR(x.i,47,2))  /*number of unaccess sections*/
   od3   = c2d(SUBSTR(x.i,49,4))   /*    offset to i/o activity*/
   ld3   = c2d(SUBSTR(x.i,53,2))   /*    length of i/o activity*/
   nd3   = c2d(SUBSTR(x.i,55,2))   /*    number of i/o activity*/
/*-------------------------------------------------------------*/
/*           Product Section                                   */
/*-------------------------------------------------------------*/
  IF opd <> Ø AND npd <> Ø Then do
     opd=opd-3
     dfplvl =  SUBSTR(x.i,opd,8)            /*   product level */
     prod   =  SUBSTR(x.i,opd+8,16)         /*  component name */
  end
/*-------------------------------------------------------------*/
/*           Object ACCESS Data Section                        */
/*-------------------------------------------------------------*/
  IF od1 <> Ø and nd1 <> Ø Then do
     od1=od1-3
     dda =  SUBSTR(x.i,od1,8)                  /*  object ddname*/
     aza =  c2d(SUBSTR(x.i,od1+8,4))           /*  object size  */
     ata=   SUBSTR(ct(c2x(SUBSTR(x.i,od1+12,4))),11,15) /* TOD  */
     tya =  c2d(SUBSTR(x.i,od1+16,1))          /*  object type  */
     ama =  c2d(SUBSTR(x.i,od1+17,1))          /*  access mode  */
     jbn =  SUBSTR(x.i,od1+18,8)        /* jobname/started task */
         SELECT
           when ama=1 then mode='Read'
           when ama=2 then mode='Update'
         END
    end
/*-------------------------------------------------------------*/
/*           Object UNACCESS Data Section                      */
/*-------------------------------------------------------------*/
```

```
    IF od2 <> Ø and nd2 <> and then do
       od2=od2-3
       uzu =  c2d(SUBSTR(x.i,od2,4))              /* object size*/
       utu =  SUBSTR(ct(c2x(SUBSTR(x.i,od2+4,4))),11,15)   /* TOD*/
      end
  /*-------------------------------------------------------------*/
  /*          Object I/O Activity Section                        */
  /*-------------------------------------------------------------*/
    IF od3 <> Ø and nd3 <> Ø then do
       od3=od3-3
       brd =  c2d(SUBSTR(x.i,od3,4))       /*    tot.no. of reads*/
       bwr =  c2d(SUBSTR(x.i,od3+4,4))     /*   tot.no. of writes*/
       brr =  c2d(SUBSTR(x.i,od3+8,4))     /* tot.no. of re-reads*/
       inc =  c2d(SUBSTR(x.i,od3+12,4))  /*tot.no.of i/o for read*/
       ouc =  c2d(SUBSTR(x.i,od3+16,4)) /*tot.no.of i/o for write*/
    end
    timedif=dif(utu,ata)      /* how long the object was accessed*/
  /*-------------------------------------------------------------*/
  /* formatting and printing a DIV entry                         */
  /*-------------------------------------------------------------*/
  divout = left(Date('N',smfdate,'J'),11) left(smftime,9),
           left(sysid,4)  left(dfplvl,8)  left(jbn,8),
           left(dda,8)    right(ata,15)   left(utu,17),
           left(aza,6)    left(uzu,6)     left(mode,7),
           right(timedif,8),
           right(brd,4)   right(bwr,5)    right(brr,5),
           right(inc,5)   right(ouc,5)
   PUSH divout
     "EXECIO 1 DISKW DIV41"
     end
   end
exit
```

SMF PROCEDURE

```
/* REXX - convert a 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 time*/
    return otime
```

CT PROCEDURE

```
/*                                                  */
/* TOD timestamp is a 16-byte EBCDIC representat */
/* The BLSUXTOD proc is described in "z/OS       */
/* V1R3 MVS IPCS Customization"                    */
/*                                                  */
arg todtime
If todtime    <> '0000000000000000' Then
  Do
    TOD_Value = X2C(todtime)
    Returned_Date = '-------------------------'
    address LINKPGM "BLSUXTOD TOD_Value Returned_Date"
  End
Else
  Returned_Date = ''
Return Returned_Date
```

DIF PROCEDURE

```
/*                                                  */
/* Dif: REXX subroutine to find the             */
/* difference between two timestamps, in this    */
/* case in seconds                                  */
/*                                                  */
arg time2,time1
 parse var time2 h2 ':' m2 ':' s2 ':' t2
 parse var time1 h1 ':' m1 ':' s1 ':' t1
 tot2 = h2*3600 + m2*60 + s2
 tot1 = h1*3600 + m1*60 + s1
 es = tot2 -tot1
 if es<0 then es=es+86400
 eh=es%3600
 es=es//3600
 ex=es
 em=es%60
 es=es//60
/* et=right(eh,2,0)':'right(em,2,0)':'right(es,2,0) */
 return ex
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Waiting for datasets

We have a group of FTP datasets ‘pushed’ to us – the first part of the dataset name is known, but the trailing qualifier is variable. The following program is started by our automation package, and is the name of a volume to look on and some further variable (time-based) information that is then used for a ‘cvaffilt’ call, which passes back the names (if any) of datasets found that match our pattern. We then extract the dataset names and write them to a dataset that is passed to the next step (obviously this could be replaced by a WTO/WTOS, a command being issued, or any other relevant action). The program is written to look for three datasets, and will wait for five minutes for them to arrive.

