160 MVS

January 2000

update

In this issue

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

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

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

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £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 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Unique SYSLOG identification across systems

INTRODUCTION

We have been adding MVS system images to our current sysplex environment recently and have come across an issue of correctly identifying the SYSLOG datasets from each system, in particular when archiving into our SYSOUT archiving package, CA-View. When we had only three LPARs, it was easy to assign a unique JES2 SYSOUT class to each SYSLOG dataset. The unique SYSOUT class was used by a CA-View exit to assign a unique name to each systems SYSLOG when it was archived, facilitating proper selection from an ISPF panel. As we have grown, it has not been feasible for us to use additional SYSOUT classes, because we are running short of available classes (JES2 is limited to 36 classes between A to Z and 0 to 9). We needed to come up with another means of uniquely identifying each SYSLOG dataset to the archival system in order to select them more easily for viewing.

I remembered seeing something in an IBM publication a few years ago addressing this issue. The publication is no longer in print, so I will not cite it here, but I will relay its contents. The suggestion was to use JES2 exit 40 (the modify SYSOUT characteristics exit) to move the SMF system identification into a field within the JES2 $PDDB (peripheral dataset definition block) control block, which would be otherwise meaningless for SYSLOG datasets. The $PDDB control block contains the characteristics of the JES2 system dataset it relates to.

Fields such as the form name, FCB, UCS, or writer name could be used. We elected to have exit 40 store the SMF system identification in the UCS field. Once there, it becomes available after the SYSLOG dataset is made available by the issuance of the MVS WRITELOG command, or when the SYSLOG line limit is reached as specified by the LOGLIM parameter of the IEASYS00 member of SYS1.PARMLIB. I resisted coding a JES2 exit just for this purpose at first. I tried using our console automation product to intercept the message that is issued when a SYSLOG is made available. My theory was that I could then issue a JES2 command to change the UCS when the new SYSLOG was allocated.Unfortunately, you cannot modify

characteristics of SYSOUT datasets until after they are spun off or otherwise made available for printing. Then I tried changing the UCS via a JES2 command once the SYSLOG was spun off. Again I was thwarted, because there are no commands available in JES2 to display the system on which the SYSOUT dataset was created (although IBM SDSF is capable of displaying both the input and execution system identifications). I relented and used the JES2 exit 40 approach.

Once we propagated this JES2 exit 40 across all systems in our shop, we modified our previous CA-View exit (SARSTCUX) to pick up the last byte of the UCS field (which now stored the SMF system identification of the system on which the SYSLOG was created) and appended it to the creating jobname (in this case the job name is SYSLOG). This then becomes a unique SYSOUT identification within the CA-View archiving system. Our SMF system identification conventions are such that the last byte is the significant identifier of the system (such as SYSA, SYSB, SYSC, etc). After implementing the new SARSTCUX code, and modifying the LOGCLS parameter of the IEASYS00 member of SYS1.PARMLIB on all MVS images to specify the same SYSOUT class (which is the class CA-View is set up to archive), all SYSLOG datasets archived are now uniquely identified. Using the SMF names mentioned earlier, we are now able to specify SYSLOGA, SYSLOGB, SYSLOGC, etc, to uniquely select a systems SYSLOG. CA-View also affords the option of specifying SYSLOG* as a name to present a list of all SYSLOG datasets archived, regardless of the system they came from.

Although this implementation uses the CA-View product, it should be easily transportable to most other SYSOUT archiving systems, such as those available from IBM (RMDS and On/Demand), Software Engineering of America ($AVRS), and Mackinney Systems (JSF), etc. I would hope that any of these products would provide access to some of the fields in the JES $PDDB for use in uniquely indentifying each systems SYSLOG dataset.