WAIT4FTP PROGRAM

```
         TITLE 'WAIT4FTP - WAIT FOR INCOMING FTP DATASETS'
*********************************************************************
*                                                                   *
* WAIT4FTP: SCAN A VOLUME, USING A DSN PATTERN, TO SEE HOW MANY     *
*           DATASETS ARE PRESENT. WAIT FOR UP TO 5 MINUTES TO SEE   *
*           IF THREE HAVE ARRIVED. IF NOT, EXIT WITH RC.            *
*                                                                   *
*           FOR FTP TRANSFER OF DATASETS.                           *
*                                                                   *
* PARMS:    TWO PARMS, BOTH MANDATORY, SEPARATED BY A COMMA -       *
*           1/ A VOLID (TO LOOK ON FOR THE DATASETS)                *
*           2/ 4 DIGITS, TO BE INSERTED INTO THE DATASET MASK. THIS *
*              MASK WILL MAKE UP A DATASET NAME, IN THE FORM:       *
*                                                                   *
*                FTPXFER.DATA.FILE.PFTP.DNNNNTY.**                  *
*                                                                   *
*              WHERE 'NNNN' ARE THE 4 DIGITS PASSED TO US.          *
*                                                                   *
* OUTPUT:   DATASET CONTAINING THE NAMES OF THE DATASETS LOCATED,   *
*           WRITTEN TO DDNAME 'DSNAMES'.                            *
*                                                                   *
*********************************************************************
         PRINT NOGEN
*********************************************************************
* HOUSEKEEPING...                                                  *
*********************************************************************
WAIT4FTP CSECT
```

```
WAIT4FTP AMODE 31
WAIT4FTP RMODE 24
         BAKR  R14,Ø                    SAVE CALLER DATA ON STACK
         LR    R12,R15                  GET ENTRY POINT
         LA    R11,2Ø48(R12)            LOAD SECOND BASE
         LA    R11,2Ø48(R11)
         USING WAIT4FTP,R12,R11         ADDRESSABILITY
         L     R9,Ø(R1)                 GET A(PARMS)
         LTR   R9,R9                    ANY?
         BZ    BADPARMS                 NO...
         CLC   Ø(2,R9),=H'11'           11-BYTE PARMS PASSED?
         BNE   BADPARMS                 NO...
         CLI   8(R9),C','               COMMA IN BETWEEN?
         BNE   BADPARMS                 NO...
         MVC   VOLID(6),2(R9)           SAVE VOLID...
         MVC   DSNSER(4),9(R9)          ...AND SERIAL
         MVC   WTO1+23(6),VOLID         MOVE VOLID TO WTO
         MVC   WTO1+35(L'DSNAME),DSNAME MOVE MASK TO WTO
WTO1     WTO   'WAIT4FTP- SCAN XXXXXX FOR: ............................X
               ........',ROUTCDE=11
*********************************************************************
* INITIALIZE BUFFER LIST HEADER (BFLH) AND ELEMENTS (BFLE)...       *
*********************************************************************
INITBFL  DS    ØH
*
         LA    R2,BFLHDEF               ADDRESS WORKAREA
         LA    R3,BFLSIZE               LOAD LENGTH TO CLEAR
         XR    R5,R5                    Ø PADDING BYTE
         MVCL  R2,R4                    SET BUFFER LIST AREA TO Ø'S
*
         LA    R1,BFLHDEF               TEMP ADDRESSABILITY TO BFLH
         USING BFLMAP,R1
         MVI   BFLHNOE,BUFFNUM          SET NUMBER OF BUFFER ELEMENTS
         OI    BFLHFL,BFLHDSCB          ID AS DSCB BUFFER ELEMENT LIST
         LA    R2,BFLHDEF+BFLHLN        R2 -> 1ST BUFFER LIST ELEMENT
         USING BFLE,R2                  TEMP ADDRESSABILITY
         LA    R3,DSCBDEF               R3 -> 1ST DSCB BUFFER
         LA    R4,BUFFNUM               R4 = NO. OF ELEMENTS & BUFFERS
BFLEINIT DS    ØH
         OI    BFLEFL,BFLECHR           REQUEST CCHHR ON RETURN
         MVI   BFLELTH,DSCBSIZE         SET BUFF LEN TO FULL DSCB SIZE
         ST    R3,BFLEBUF               SET A(DSCB BUFFER)
         LA    R2,BFLELN(R2)            R2 -> NEXT BUFFER LIST ELEMENT
         LA    R3,DSCBSIZE(R3)          R3 -> NEXT DSCB BUFFER
         BCT   R4,BFLEINIT              LOOP THROUGH ALL ELEMENTS
         DROP  R1,R2                    DROP TEMP ADDRESSABILITY
*********************************************************************
* INITIALIZE THE FILTER CRITERIA LIST (FCL) HEADER AND ELEMENT...   *
*********************************************************************
         XC    FCLDEF(FCLSIZE),FCLDEF   SET FCL AREA TO Ø
```

```
         LA    R1,FCLDEF               R1 -> FCL HEADER
         USING FCLMAP,R1               TEMP ADDRESSABILITY TO FCL
         MVC   FCLID,=C'FCL '          SET EYECATCHER 'FCL '
         MVC   FCLCOUNT,=H'1'          SET NUMBER OF FCL ELEMENTS = 1
         OI    FCL1FLAG,FCL1EQF1       RETURN ONLY FORMAT1 DSCBS
         LA    R2,FCLHDEND             R2 -> 1ST (ONLY) FCL ELEMENT
         USING FCLDSN,R2               TEMP ADDRESSABILITY
         MVC   FCLDSNLG,DSNAMELN       SET LENGTH OF DSN PATTERN
         LA    R3,DSNAME               GET DSN PATTERN ADDRESS
         ST    R3,FCLDSNA              ...SAVE IN FCL
         DROP  R1,R2
*********************************************************************
* SCAN THROUGH THE UCBS, LOOKING FOR OUR VOLUME.                    *
*********************************************************************
SCANVOLS DS    ØH
         USING UCBOB,R4                ADDRESSABILITY TO UCB
         LA    R4,UCBAREA              LOCATE UCB WORKAREA
         XC    UCBWORK,UCBWORK         +INITIALIZE UCBSCAN WORKAREA
UCBLOOP  DS    ØH
*
         UCBSCAN COPY,                                              X
               WORKAREA=UCBWORK,                                    X
               UCBAREA=UCBAREA,                                     X
               DCEAREA=NONE,                                        X
               DCELEN=Ø,                                            X
               VOLSER=VOLID,           SELECT BY VOLSER             X
               DEVN=Ø,                 START WITH FIRST UCB         X
               DYNAMIC=YES,            INCLUDE DYNAMICALLY ADDED UCBS X
               RANGE=ALL,              *ALL* UCBS (3&4 DIGIT)       X
               NONBASE=NO,                                          X
               DEVCLASS=DASD,          DASD ONLY                    X
               DEVCID=Ø,               DON'T SELECT BY DEVICE CHAR. X
               IOCTOKEN=NONE,          NO IODEVICE TABLE TOKEN      X
               LINKAGE=SYSTEM,         USE PC CALL                  X
               PLISTVER=MAX
*
         LTR   R15,R15                 GOT UCB OK?
         BZ    GETREAL                 YES..GET THE REAL UCB ADDRESS
         C     R15,=F'4'               END OF UCBS?
         BE    NOTFND                  YES..SAY WE COULDN'T FIND VOL
         B     BADCALL                 NO...ERROR
*********************************************************************
* NOW WE HAVE OUR VOLUME - GET THE *REAL* UCB ADDRESS BY CALLING THE  *
* "UCBLOOK" MACRO...                                                *
*********************************************************************
GETREAL  DS    ØH
*
         MODESET KEY=ZERO,MODE=SUP     SUPERVISOR STATE FOR "UCBLOOK"
*
UCBLOOK  UCBLOOK VOLSER=UCBVOLI,       USE VOLID FROM UCB COPY      X
```

```
                UCBPTR=UCBADDR,          SAVE UCB ADDRESS IN HERE       X
                LOC=ANY,                 UCB CAN BE ANYWHERE            X
                NOPIN,                   DON'T PIN UCB                  X
                RANGE=ALL,               3&4 DIGIT UCBS                 X
                DEVCLASS=DASD,           MAKE SURE ITS A DISK!          X
                DYNAMIC=YES              CHECK DYNAMIC UCBS
*
          ST    R15,SAVER15              SAVE RC
*
          MODESET KEY=NZERO,MODE=PROB    BACK TO NORMAL
*
          L     R15,SAVER15              RELOAD RC
          LTR   R15,R15                  OK?
          BNZ   LOOKERR                  NO...PANIC...
*********************************************************************
* NOW WE HAVE THE VOLUME WE WANT - ISSUE A CVAFFILT 'ACCESS=READ'     *
* REQUEST TO SEE IF THERE ARE ANY RELEVANT FTP DATASETS, AND, IF SO,  *
* HOW MANY...                                                         *
*********************************************************************
          LA    R7,5                     WAIT UP TO 5 TIMES
CVPLLOOP  DS    ØH
*
CVPL      CVAFFILT ACCESS=READ,                                        X
                FCL=FCLDEF,                                          X
                BUFLIST=BFLHDEF,                                     X
                UCB=UCBADDR
*
          C     R15,F4                   WAS RC=4?
          BE    MOREBUFF                 YES..NEED MORE BUFFERS
          LTR   R15,R15                  RC OTHER THAN Ø?
          BNZ   CVAFERR                  YES...DODGY
          LA    R3,FCLDEF                R3 -> FCL HEADER
          USING FCLMAP,R3                TEMP ADDRESSABILITY TO FCL
          LH    R2,FCLDSCBR              GET DSCB COUNT
          CVD   R2,DWORD                 MAKE IT DECIMAL
          UNPK  UNPKFLD(5),DWORD+5(3)    UNPACK IT
          OI    UNPKFLD+4,X'FØ'          SET CORRECT SIGN
          MVC   WTO2+49(1),UNPKFLD+4
WTO2      WTO   'WAIT4FTP- NUMBER OF FTP DATASETS LOCATED=?',ROUTCDE=11
          CLC   FCLDSCBR,=H'3'           GOT ALL 3?
          BE    GOTEM                    YES..CARRY ON
WTO3      WTO   'WAIT4FTP- NOT ALL DATASETS HAVE ARRIVED - WAITING...', X
                ROUTCDE=11
          STIMER WAIT,BINTVL=BIN6Ø       WAIT 6Ø SECONDS
          BCT   R7,CVPLLOOP              HAVE ANOTHER GO
WTO4      WTO   'WAIT4FTP- NOT ALL DATASETS HAVE ARRIVED - TERMINATING',X
                ROUTCDE=11
          MVC   RETC,F16                 SET RC=16
          B     RETURN
GOTEM     DS    ØH
```

```
         OPEN  (DSNAMES,OUTPUT)          OPEN OUTPUT DCB
         LA    R3,DSCBDEF                POINT TO FIRST DSCB
DSNLOOP  DS    ØH
         MVC   OUTREC(44),Ø(R3)          MOVE DSNAME TO OUTPUT RECORD
         PUT   DSNAMES,OUTREC            WRITE IT OUT
         LA    R3,DSCBSIZE(R3)           BUMP TO NEXT ONE
         BCT   R2,DSNLOOP                GET THE REST
         CLOSE DSNAMES
RETURN   DS    ØH
         L     R15,RETC                  LOAD RETURN CODE
         PR    ,                         RESTORE CALLER DATA, RETURN
*********************************************************************
* DIDN'T FIND OUR VOLUME...                                         *
*********************************************************************
NOTFND   DS    ØH
         MVC   WTO5+26(6),VOLID          MOVE VOLID TO WTO
WTO5     WTO   'WAIT4FTP- VOLUME "XXXXXX" NOT LOCATED...',ROUTCDE=11
         MVC   RETC,F8
         B     RETURN
*********************************************************************
* INSUFFICIENT BUFFERS TO HOLD DSCBS...                             *
*********************************************************************
MOREBUFF DS    ØH
         WTO   'WAIT4FTP- TOO MANY DSCBS LOCATED FOR THE # OF BUFFERS DX
               EFINED...',ROUTCDE=11
         MVC   RETC,F12
         B     RETURN
*********************************************************************
* BAD RETURN CODE FROM 'UCBSCAN'...                                 *
*********************************************************************
BADCALL  DS    ØH
         LR    R1Ø,R15                   SAVE RETC
         WTO   'WAIT4FTP- BAD CALL TO "UCBSCAN"...',ROUTCDE=11
         LR    R15,R1Ø                   RELOAD RETC
         DS    F                         BANG
*********************************************************************
* NO/INVALID PARMS PASSED...                                        *
*********************************************************************
BADPARMS DS    ØH
         WTO   'WAIT4FTP- NO/INVALID PARMS PASSED...',ROUTCDE=11
         ABEND 111,DUMP
*********************************************************************
* BAD RETURN CODE FROM 'UCBLOOK'...                                 *
*********************************************************************
LOOKERR  DS    ØH
         LR    R1Ø,R15                   SAVE RETC
         ST    RØ,SAVERØ                 SAVE REASON
         WTO   'WAIT4FTP- BAD CALL TO "UCBLOOK"...',ROUTCDE=11
         L     RØ,SAVERØ                 RELOAD REASON
         LR    R15,R1Ø                   RELOAD RETC
```

```
         ABEND 123                      BANG
*************************************************************************
* ERROR IN 'CVAFFILT' MACRO...                                          *
*************************************************************************
CVAFERR  DS    ØH
         LA    R1,CVPL+4                TEMP ADDRESSABILITY TO CVAF
         USING CVPLMAP,R1
         STH   R15,HWORD                SAVE RETC
         BAL   R9,CONVR15               CONVERT RETC TO PRINTABLE HEX
         MVC   CVAFR15(2),UNPKFLD       MOVE RETC TO DETAIL LINE
         MVI   HWORD,X'ØØ'              ZEROISE 1ST BYTE
         MVC   HWORD+1(1),CVSTAT        SAVE CVSTAT STATUS
         BAL   R9,CONVR15               CONVERT STAT TO PRINTABLE HEX
         MVC   CVAFSTAT(2),UNPKFLD      MOVE STAT TO DETAIL LINE
         MVC   WTO6+18(58),CVAFMSG      SHOW CVAFFILT ERROR
WTO6     WTO   'WAIT4FTP- XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
               XXXXXXXXXXXXX',ROUTCDE=11
         ABEND 555                      BANG...
         DROP  R1
*************************************************************************
* SUBROUTINE: CONVERT RETURN CODES, ETC INTO PRINTABLE HEX...           *
*************************************************************************
CONVR15  DS    ØH
         UNPK  UNPKFLD(3),HWORD+1(2)    UNPACK RETC + 1 BYTE
         TR    UNPKFLD(2),TRTAB-24Ø     XLATE TO PRINTABLE HEX
         BR    R9                       RETURN FROM ROUTINE
         EJECT
*-----------------------------------------------------------------------*
*
         LTORG                          LITERAL POOL
*
DSNAME   DC    C'FTPXFER.DATA.FILE.PFTP.DNNNNTY.**'
DSNSER   EQU   DSNAME+24,4
DSNAMELN DC    AL1(*-DSNAME)
VOLID    DC    CL6'??????'
UNPKFLD  DS    XL5
TRTAB    DC    C'Ø123456789ABCDEF'
HWORD    DC    H'Ø'
DWORD    DC    D'Ø'
F4       DC    F'4'
F8       DC    F'8'
F12      DC    F'12'
F16      DC    F'16'
RETC     DC    F'Ø'
UCBADDR  DC    F'Ø'
SAVER15  DC    F'Ø'
SAVERØ   DC    F'Ø'
SWITCH   DC    C'N'
BIN6Ø    DC    F'1ØØØ'                  6Ø SEC INTERVAL
CVAFMSG  DS    ØCL58
```

```
         DC    CL35'ERROR IN "CVAFFILT" MACRO - RC = X"'
CVAFR15  DC    CL2' '
         DC    CL13'" CVSTAT = X"'
CVAFSTAT DC    CL2' '
         DC    CL6'"'
*-------------------------------------------------------------------*
* UCB SCAN ROUTINE PARAMETERS...                                    *
*-------------------------------------------------------------------*
         DS    ØF
UCBAREA  DS    XL48                    HOLDS UCB COMMON & DEV SEGS
UCBWORK  DS    XL1ØØ                   UCBSCAN WORKAREA
*-------------------------------------------------------------------*
* DCB FOR OUTPUT...                                                 *
*-------------------------------------------------------------------*
DSNAMES  DCB   DDNAME=DSNAMES,                                      X
               LRECL=8Ø,                                            X
               RECFM=FB,                                            X
               DSORG=PS,                                            X
               MACRF=PM
OUTREC   DC    CL8Ø' '
*-------------------------------------------------------------------*
* SPACE ALLOCATION FOR CVPL, FCL, BFL, AND DSCB BUFFERS...          *
*-------------------------------------------------------------------*
FCLDEF   DS    (FCLHDLEN+FCLDSNEL)X   FCL HEADER AND 1 FCL ELEMENT
FCLSIZE  EQU   *-FCLDEF
*========DEFINE A CVAF BUFFER LIST WITH 'N' BUFFER LIST ELEMENTS=======
BFLHDEF  DS    (BFLHLN)X          BUFFER LIST HEADER (BFLH)
BFLEDEF  DS    (BUFFNUM*BFLELN)X  'N' BUFFER LIST ELEMENTS (BFLE'S)
BFLSIZE  EQU   *-BFLHDEF
*========DEFINE 'N' FULL DSCB BUFFERS================================
DSCBDEF  DS    (BUFFNUM*DSCBSIZE)X
*-------------------------------------------------------------------*
* REGISTERS EQUATES, ETC...                                         *
*-------------------------------------------------------------------*
BUFFNUM  EQU    3                  3 BUFFER LIST ELEMENTS AND BUFFERS
*
         YREGS
*
CVPLMAP  ICVAFPL CVPLFSA=YES
FCLMAP   ICVFCL
BFLMAP   ICVAFBFL
         IEFUCBOB ,
         PRINT NOGEN
         CVT   DSECT=YES
DSCBMAP  DSECT
         IECSDSL1 (1)             FORMAT1 DSCB MAPPING TO BEF BUFFSIZE
DSCBSIZE EQU   *-IECSDSL1
*
         END                     , END OF PROGRAM
```

SAMPLE JCL TO RUN WAIT4FTP

```
//JOBCARD
//S1       EXEC PGM=WAIT4FTP,PARM='FTPØØ1,1312' <-- VOLID, TIMESTAMP
//STEPLIB   DD DISP=SHR,DSN=TEST.LINKLIB
//SYSPRINT  DD SYSOUT=*
//DSNAMES   DD DSN=&&DSNAMES,DISP=(MOD,PASS),UNIT=SYSDA,SPACE=(TRK,1)
```

Grant Carson
Systems Programmer (UK)

Data conversion

This utility takes a COBOL copybook and converts it into a REXX include file. If you compile REXX programs, the REXX compiler parses the program looking for special INCLUDE instructions and then copies that code into the source before finishing the compile – much like a pre-processor.

The utility was written to perform data conversion. A project leader in the group decided that we REXX users had to use the same names for our variables as the COBOL people. The COBOL people were getting their copybooks from a data-mapping group, but that left us REXXers out in the cold. Initially, some of us started to manually convert COBOL copybooks to a REXX-like format, but with frequent changes in the data-mapping. Well, you can see that it had problems.

So I wrote the attached utility. With it we did the majority of the conversion using REXX rather than COBOL (much to the chagrin of the aforementioned project leader), and we came in a full five months ahead of schedule.

I have recently found another use for it, and once again it's a lifesaver.

COPYBOOK REXX

```
/* --------------------------- Rexx  -------------------------------
    - Title:  Convert COBOL Copy Books To REXX Include Format          -
    - ================================================================-
      - Program Function:                                             -
      -                                                               -
      -  Converts a COBOL copybook to something REXX can work with.   -
      -                                                               -
      - ------------------------------------------------------------ -
      -      Feel free to modify this program as you see fit.         -
      - ------------------------------------------------------------ -
      -                                                               -
      - Syntax:                                                       -
      -                                                               -
      -          COPYBOOK MEMBER SOURCEPDS DESTPDS                    -
      -                                                               -
      -  Where:                                                       -
      -          COPYBOOK  ->  This program.                          -
      -          MEMBER    ->  Name of COBOL copybook member to be    -
      -                        converted to REXX.                     -
      -          SOURCEPDS ->  PDS containing COBOL copybook members. -
      -          DESTPDS   ->  PDS that will contain the converted    -
      -                        copybook in REXX format.               -
      -                                                               -
      ---------------------------------------------------------------*/
ConvertCobolCopybooks:
      arg member source dest .
      call Initialize
      call PerformCopyBookConversion
      call Finalize
return
/* ---------------------------------------------------------------*/
Initialize:
      k = Ø; offset = 1; true = 1; false = Ø ; fieldlen = Ø
      level.Ø = ''; variable.Ø = ''; offset.Ø = ''; stackptr = Ø
/* Initialize stacks */
      levelq.Ø = ''; variableq.Ø = ''; offsetq.Ø = ''; queuetop = Ø
/* Initializq queues */
      lastlevel = Ø
/* Just what it says */
      i = Ø
return
/* ---------------------------------------------------------------*/
PerformCopyBookConversion:
      call ReadCopybook
      do while (i < copybook.Ø)
         i = i + 1
         copybook = PreProcessCopybook(copybook.i)
```

```
         parse var copybook level variable pic field remainder
         if ((level < lastlevel) | (pic = '' & level = LevelStackTop()))
then
              do while (level <= LevelStackTop())
                 call PopStacks
              end
         lastlevel = level
         select
            when (left(copybook,1) = "*") then
                nop
            when (pic = 'REDEFINES') then
                do
                   offset = RedefinesProcessing(field)
                   parse var copybook level variable redword redvar
pic field remainder
                   if (field ¬= '') then
                      call ProcessCobolLevel
                   else
                      call PushStacks
                end
            when level = 88 then
                call Level88Processing
            when pic = '' then
                call PushStacks
            otherwise
                call ProcessCobolLevel
         end
      end
      do while (stackptr ¬= Ø)
         call PopStacks
      end
return
/* ---------------------------------------------------------------*/
PreProcessCopybook:
         arg copybook
         if (left(copybook,1) ¬= " ") | (left(copybook,1) ¬= "*") then
            parse var copybook 1 junk 7 copybook 73 .
         copybook = strip(copybook)
         select
            when (copybook = '') then
               copybook = '*'
            when (left(copybook,1) ¬= '*') then
               copybook = BuildCopybookLine(copybook)
            otherwise nop
         end
return copybook;
/* ---------------------------------------------------------------*/
BuildCopyBookLine:
      arg line
```

```
      line = strip(line)
      do while ((substr(line,length(line),1) ¬= '.') & (left(line,1)
¬= "*") & (i < copybook.Ø))
         i = i+1; line =line' 'strip(copybook.i)
      end
      line = substr(line,1,length(line)-1) /* gets rid of period */
return line
/* ------------------------------------------------------------------*/
LevelStackTop:
return level.stackptr
/* ------------------------------------------------------------------*/
VariableStackTop:
return variable.stackptr
/* ------------------------------------------------------------------*/
PushStacks:
     stackptr = stackptr + 1
     variable.stackptr = variable
     level.stackptr = level
     offset.stackptr = offset
return
/* ------------------------------------------------------------------*/
PopStacks:
     newoffset = offset.stackptr
     newlevel = level.stackptr
     newvariable = translate(variable.stackptr,'_','-')
     stackptr = stackptr - 1
     k = k + 1
     outrec.k = "  parse var record" left(newoffset,6)
left(newvariable, 4Ø) offset
     x = Enqueue(newlevel,newvariable,newoffset)
return
/* ------------------------------------------------------------------*/
Level88Processing:
     variable = translate(variable,'_','-')
     field = TranslateField(field)
     k = k + 1
     outrec.k = "     /* 88-Level */ "||variable||" =
("||lastVariable||"="||field||")";
     if (strip(remainder) ¬= "") then
       do
          string = (pos("'",remainder)¬=Ø);
          if (string) then
             do while (remainder ¬= "")
                parse var remainder "'"val"'" remainder
                outrec.k = outrec.k||" |
("||lastVariable||"='"||val||"')";
             end
          else
             do while (remainder ¬= "")
```

```
                parse var remainder val remainder
                outrec.k = outrec.k||" |
("||lastVariable||"="||val||")";
             end
       end
return
/* ------------------------------------------------------------------*/
TranslateField: procedure expose fieldlen
     arg field
     select
        when (field = 'SPACES') | (field = 'SPACE') then
             field = "'"copies(" ",fieldlen)"'"
        when (field = 'ZERO') | (field = 'ZEROS') then
            field = Ø
        when (field = 'LOW-VALUES') then
            field = copies('ØØ'x,fieldlen)
        when (field = 'HIGH-VALUES') then
            field = copies('FF'x,fieldlen)
        otherwise nop
     end
return field
/* ------------------------------------------------------------------*/
ProcessCobolLevel:
      value = ''
      computational = (pos("COMP-3",remainder) ¬= Ø)
      binary = (pos("COMP ",remainder) ¬= Ø)
      if pos("VALUE",remainder) ¬= Ø then
         parse var remainder "VALUE" value
      fieldlen = DetermineFieldLength(field)
      newoffset = fieldlen + offset
      k = k + 1
      variable = translate(variable,'_','-')
      lastvariable = variable   /* storage bucket in case of 88 levels*/
      outrec.k = "  parse var record" left(offset,6) left(variable,4Ø)
newoffset
      x = Enqueue(level,variable,offset)
      offset = newoffset
      if value ¬= '' then
         do
            k = k + 1
            outrec.k = variable" = "translateField(value);
         end
return
/* ------------------------------------------------------------------*/
RedefinesProcessing: procedure expose variableq. offsetq. queuetop
     arg redefinesvariable
     queueptr = 1
     redefinesvariable = translate(redefinesvariable,'_','-')
     do while redefinesvariable ¬= variableq.queueptr & queueptr <
```

```
queuetop
        queueptr = queueptr + 1
     end
return offsetq.queueptr
/* ------------------------------------------------------------------*/
Enqueue: procedure expose levelq. variableq. offsetq. queuetop
     parse arg level,variable,offset
     queuetop = queuetop + 1
     levelq.queuetop = level
     variableq.queuetop = variable
     offsetq.queuetop = offset
return rc
/* ------------------------------------------------------------------*/
DetermineFieldLength: procedure expose computational binary
      arg picmap
      parse var picmap decimal'V'fraction
      fieldlen = 0
      if decimal ¬= '' then
         do
             parse var decimal pre'('len')'
             if len = '' then
                do
                   if substr(pre,1,1) = 'S' then
                      len = length(pre) - 1
                   else
                      len = length(pre)
                end
             fieldlen = fieldlen + len
         end
      if fraction ¬= '' then
         do
             parse var fraction pre'('len')'
             if len = '' then
                len = length(pre)
             fieldlen = fieldlen + len
         end
      select
         when computational then
            fieldlen = trunc((fieldlen + 1)/2)
         when binary then
            select
               when fieldlen <= 4 then fieldlen = 2
               when fieldlen <= 9 then fieldlen = 4
               when fieldlen <= 16 then fieldlen = 6
               otherwise fieldlen = 8
            end
         otherwise nop
      end
return fieldlen
/* ------------------------------------------------------------------*/
```

```
ReadCopybook:
      source = source'('member')';
      if sysdsn("'"source"'") ¬= "OK" then
         exit 8
      else
         x = ReadDataset(source,"COPYBOOK")
return
/* ---------------------------------------------------------------*/
ReadDataset:
      parse arg file,stemvar .
      address tso "alloc dd(INDD) da('"file"') shr"
      "execio * diskr INDD (stem "stemvar". finis"
      address tso "free dd(INDD)"
return rc
/* ---------------------------------------------------------------*/
Finalize:
     convertedcopybook = dest"("member")"
     if sysdsn("'"dest"'") ¬= 'OK' then
         do
       address tso "alloc dd(TEMPDD) da('"dest"') new reuse tr dir(2Ø)",
                 "sp(3Ø 3Ø) lrecl(255) recfm(V B) dsorg(PO) blksize(Ø)"
            address tso "free dd(TEMPDD)"
         end
     address tso "alloc dd(OUTDD) da('"convertedcopybook"') shr"
     "execio "k" diskw OUTDD (stem outrec. finis"
     address tso "free dd(OUTDD)"
return
/* -=============================================================-*/
```