SAMPLE JES2 EXIT 40

```
HASX40   TITLE 'JES2 USER EXIT 40'
*********************************************************************
* MODULE NAME = HASX40   (HASX40 load module)                       *
*                                                                   *
* DESCRIPTIVE NAME = SYSOUT CHARACTERISTICS SET                     *
```

```
*                                                                     *
* FUNCTION = Assign SMF sysid to UCS dataset                          *
*    DEPENDENCIES = JES2 $EXIT FACILITY,STD.JES2 SERVICES, ETC.       *
*    REGISTER CONVENTIONS = See entry point documentation             *
*    PATCH LABEL = NONE                                               *
*                                                                     *
* MODULE TYPE = Procedure ( CSECT type )                              *
*    PROCESSOR = IBM High Level Assembler/MVS                         *
*    MODULE SIZE = See $MODEND macro expansion at end of assembly     *
*    ATTRIBUTES = JES2 REENTRANT, RMODE ANY, AMODE 31                 *
*                                                                     *
* ENTRY POINTS = EXIT 40 SYSOUT CHARACTERISTICS SET                   *
*                                                                     *
* MACROS = JES2 - $ENTRY, $ESTAE, $MODEND, $MODULE, $RETURN, $SAVE,   *
*                 $SETRP                                              *
*                                                                     *
* MACROS = MVS  - CVT, SMCA                                           *
***********************************************************************
         TITLE 'JES2 USER EXIT 40 -- PROLOG   - $HASPGBL'
         COPY  $HASPGBL
         TITLE 'JES2 USER EXIT 40 -- PROLOG   - $MODULE'
HASX40   $MODULE ENVIRON=JES2,                                        C
               RMODE=ANY,                                             C
               IBMJES2=SAMPLE,                                        C
               TITLE='JES2 USER EXIT 40',                             C
               $CADDR,             JES2 CADDR DSECT                   C
               $ERA,               JES2 ERA DSECT                     C
               $HASPEQU,           JES2 HASPEQU DSECT                 C
               $HCT,               JES2 HCT DSECT                     C
               $JCT,               JES2 JCT DSECT                     C
               $JQE,               JES2 JQE DSECT                     C
               $MIT,               JES2 MIT DSECT                     C
               $MITETBL,           JES2 MTE DSECT                     C
               $PADDR,             JES2 PADDR DSECT                   C
               $PARMLST,           JES2 PARMLST DSECT                 C
               $PDDB,              JES2 PDDB DSECT                    C
               $PCE,               JES2 PCE DSECT                     C
               $PRE,               JES2 PRE DSECT                     C
               $PSV,               JES2 PSV DSECT                     C
               $USERCBS,           JES2 USERCBS DSECT                 C
               $XECB,              JES2 XECB DSECT                    C
               $XPL                JES2 XPL DSECT
         TITLE 'JES2 USER EXIT 40 -- Exit routine EXIT40'
***********************************************************************
* TITLE :    EXIT40                                                   *
* FUNCTION : Assign SMF sysid as UCS for SYSLOG datasets.             *
* LINKAGE :  BALR R14,R15                                             *
* OPERATION :                                                         *
*         LOADMOD(HASX40) STORAGE=PVT                                 *
*         EXIT(40) ROUTINES=EXIT40,STATUS=ENABLED                     *
*                                                                     *
*         MVS MUST BE BROUGHT UP                                      *
*                                                                     *
```

```
* ENVIRONMENT :                                                        *