SAMPLE COBOL COPYBOOK

This is an example of a COBOL copybook:

```
*****************************************************************
*  CABTØØ25 - CUI JURISDICTION TABLE RECORD *
*               GTE HISTORY SECTION                    TCID *
*   DATE      INTIAL     RVL      DESCRIPTION          DPSR *
*  Ø7/Ø1/87   KAN      R1VØ1LØ1   CABS ENHANCEMENT *
*  12/28/88   SAC      R2VØ1LØ1   DATE SENSITIVITY   CABDØØ7Ø *
*  12/28/88   SAC      R2VØ1LØ1   FGB MPB INDICATOR  CABDØ119 *
*  Ø6/Ø5/9Ø   REW      R2VØ1LØ1   AT&T Ø EMR11Ø1 CABDØ273 *
*  Ø4/28/91   SAC      R2VØ6LØ1   DB8ØØ INDICATORS   CABDØ3Ø6 *
*  Ø4/28/91   SAC      R2VØ6LØ1   CELL NATL THRESHLD CABDØ3Ø7 *
*  Ø3/12/93   LB       R1V24LØ1   ADD VALUE OF 'L'   CBSDØ838 *
*                                 TO TRAFFIC-TYPE- *
*                                 BILLABLE-INDICATOR *
*****************************************************************
   Ø3  TØØ25-TABLE-IDENT              PIC X(Ø5).
```

```
    88  T0025-TABLE-ID              VALUE 'T0025'.
03  T0025-TABLE-DATA.
  05  T0025-KEY.
      10  T0025-PARTIAL-KEY.
          15  T0025-FROM-NPA-NXX.
              20 T0025-FROM-NPA         PIC X(03).
              20 T0025-FROM-NXX         PIC X(03).
          15  T0025-TO-NPA              PIC X(03).
      10  T0025-EFFECTIVE-START         PIC 9(6).
      10  T0025-EFFECTIVE-END           PIC 9(6).
  05  T0025-FUNCTION.
      10  T0025-TABLE-LEVEL         PIC X(1).
      10  T0025-JURISDICTION        PIC 9.
      10  T0025-STATE-CODE          PIC X(02).
      10  T0025-BILLING-LOCATION-CODE. 00002400
          15  T0025-OPERATING-GROUP PIC X(01).
          15  T0025-CONSOL-COMPANY  PIC X(01).
          15  T0025-COMPANY         PIC X(01).
          15  T0025-AREA-DIV        PIC X(01).
          15  T0025-DIV-DIST        PIC X(02).
          15  T0025-PLANT-CODE      PIC X(04).
      10  T0025-BAN-STATE-CODE      PIC X(01).
      10  T0025-ORIG-MMU            PIC X(01).
      10  T0025-ORIG-RATE-CNTR      PIC 9(03).
      10  T0025-OLATA-CODE          PIC 9(03).
      10  T0025-BILLING-RAO         PIC X(03).
      10  T0025-ASSOC-BELL-RAO      PIC X(03).
      10  T0025-RCC-THRESHOLD-SEC   PIC 9(02).
      10  T0025-IND-EQUAL-ACCESS    PIC X(01).
          88  T0025-EQUAL-ACCESS        VALUE 'Y'.
          88  T0025-NON-EQUAL-ACCESS    VALUE 'N'.
      10  T0025-FGB-MPB-IND         PIC X(01).
      10  T0025-FGCD-MPB-IND        PIC X(01).
      10  T0025-CONV-OPH-INTRA      PIC S9V9(02) COMP-3.
      10  T0025-CONV-OPH-INTER      PIC S9V9(02) COMP-3.
      10  T0025-CONV-DIAL           PIC S9V9(02) COMP-3.
      10  T0025-RECORD-POINT-DDD    PIC S9(06)  COMP-3.
      10  T0025-RECORD-POINT-OPH    PIC S9(06)  COMP-3.
      10  T0025-RECORD-POINT-TERM   PIC S9(06)  COMP-3.
      10  T0025-ATT-SOURCE-OF-DATA.
          15  T0025-ATT-OPH-ZERO-PLUS      PIC X(01).
              88  T0025-ATT-TYPE-1-TOLL      VALUE 'T' 'L' 'C'
              88  T0025-ATT-TYPE-1-UMS       VALUE 'U'.
              88  T0025-ATT-TYPE-1-EMR       VALUE 'D'.
          15  T0025-ATT-OPH-ZERO-MINUS     PIC X(01).
              88  T0025-ATT-TYPE-2-TOLL      VALUE 'T' 'L' 'C'
              88  T0025-ATT-TYPE-2-UMS       VALUE 'U'.
              88  T0025-ATT-TYPE-2-EMR       VALUE 'D'.
          15  T0025-ATT-MTS-ORIG           PIC X(01).
              88  T0025-ATT-TYPE-3-TOLL      VALUE 'T' 'L'.
```

```
        88  T0025-ATT-TYPE-3-UMS      VALUE 'U'.
    15  T0025-ATT-700-ORIG          PIC X(01).
        88  T0025-ATT-TYPE-4-TOLL     VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-4-UMS      VALUE 'U'.
    15  T0025-ATT-800-ORIG          PIC X(01).
        88  T0025-ATT-TYPE-5-TOLL     VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-5-UMS      VALUE 'U'.
    15  T0025-ATT-DB800-ORIG        PIC X(01).
        88  T0025-ATT-DB800-TOLL      VALUE 'T' 'L'.
        88  T0025-ATT-DB800-UMS       VALUE 'U'.
        88  T0025-ATT-DB800-EMR       VALUE 'E'.
    15  T0025-ATT-900-ORIG          PIC X(01).
        88  T0025-ATT-TYPE-6-TOLL     VALUE 'T' 'L'.
        88  T0025-ATT-TYPE-6-UMS      VALUE 'U'.
    15  T0025-ATT-MTS-TERM          PIC X(01).
        88  T0025-ATT-TYPE-7-UMS      VALUE 'U'.
10  T0025-NONATT-SOURCE-OF-DATA.
    15  T0025-NONATT-OPH-ZERO-PLUS  PIC X(01).
        88  T0025-NONATT-TYPE-1-TOLL  VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-1-UMS   VALUE 'U'.
    15  T0025-NONATT-OPH-ZERO-MINUS PIC X(01).
        88  T0025-NONATT-TYPE-2-TOLL  VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-2-UMS   VALUE 'U'.
    15  T0025-NONATT-MTS-ORIG       PIC X(01).
        88  T0025-NONATT-TYPE-3-UMS   VALUE 'U'.
    15  T0025-NONATT-700-ORIG       PIC X(01).
        88  T0025-NONATT-TYPE-4-UMS   VALUE 'U'.
    15  T0025-NONATT-800-ORIG       PIC X(01).
        88  T0025-NONATT-TYPE-5-TOLL  VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-5-UMS   VALUE 'U'.
    15  T0025-NONATT-DB800-ORIG     PIC X(01).
        88  T0025-NONATT-DB800-TOLL   VALUE 'T' 'L'.
        88  T0025-NONATT-DB800-UMS    VALUE 'U'.
        88  T0025-NONATT-DB800-EMR    VALUE 'E'.
    15  T0025-NONATT-900-ORIG       PIC X(01).
        88  T0025-NONATT-TYPE-6-TOLL  VALUE 'T' 'L'.
        88  T0025-NONATT-TYPE-6-UMS   VALUE 'U'.
    15  T0025-NONATT-MTS-TERM       PIC X(01).
        88  T0025-NONATT-TYPE-7-UMS   VALUE 'U'.
10  T0025-CIC000-DB800-ORIG       PIC X(01).
    88  T0025-CIC000-DB800-TOLL   VALUE 'T' 'L'.
    88  T0025-CIC000-DB800-UMS    VALUE 'U'.
    88  T0025-CIC000-DB800-EMR    VALUE 'E'.
```

SAMPLE COPYBOOK REXX

Here is the same sample after it has been run through the utility and converted:

```
   parse var record 1      T0025_TABLE_IDENT                          6
      /* 88-Level */ T0025_TABLE_ID = (T0025_TABLE_IDENT='T0025')
   parse var record 6      T0025_FROM_NPA                             9
   parse var record 9      T0025_FROM_NXX                             12
   parse var record 6      T0025_FROM_NPA_NXX                         12
   parse var record 12     T0025_TO_NPA                               15
   parse var record 6      T0025_PARTIAL_KEY                          15
   parse var record 15     T0025_EFFECTIVE_START                      21
   parse var record 21     T0025_EFFECTIVE_END                        27
   parse var record 6      T0025_KEY                                  27
   parse var record 27     T0025_TABLE_LEVEL                          28
   parse var record 28     T0025_JURISDICTION                         29
   parse var record 29     T0025_STATE_CODE                           31
   parse var record 31     T0025_OPERATING_GROUP                      32
   parse var record 32     T0025_CONSOL_COMPANY                       33
   parse var record 33     T0025_COMPANY                              34
   parse var record 34     T0025_AREA_DIV                             35
   parse var record 35     T0025_DIV_DIST                             37
   parse var record 37     T0025_PLANT_CODE                           41
   parse var record 31     T0025_BILLING_LOCATION_CODE                41
   parse var record 41     T0025_BAN_STATE_CODE                       42
   parse var record 42     T0025_ORIG_MMU                             43
   parse var record 43     T0025_ORIG_RATE_CNTR                       46
   parse var record 46     T0025_OLATA_CODE                           49
   parse var record 49     T0025_BILLING_RAO                          52
   parse var record 52     T0025_ASSOC_BELL_RAO                       55
   parse var record 55     T0025_RCC_THRESHOLD_SEC                    57
   parse var record 57     T0025_IND_EQUAL_ACCESS                     58
      /* 88-Level */ T0025_EQUAL_ACCESS = (T0025_IND_EQUAL_ACCESS='Y')
      /* 88-Level */ T0025_NON_EQUAL_ACCESS =
(T0025_IND_EQUAL_ACCESS='N')
   parse var record 58     T0025_FGB_MPB_IND                          59
   parse var record 59     T0025_FGCD_MPB_IND                         60
   parse var record 60     T0025_CONV_OPH_INTRA                       62
   parse var record 62     T0025_CONV_OPH_INTER                       64
   parse var record 64     T0025_CONV_DIAL                            66
   parse var record 66     T0025_RECORD_POINT_DDD                     69
   parse var record 69     T0025_RECORD_POINT_OPH                     72
   parse var record 72     T0025_RECORD_POINT_TERM                    75
   parse var record 75     T0025_ATT_OPH_ZERO_PLUS                    76
      /* 88-Level */ T0025_ATT_TYPE_1_TOLL =
(T0025_ATT_OPH_ZERO_PLUS='T') | (T0025_ATT_OPH_ZERO_PLUS='L') |
(T0025_ATT_OPH_ZERO_PLUS='C')
      /* 88-Level */ T0025_ATT_TYPE_1_UMS =
(T0025_ATT_OPH_ZERO_PLUS='U')
      /* 88-Level */ T0025_ATT_TYPE_1_EMR =
(T0025_ATT_OPH_ZERO_PLUS='D')
   parse var record 76     T0025_ATT_OPH_ZERO_MINUS                   77
      /* 88-Level */ T0025_ATT_TYPE_2_TOLL =
(T0025_ATT_OPH_ZERO_MINUS='T') | (T0025_ATT_OPH_ZERO_MINUS='L') |
```

```
(T0025_ATT_OPH_ZERO_MINUS='C')
      /* 88-Level */ T0025_ATT_TYPE_2_UMS =
(T0025_ATT_OPH_ZERO_MINUS='U')
      /* 88-Level */ T0025_ATT_TYPE_2_EMR =
(T0025_ATT_OPH_ZERO_MINUS='D')
   parse var record 77     T0025_ATT_MTS_ORIG                      78
      /* 88-Level */ T0025_ATT_TYPE_3_TOLL = (T0025_ATT_MTS_ORIG='T')
| (T0025_ATT_MTS_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_3_UMS = (T0025_ATT_MTS_ORIG='U')
   parse var record 78     T0025_ATT_700_ORIG                      79
      /* 88-Level */ T0025_ATT_TYPE_4_TOLL = (T0025_ATT_700_ORIG='T')
| (T0025_ATT_700_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_4_UMS = (T0025_ATT_700_ORIG='U')
   parse var record 79     T0025_ATT_800_ORIG                      80
      /* 88-Level */ T0025_ATT_TYPE_5_TOLL = (T0025_ATT_800_ORIG='T')
| (T0025_ATT_800_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_5_UMS = (T0025_ATT_800_ORIG='U')
   parse var record 80     T0025_ATT_DB800_ORIG                    81
      /* 88-Level */ T0025_ATT_DB800_TOLL = (T0025_ATT_DB800_ORIG='T')
| (T0025_ATT_DB800_ORIG='L')
      /* 88-Level */ T0025_ATT_DB800_UMS = (T0025_ATT_DB800_ORIG='U')
      /* 88-Level */ T0025_ATT_DB800_EMR = (T0025_ATT_DB800_ORIG='E')
   parse var record 81     T0025_ATT_900_ORIG                      82
      /* 88-Level */ T0025_ATT_TYPE_6_TOLL = (T0025_ATT_900_ORIG='T')
| (T0025_ATT_900_ORIG='L')
      /* 88-Level */ T0025_ATT_TYPE_6_UMS = (T0025_ATT_900_ORIG='U')
   parse var record 82     T0025_ATT_MTS_TERM                      83
      /* 88-Level */ T0025_ATT_TYPE_7_UMS = (T0025_ATT_MTS_TERM='U')
   parse var record 75     T0025_ATT_SOURCE_OF_DATA                83
   parse var record 83     T0025_NONATT_OPH_ZERO_PLUS              84
      /* 88-Level */ T0025_NONATT_TYPE_1_TOLL =
(T0025_NONATT_OPH_ZERO_PLUS='T') | (T0025_NONATT_OPH_ZERO_PLUS='L')
      /* 88-Level */ T0025_NONATT_TYPE_1_UMS =
(T0025_NONATT_OPH_ZERO_PLUS='U')
   parse var record 84     T0025_NONATT_OPH_ZERO_MINUS             85
      /* 88-Level */ T0025_NONATT_TYPE_2_TOLL =
(T0025_NONATT_OPH_ZERO_MINUS='T') | (T0025_NONATT_OPH_ZERO_MINUS='L')
      /* 88-Level */ T0025_NONATT_TYPE_2_UMS =
(T0025_NONATT_OPH_ZERO_MINUS='U')
   parse var record 85     T0025_NONATT_MTS_ORIG                   86
      /* 88-Level */ T0025_NONATT_TYPE_3_UMS =
(T0025_NONATT_MTS_ORIG='U')
   parse var record 86     T0025_NONATT_700_ORIG                   87
      /* 88-Level */ T0025_NONATT_TYPE_4_UMS =
(T0025_NONATT_700_ORIG='U')
   parse var record 87     T0025_NONATT_800_ORIG                   88
      /* 88-Level */ T0025_NONATT_TYPE_5_TOLL =
(T0025_NONATT_800_ORIG='T') | (T0025_NONATT_800_ORIG='L')
      /* 88-Level */ T0025_NONATT_TYPE_5_UMS =
(T0025_NONATT_800_ORIG='U')
```

```
   parse var record 88     T0025_NONATT_DB800_ORIG                  89
      /* 88-Level */ T0025_NONATT_DB800_TOLL =
(T0025_NONATT_DB800_ORIG='T') | (T0025_NONATT_DB800_ORIG='L')
      /* 88-Level */ T0025_NONATT_DB800_UMS =
(T0025_NONATT_DB800_ORIG='U')
      /* 88-Level */ T0025_NONATT_DB800_EMR =
(T0025_NONATT_DB800_ORIG='E')
   parse var record 89     T0025_NONATT_900_ORIG                    90
      /* 88-Level */ T0025_NONATT_TYPE_6_TOLL =
(T0025_NONATT_900_ORIG='T') | (T0025_NONATT_900_ORIG='L')
      /* 88-Level */ T0025_NONATT_TYPE_6_UMS =
(T0025_NONATT_900_ORIG='U')
   parse var record 90     T0025_NONATT_MTS_TERM                    91
      /* 88-Level */ T0025_NONATT_TYPE_7_UMS =
(T0025_NONATT_MTS_TERM='U')
   parse var record 83     T0025_NONATT_SOURCE_OF_DATA              91
   parse var record 91     T0025_CIC000_DB800_ORIG                  92
      /* 88-Level */ T0025_CIC000_DB800_TOLL =
(T0025_CIC000_DB800_ORIG='T') | (T0025_CIC000_DB800_ORIG='L')
      /* 88-Level */ T0025_CIC000_DB800_UMS =
(T0025_CIC000_DB800_ORIG='U')
      /* 88-Level */ T0025_CIC000_DB800_EMR =
(T0025_CIC000_DB800_ORIG='E')
   parse var record 27     T0025_FUNCTION                           92
   parse var record 6      T0025_TABLE_DATA                         92
```

Rick Myers
Technical Trainer
Verizon Communications

Space abend reporter

INTRODUCTION

Since the dawn of computers, every now and then you get given the ‘expert’ advice: “If you prevent application abends, you will improve throughput”. No matter how easy it may sound, doing it for an entire installation is not such a trivial job. The fact is that exponential growth in data storage resulting from data-intensive applications, e-business, and regulatory requirements to retain data have driven the need for more efficient systems to manage

storage resources. It is also a fact that few businesses today can afford processing delays or application failures and reruns.

However, application failures and errors do occur. Although not explicitly responsible for failures in application programs, a storage administrator is the one who is called upon when failures occur, especially if there is even the slightest hint or suspicion that the error is related to storage management. On the other hand, the interpretation of cryptic and often hard to understand IBM operating system error codes and messages can be tedious and frustrating, especially for application programmers. Often the appropriate manuals are not readily available for the users to access. If the manuals are available, the error description and corrective procedure are often impossible to interpret – it is the storage administrator again who must interpret the error and find a cause, and provide the 'patch'.

Amongst a large variety of possible errors the most common type is the inability to acquire a sufficient amount of DASD space. This usually produces a JCL error if primary space cannot be obtained while an *x*37 type of error (B37, D37, or E37 abend) is generated if secondary space is unavailable. These abends are simply telling us that the job was not able to obtain enough DASD space to continue. The most frequent response to it is to increase the amount of space requested. Calculating how much would be enough is not an easy task, however, and space requirements are often guessed at – and generally guessed high, to avoid any chance of *x*37 abends. This can only waste resources and increase volume fragmentation without eliminating *x*37 abends: a job that requests much more space than it will really use may not only run short of space itself, but may cause errors when the space it holds is unavailable for other jobs. It is often interesting to look at file usage statistics on a system, and one may find some files with up to 90% of their allocated space free. Therefore increasing the amount of space requested may make the situation worse rather than better.

In order to prevent these out-of-space errors, some installations scrutinize applications' JCL parameters, which turns out to be

not only unreliable but also time-consuming and expensive in terms of human resources.

A more effective and efficient technique is to trap abends, and make available a variety of actions that would allow the job to complete. An important aspect of this automatic 'only-as-needed' approach is the assurance that overhead, fragmentation, and other side-effects that waste space are minimized. When a storage administrator develops a management structure and assigns constructs to DFSMS storage groups, the goal is to anticipate and avoid running out of allowable space and abending a job. Even when additional parameters are made available in DFSMS to handle space-related problems, the same issues remain: the complex implementation of global space recovery functions that are processed in an all-or-none fashion. High-use thresholds are established for each storage group in order to leave some percentage of space unused in the storage group to service an unusual space request, or some volumes/storage groups are placed in quiesced status, so that files are allocated only on these volumes/storage groups when a high threshold is exceeded.

Success in reducing the number of *x*37s is directly related to the number of volumes in the storage group – the more volumes each group has, the merrier. The use of dataclass is paramount too: a proper dataclass will ensure extra candidate volumes are allocated to the datasets, so even if the user does allocate insufficient space, DFSMS will prevent it from failing by allowing it to extend to additional volumes. However, if you abuse this multivolume concept, you will have too much waste space, because RLSE is not issued against multivolume datasets. Some users have found this to be a major stumbling block because of the JCL changes necessary to remove the traditional space parameters and to specify the proper dataclass.

Worth the attention of the storage administrator are the space constraint relief enhancements – they do two main things to address out-of-space abends:

- They relax the limit on the number of extents that can be used to satisfy allocations for new datasets or extends to a new volume.
- The amount of space requested can be reduced by a percentage if the initial attempt fails, thus increasing the likelihood of a successful allocation (and, of course, being made up for by subsequent secondary extents).

These are designed to reduce the likelihood of *x*37 abends, and experience so far shows that they work fairly well.

With z/OS 1.3, DFSMS gets even better at handling out-of-space (*x*37) abends:

- Extend storage groups – one can specify a second storage group name when defining a storage group. If a dataset wants to extend to another volume, but there is insufficient space in the primary group, the dataset can extend into this storage group. Primary allocations will not use this storage group.
- Overflow storage groups – similar to the above, except that they are used when DFSMS cannot initially allocate onto the primary storage group. This storage group becomes the dataset's primary storage group, ie second and other volumes must go into this storage group.

One can combine both features so that a storage group can be both an extend and overflow storage group. There is a lot of good information in the redbook *z/OS V1R3 DFSMS Technical Guide* (SG24-6569).

When it comes to dealing with out-of-space failures for VSAM datasets, things get a bit more complicated. The errors that occur when creating or extending VSAM datasets are believed to be the most cryptic in MVS, mainly consisting of just a return code and a reason code. Among these messages one of the most common is IDC3009I, issued when a catalog function fails. This particular message includes literally hundreds of possible return/reason codes, a few of which are encountered more frequently

than others. A return code 68 says that there is a space error in DADSM (no space is available on the user volume). When accompanied by reason code 20, 22, or 24, it tells us that the storage group did not contain sufficient space to allocate the dataset. More precisely, it means that an attempt to allocate a DFSMS-managed dataset, using the best-fit interface, allocated space on all supplied volumes, but the amount of space allocated was not the total amount required. The DADSM error codes are fully documented in the manual *DFPSMSdfp Diagnosis Reference* (LY27-9606).

COLLECTING X37 DATA

No matter how fine-tuned DFSMS might be, the *x*37 abends will not disappear, simply because DFSMS won't keep applications from abending – no such facility exists in DFSMS. Obviously, *x*37 errors can be reduced through the use of DFSMS facilities like extended format datasets or space constraint relief, extend storage groups, and/or overflow storage groups. These are good facilities that can greatly reduce the likelihood, but they don't stop a job from aborting when allocated space is insufficient.

In order to reduce the occurrence of *x*37 abends as well as to keep things under control a 'space abend' reporting program was written. Running it on a daily basis may help the storage administrator by identifying necessary pool size changes that can be made, and in discussion with the applications programmer on how the dataset got to be bigger than anticipated. This report could also be used to measure and predict how the storage policies impact the user community.

One of the best ways to report on space abends is through the SMF records. Before proceeding any further it might be helpful to understand the context in which certain SMF records, and the values within them, are written. In this particular case, one may be tempted to choose to process less useful SMF type 4 and type 30 records, and report the jobs that abended with B37, D37, and E37 abends.

When enabled by the proper SMFPRM*xx* TYPE parameter, SMF creates type 42, subtype 9 records, which are written each time an *x*37 failure occurs for non-VSAM files, while record type 64 (VSAM component/cluster status) was used to report out-of-space abends pertaining to a VSAM dataset. A detailed description of the layout of SMF type 42 and 64 records can be obtained from the *MVS System Management Facilities (SMF)*, SA22-7630-03 manual. One can also find the type 42 subtype descriptions in macro IGWSMF in SYS1.MACLIB. For type 64 one should consult macro IDASMF64 in the same library.

CODE

Based on record descriptions obtained from the manual, a sample report writer was written.

The code is a two-part stream. In the first part (COPYSMF) selected SMF records (selection being defined by INCLUDE conditions) are copied from the SMF dataset to a file, which can be used for archived records.

In the second part (ABD37), the captured records are formatted by invoking EXEC (AB37) and two reports are produced.

The X37 report shows information such as jobname, dataset name, volume serial number, number of dataset extents, DFSMS-related information, as well as date and time. The second report (under DDn V37) is for VSAM out-of space conditions and it provides information such as jobname, component name, number of extents, etc.

COPYSMF

```
//DEL       EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD *
    DELETE uid.ABEND.TEST
    DELETE uid.X377.DATA
    DELETE uid.V377.DATA
    SET MAXCC=Ø
```

```
//COPYSMF  EXEC PGM=ICETOOL
//TOOLMSG  DD SYSOUT=*
//DFSMSG   DD SYSOUT=*
//RAWSMF   DD DSN=your.smf.dataset,DISP=SHR
//SMFX37   DD DSN=uid.ABEND.TEST,
//         SPACE=(CYL,(1)),UNIT=SYSDA,
//         DISP=(NEW,CATLG,KEEP),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760)
//TOOLIN   DD *
  COPY FROM(RAWSMF) TO(SMFX37) USING(SMFI)
//SMFICNTL DD *
  OPTION SPANINC=RC4,VLSHRT
  INCLUDE COND=(6,1,BI,EQ,42,AND,23,2,BI,EQ,9,    * Get SMF type 42.9
         OR,(6,1,BI,EQ,64,AND,43,1,BI,EQ,X'20')) * also copy SMF 64
/*
//ABD37    EXEC PGM=IKJEFT01,REGION=0M,DYNAMNBR=50,PARM='%AB37'
//SYSEXEC  DD DISP=SHR,DSN=your.rexx.library
//SMF      DD DISP=SHR,DSN=uid.ABEND4.TEST
//X37      DD DSN=uid.X377.DATA,           <-- X37 report list
//         SPACE=(CYL,(1,1)),UNIT=SYSDA,
//         DISP=(NEW,KEEP),DCB=(RECFM=FB,LRECL=145)
//V37      DD DSN=uid.V377.DATA,           <-- VSAM report list
//         SPACE=(CYL,(1,1)),UNIT=SYSDA,
//         DISP=(NEW,KEEP),DCB=(RECFM=FB,LRECL=95)
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
/*
```

AB37 EXEC

```
/* REXX EXEC to read and format space abend records  */
/* trace ?r */
/*-----------------------------------------------------------------*/
/* Print Space Abend report header and labels                      */
/*-----------------------------------------------------------------*/
 Out.1 = left(' ',30,' '),
         ||center('Space Abend Report ',22,),
         ||left(' ',15,' ')
 Out.2 = left(' ',20,' '),
         ||center('Report produced on',18,),
         ||left(' ',1,' ')||left(date(),11),
         ||left(' ',1,' ')||left('at ',3,' '),
         ||left(time(),10)
 Out.3 = '  '
 Out.4 = left('Date',11)     left('Time',9),
         left('Job',6)       left('Step',4),
         left('X37',4)       left('Data Set',41),
```

```
         right('Disp',4)     right('Dsorg',6),
         left('Volume',7)    left('Ext.',4),
         left('Trks',4)      left('Stor.Cl.',12),
         left('Mngt.Cl.',10) left('Data Cl.',10)
 Out.5 = LEFT('-',145,'-')

 "EXECIO * DISKW X37 (STEM Out.)"
/*-------------------------------------------------------------*/
/* Print VSAM 'out-of space' header and labels                 */
/*-------------------------------------------------------------*/
 lin.1 = left(' ',30,' '),
         ||center('VSAM out-of-space Report ',30,),
         ||left(' ',15,' ')
 lin.2 = left(' ',20,' '),
         ||center('Report produced on',18,),
         ||left(' ',1,' ')||left(date(),11),
         ||left(' ',1,' ')||left('at ',3,' '),
         ||left(time(),10)
 lin.3 = '  '
 lin.4 = left('Date',11) left('Time',9),
         left('Job',8)   left('Component name',45),
         left('Trks',4)  right('Ext',4) left('Volume',6)
 lin.5 = LEFT('-',145,'-')

 "EXECIO * DISKW V37 (STEM Lin.)"

/*-------------------------------------------------------------*/
/*    Header for SMF records                                   */
/*-------------------------------------------------------------*/
 'EXECIO * DISKR SMF ( STEM x. FINIS'
    do i = 1 to x.0

  smftype  = c2d(SUBSTR(x.i,2,1))             /* SMF record type */
  smftime  = smf(c2d(SUBSTR(x.i,3,4)))        /* Decode SMF time */
  smfdate = SUBSTR(c2x(SUBSTR(x.i,7,4)),3,5)  /* Unpack SMF date */

  IF smftype = '64' Then call SMF64
  IF smftype = '42' Then
  Do
   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  */

/*-------------------------------------------------------------*/
/*    Product Section                                          */
/*-------------------------------------------------------------*/

   ops = c2d(SUBSTR(x.i,25,4))       /*Offset to product section */
   lps = c2d(SUBSTR(x.i,29,4))       /*Lenght to product section */
```

```
   nps = c2d(SUBSTR(x.i,31,4))       /*Number to product sections*/
 IF ops <> 0 AND lps <> 0 Then do
    ops=ops -3
    pdl = SUBSTR(x.i,ops,8)                    /*  Product level*/
    pdn = SUBSTR(x.i,ops+8,10)                 /*   Product name*/
    psv = c2d(SUBSTR(x.i,ops+18,1))   /*   Subtype version number*/
      SELECT
        when psv='0' then vhead ='No vol.header section'
        when psv='1' then vhead ='Vol. header exists'
      END
  end

/*-------------------------------------------------------------*/
/*   SMF42 subtype 9 header section (B37/D37/E37 abend)        */
/*-------------------------------------------------------------*/
   abo = c2d(SUBSTR(x.i,33,4))     /*Offset to X37 abend section */
   abl = c2d(SUBSTR(x.i,37,2))     /*Length of X37 abend section */
   abn = c2d(SUBSTR(x.i,39,2))     /*Number of X37 abend sections*/
   smo = c2d(SUBSTR(x.i,41,4))     /*Offset to SMS data section  */
   sml = c2d(SUBSTR(x.i,45,2))     /*Length of SMS data section  */
   smn = c2d(SUBSTR(x.i,47,2))     /*Number of SMS data sections */

/*-------------------------------------------------------------*/
/*   B37/D37/E37 Abend Data Section (part 1)                   */
/*-------------------------------------------------------------*/
 IF abo <> 0 AND abl <> 0 Then do
   abo=abo -3
   sys1 = SUBSTR(x.i,abo,4)                          /* System ID*/
   job  = SUBSTR(x.i,abo+4,8)                        /*  Job name*/
   time = smf(c2d(SUBSTR(x.i,abo+12,4)))             /*Start time*/
   date = SUBSTR(c2x(SUBSTR(x.i,abo+16,4)),3,5)      /*Start date*/
   uid  = SUBSTR(x.i,abo+20,8)                       /*  User id.*/
   step = c2d(SUBSTR(x.i,abo+28,1))                  /*  Step no.*/

     a37b = SUBSTR(x.i,abo+29,1)                     /* X37 flags*/
      SELECT
        when a37b='10000000'b then abend ='B37'
        when a37b='01000000'b then abend ='D37'
        when a37b='00100000'b then abend ='E37'
      END

     dsorg= c2x(SUBSTR(x.i,abo+34,2))      /*Dataset organization*/
      SELECT
        when dsorg='4000'  then Dorg='PS'
        when dsorg='2000'  then Dorg='DA'
        when dsorg='0200'  then Dorg='PO'
        otherwise dorg='PO'
      END
```

```
     disp = SUBSTR(x.i,abo+36,1)                /*   Dataset disp*/
      SELECT
        when disp ='11000001'b then dis ='Temp'
        when disp ='11000000'b then dis ='New'
        when disp ='10000000'b then dis ='Mod'
        when disp ='01000000'b then dis ='Old'
        when disp ='01001000'b then dis ='Shr'
      END

     dsn  = SUBSTR(x.i,abo+37,44)                /*  Dataset name*/
     vol  = SUBSTR(x.i,abo+81,6)         /*Volser of current volume*/
     ucb  = c2d(SUBSTR(x.i,abo+87,4))       /*UCB of current volume*/

     ext  = c2d(SUBSTR(x.i,abo+91,1)) /*No. of extents for dataset*/
                                           /*  on current volume*/

     trk  = c2d(SUBSTR(x.i,abo+92,4))  /*Total tracks allocated for*/
                                       /*dataset on current volume*/

     sal  = c2x(SUBSTR(x.i,abo+96,4)) /*Secondary allocation amount*/
     bl   = c2d(SUBSTR(x.i,abo+100,3))  /*Average data block length*/
                                        /*             if specified*/

/*-----------------------------------------------------------------*/
/*   SMS Data Secton                                               */
/*-----------------------------------------------------------------*/
      IF smo > 0  Then do
         smo=smo -3
         mclas  = SUBSTR(x.i,smo,10)       /* Management class name*/
         sclas  = SUBSTR(x.i,smo+30,12)    /*    Storage class name*/
         dclas  = SUBSTR(x.i,smo+60,10)    /*       Data class name*/
      end
         else do
         mclas  ='   '
         sclas  = 'Non SMS file'
         dclas  = '   '
      end
   end

  x37out = right(Date('N',date,'J'),11) left(smftime,9),
           left(job,8)  right(step,2)   left(abend,4),
           left(dsn,39) right(dis,6)    center(dorg,6),
           left(vol,8)  right(ext,3)    right(trk,4),
           right(sclas,12)  right(mclas,10)  right(dclas,10)

   PUSH x37out
     "EXECIO 1 DISKW X37"
```

```
   end
  end
/*-----------------------------------------------------------------*/
/* Print abend legend                                              */
/*-----------------------------------------------------------------*/
 Leg.1 = LEFT('-',145,'-')
 Leg.2 = '  '
 Leg.3 = left('X37 Abend legend:',20)
 Leg.4 = '  '
 Leg.5 = left('B37: All 16 extents used or',60)
 Leg.6 = left(' ',5,' '),
         ||left('No more space was available on the volume or',80)
 Leg.7 = left(' ',4,' '),
         left('Unable to upadate the VTOC (full)',78)
 Leg.8 = left('D37: Used all the primary space,',33),
         ||left('and no secondary space was requested',40)
 Leg.9  = left('E37: Used all space available on',32),
         ||left(' ',1,' '),
         ||left('the current volume,',31)
 Leg.10= left(' ',5,' '),
         ||left('and no more volumes were available',40)
 Leg.11 = '  '
 Leg.12 = left('For details see the following messages',70)
 Leg.13 = left('B37: IEC030I; D37: IEC031I; E37: IEC032I',80)
 "EXECIO * DISKW X37 (STEM Leg.)"
exit

/*-----------------------------------------------------------------*/
/*   Selected SMF64 variables                                      */
/*-----------------------------------------------------------------*/
SMF64:
   jbb = SUBSTR(x.i,15,8)                              /*  Job name*/
   rin = SUBSTR(x.i,39,1)                    /*Recording indicators*/
     SELECT
       when rin='10000000'b then sit='Component closed';
       when rin='01000000'b then sit='Vol switched';
       when rin='00100000'b then sit='No space avail';
       when rin='00010000'b then sit='Cat or CRA rec';
       when rin='00001000'b then sit='Closed type=t ';
       when rin='00000100'b then sit='Abend process ';
       when rin='00000010'b then sit='VVDS or ICF   ';
       otherwise  sit='Reserved'
     END
   dnm  =    SUBSTR(x.i,85,44)                     /*  Dataset name*/
   ntr  =    c2d(SUBSTR(x.i,129,2))     /*Number of tracks required*/
   esl  =    c2d(SUBSTR(x.i,135,2))         /*Extent segment length*/

/*-----------------------------------------------------------------*/
/* Extent table - one per volume online at time record written    */
```

```
/*-------------------------------------------------------------*/
   noext=esl/26                              /*No. of extent segment*/
     do extno=Ø to noext - 1
        incr = (14Ø + (extno*26)) - 3
          vol.extno = SUBSTR(x.i,incr+8,6)  /*Volser of curr.volume*/
          volume    = SUBSTR(x.i,incr+8,6)  /*Volser of last volume*/
          cuu =c2d(substr(x.i,incr+14,2))         /* device number*/
          ind =substr(x.i,incr+16,2)              /*   Spindle id.*/
          uty =c2d(substr(x.i,incr+18,4))    /* Unit type - ucbtype*/
     end

  v37out = right(Date('N',smfdate,'J'),11) left(smftime,9),
           left(jbb,8)  left(dnm,44),
           right(ntr,4) right(noext,5) left(volume,6)

   PUSH v37out
     "EXECIO 1 DISKW V37"
  return

SMF: procedure
/* REXX - convert a 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 time*/
    return otime
```

Mile Pekic
Systems Programmer (Serbia and Montenegro)

Extending the life of your mainframe with in-memory table management

In a perfect world, mainframe computing is all about blazing performance, unparalleled scalability, and 99.999% availability. In reality, today's influx of process/transaction-intensive applications, such as CRM, knowledge management, data mining, Internet-based applications, etc, are taxing mainframe systems. Furthermore, because of budget cutbacks, most companies today are either reluctant or unable to spend extra money on mainframe hardware upgrades that are required to support the increased processing that these large and complex systems necessitate. For many, the answer to this dichotomy is found by making effective use of tables in software applications, coupled with the implementation of an in-memory table management system. For over 20 years, in-memory table management technology has helped Fortune 1000 companies to improve significantly the speed and performance of their mainframe applications, maximize transaction volumes, and support more end-users, without consuming additional hardware resources.

Most of you are aware that using tables in application development means easier and less costly application maintenance because table data, such as a product price list for example, is easily updated without incurring the costs of re-testing the application – and the risk of introducing new errors is virtually eliminated.

Although tables can appear to be a programmer's best friend, often their processing efficiency is compromised because of the I/O required, especially if the tables are frequently-accessed.

By using an in-memory table management system that loads tables into memory and allows these tables to be shared among applications and regions, application performance is significantly increased while I/O and CPU usage is significantly decreased. This is but one advantage of using an in-memory data management system. There are many others.

An in-memory data management system assures optimal use of existing hardware, and can even go so far as to support the postponement of costly hardware upgrades by making the applications more efficient – the more efficient the application, the less CPU usage is required. In addition, a table management system manages the tables and the memory, unburdening the development and maintenance staff to focus on other tasks.

For those unfamiliar with a table management system, it is similar to a database management system (DBMS), except that tables are loaded into memory. Since data manipulation is performed directly in memory, data access speed is greatly enhanced and the extensive I/O and CPU overhead of disk is avoided. The result is a radical increase in the performance of mainframe applications while extending the useful life of the mainframe system.

Other data management schemes do not provide holistic access to large volumes of structured data. They typically provide only sequential access to small amounts of data – normally thought of as a record, such as customer records from a CRM database or an employee record from an HR application. However, this type of record-level access is not optimal *if*:

- The data contained in the record needs to be accessed more than once – and possibly repeatedly – during an application run, which means 24x7 for several consecutive weeks for online applications.
- The data needs to be accessed by multiple mainframe applications at the same time, possibly from different regions.
- The data is transient, and only exists for a short period of time, like stock-price quote data.

Data falling into the above categories is handled effectively using a table-driven application design approach coupled with an in-memory table management system.

LOOKING OVER TABLES

Most people accept that a table is an iterative data structure consisting of a series of homogeneous data holders known as rows. Most also realize that a table is a logical structure, independent of its physical representation in memory or on a disk drive.

In the case of programming and database scenarios, the contents of tables can be thought of as consisting of two parts: keys (or indexes) and the data accessed using those keys. When dealing with most catalogue type applications, for example, a part number is used as the key required to access the detailed product description. In many cases, the key occurs within a specific column within the table. See column 1 in the mail-order catalogue example below:

Item #	*Product*	*Price*	*Availability*	*Link*
AP1183	Adobe Photoshop 6.5	549.00	Y	www.adobe.com …
AP1266	Adobe Acrobat Full 5.4	499.00	Y	www.adobe.com …
AP1270	Adobe InDesign 2.0	699.00	N	www.adobe.com …
AP4881	AutoCAD LT 2002	619.95	N	www.autodesk.com …
AP4895	AutoDesk Symbol 2000	94.95	Y	www.autodesk.com …

The single-key-in-one-column approach may not be flexible or powerful enough in more complex situations. The application may want to use *multiple* keys – to search for items that are below $500 *and* are in stock for example – or even use a retrieval mechanism that is not key-centric. In some instances, various applications – and even modules within them – may want to access the data using dissimilar criteria.

Tables, given their inherent logical nature, may be implemented in a number of different ways. Some popular approaches for implementing tables are:

- As structured, and possibly even indexed, files, like VSAM on disk.
- With a relational database using a database management system (DBMS), such as DB2 or Oracle.

- As a programming language array within a specific application, just for use by that application.
- As application-independent data structures resident in memory, external to any specific application programs, that are created, accessed, and maintained using a table management system.

Using memory-based data structures external to applications, which is true in-memory table management, is the only approach that delivers blazing performance, and simplified addition of new data extraction, data manipulation, and data correlation functions to existing applications.

COMBINING DBMSS AND IN-MEMORY TABLE MANAGEMENT

DBMSs and almost all other data-access technologies are designed to work with files, file structures or databases. They operate on the premise that, any time an update is performed, the DBMS will journal that update, reconcile all necessary changes and then make the updated record available to the next user as quickly as possible. Most enterprise data is processed this way. However, this approach is extremely I/O intensive and saps up valuable CPU cycles.

With in-memory table management, these same data tables can be loaded into memory and be readily available. This permits access to that data much faster than DBMS, and ensures that all requisite data manipulations can be performed directly within memory. For heavily-accessed data, application run-times and on-line responses can improve by as much as 90%.

EXTENDING LEGACY TO THE WEB

In-memory table management also helps to support personalization, and is proving particularly useful for extending legacy applications to the Web for real-time, high-volume e-commerce or e-business. It can help accommodate the resulting increase in traffic volumes within existing mainframe CPU, LPAR, and storage constraints, reducing or even eliminating the

need for hardware reallocation or additional hardware resources.

Three specific features make table management technology ideally suited for implementing fine-tuned personalization into corporate portal and e-business applications: speed, flexibility, and the capability to develop rules-based applications. All three support a successful delivery of customized user experiences to individual users. For instance, banking institutions that have adopted this technology reportedly credit memory-based table management as the enabling technology for providing personalized statements via the Web. These banks also use table management technology to produce consolidated and personalized master account statements for their customer base. To achieve this, the table-management-centric statement engine application applies 15 to 20 dynamic business rules per customer account in the process of handling millions of transactions per day.

In-memory table management is also suited for implementing incisive rules-based security policies for Web sites and corporate portals, and can be used for interactive, high-volume currency conversion applications as well as sales tax calculation applications. One of Ireland's largest banking corporations uses an in-memory table management system to handle all of its interactive currency conversion requirements, including the standardization to the euro currency in 2002. The bank's currency conversion application falls into the repetitive data-access category discussed earlier. Real-time sales tax calculation for point-of-sale applications and online mail-order applications also fall into this repetitive processing category. So do many security-related functions pertaining to corporate portals, whether they are for rules-based security policy enforcement, such as controlling access rights, or data encryption and decryption functions, like those related to SSL security.

BOTTOM LINE

Application design that takes advantage of tables can facilitate a host of requirements, from adding new functionality to existing

applications without requiring any modification or rewrites, to extending mainframe applications to the Web. Combining this approach with an in-memory table management system enables companies to speed-up process-intensive applications, support increased user volumes, optimize system efficiency and performance, facilitate and accelerate application development, and ultimately reduce costs. All in all, in-memory table management is a 20-year-old software methodology that is still in profitable use today, granting a continued lease on life to mission-critical mainframe applications.

Robert Koblovsky
Vice President of Sales and Marketing
Data Kinetics Ltd (USA)

BMC Software has announced three new MAINVIEW Storage Resource Manager (SRM) solutions – MAINVIEW SRM Allocation, MAINVIEW SRM Reporting, and MAINVIEW SRM Automation.

These products help customers to manage their mainframe storage environment by reducing storage costs and space shortages, providing automation capabilities that maximize staff and CPU time and minimize wasted space and schedule interruptions.

With these new products, BMC has automated allocation features to reduce the incidence of space-related processing problems and automated actions for the prevention and recovery from abends, ensuring higher levels of availability.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042, USA.
Tel: (713) 918 8800.
URL: http://www.bmc.com/products/proddocview/0,2832,19052_19429_6115596_9804,00.html.

* * *

Serena Software has announced Version 4.1 of its Application Performance Manager, StarTool.

StarTool APM is a performance measurement and analysis system that helps to resolve OS/390 and z/OS job performance issues. Users can focus their activities on tuning specific areas of an application to boost productivity.

Version 4.1 provides improved, real-time monitoring capabilities with extended support for a wider range of subsystems including CICS, IMS, and DB2.

In addition, Version 4.1 delivers better reporting by providing customers with more in-depth information on an application to help quickly solve performance problems and eliminate the need for further analysis.

For further information contact:
Serena Software, 2755 Campus Drive, 3rd Floor, San Mateo, California 94403, USA.
Tel: (650) 522 6600.
URL: http://www.serena.com/product/aa_st_apm_ov.html.

* * *

iWay Software has announced improvements in its data integration capabilities, which include new cluster management, remote installation and management, tighter integration with SQL Server, and new data adapters.

The Remote Installation/Management Console is used for remote installations, which reduces the manual tasks involved in an OS/390, z/OS server installation. Server installations are simplified because the installation files can be transferred and configured from a local laptop or a remote computer, eliminating the need for certain types of technical and operational support.

Intelligent adapters have been added and enhanced to support DB2 Version 8 and IMS Version 8. Additionally, iWay's Adapter for CICS now supports RRS (Resource Recovery Services) and integrated two-phase commit.

For further information contact:
iWay Software, Two Penn Plaza, New York, NY 10121-2898, USA.
Tel: (212) 330 1700.
URL: http://www.iwaysoftware.com/products/index.html.