*        JES2 MAIN TASK                                                *
*                                                                      *
* RECOVERY :                                                           *
*        $ESTAE RETRY ROUTINE SIMPLY RETURNS FROM THIS EXIT.           *
*                                                                      *
* RETURN CODES (R15 ON EXIT) :                                         *
*        ØØ - ENABLE OTHER EXITS ASSOCIATED WITH THIS ONE              *
*                                                                      *
* OTHER CONSIDERATIONS :                                               *
*        N/A                                                           *
************************************************************************
EXIT4Ø   $ENTRY BASE=R12           DECLARE ROUTINE ENTRY
         $SAVE  NAME=EXIT4Ø,TRACE=YES SAVE CALLER'S REGISTERS
         LR    R12,R15             GET ROUTINE ADDRESSABILITY
         LR    R7,R1               COPY PARAMETER LIST ADDR
         USING XPL,R7              DECLARE XPL ADDRESSABILITY
         $ESTAE RECADDR=RECRTN,    CREATE RECOVERY ENVIRONMENT         x
               RETRY=RETRYRTN,                                         X
               BASE=EXIT4Ø
         USING PRE,R1              DECLARE PRE ADDRESSABILITY
         MVC   PRETRACK(L'EXITMSG),EXITMSG  SET SDWA VRA TEXT
         MVI   PRELOGLN,L'EXITMSG            AND LENGTH
         DROP  R1                  DROP PRE ADDRESSABILITY
************************************************************************
*        CHECK FOR JOBNAME OF SYSLOG. IF FOUND, MOVE SMF SYSID INTO    *
*        THE PDBUCS FIELD.  THIS WILL BE USED AS AN INDICATOR AS TO    *
*        WHICH SYSTEM GENERATED THIS SYSLOG DATASET.                   *
************************************************************************
         L     R5,X04ØJQE          GET JQE  ADDRESS FROM XPL
         USING JQE,R5              DECLARE JQE  ADDRESSABILITY
         CLC   JQEJNAME,SYSLOG     IS THE JOB NAME SYSLOG?
         BNE   NOSETID             IF NOT, SKIP SETID
         TM    JQEFLAG3,JQE3STC    IS THIS A STARTED TASK?
         BZ    NOSETID             IF NOT, SKIP SETID
         L     R4,X04ØPDDB         GET PDDB ADDRESS FROM XPL
         USING PDB,R4              DECLARE PDDB ADDRESSABILITY
         L     R15,CVTPTR          GET CVT ADDRESS
         L     R15,CVTSMCA-CVT(,R15) GET SMCA ADDRESS
         MVC   PDBUCS(L'SMCASID),SMCASID-SMCABASE(R15) USE SMF ID
NOSETID  DS    ØH                  DONE WITH SETID
************************************************************
*        THIS IS POINT WHERE THE RECOVERY ROUTINE RETRIES    *
************************************************************
RETRYRTN DS    ØH                  RETRY POINT FOR RECOVERY
         $ESTAE CANCEL             CANCEL $ESTAE ENVIRONMENT
         $RETURN TRACE=YES         RETURN TO CALLER
EXITMSG  DC    C'ESTAE SET UP FOR EXIT 4Ø'     EXIT MESSAGE TEXT
SYSLOG   DC    CL8'SYSLOG'
         DROP  R4                  DROP PDDB ADDRESSABILITY
         DROP  R5                  DROP JQE  ADDRESSABILITY
         DROP  R7                  DROP XPL ADDRESSABILITY
         DROP  R12                 DROP ROUTINE ADDRESSABILITY
```

```
         LTORG ,                   DECLARE LITERAL ORIGIN
*************************************************************
*        RECOVERY ROUTINE                                   *
*************************************************************
         USING RECRTN,R12          DECLARE RTN ADDRESSABILITY
RECRTN   $SAVE  TRACE=YES          SAVE CALLER'S REGISTERS
         LR    R12,R15             GET ROUTINE ADDRESSABILITY
         LR    R8,R1               COPY ERA ADDRESS
         USING ERA,R8              DECLARE ERA ADDRESSABILITY
         L     R1,ERAPRE           ADDRESS OF PRE
         MVC   ERAREG12,PREBASE-PRE(R1) SET BASE FOR RETRY
         L     R2,PRERESUM-PRE(,R1) GET RESUME ADDRESS
         $SETRP RECOVER,RESUME=(R2) SET RECOVERY ADDRESS
         $RETURN TRACE=YES         RETURN FROM RECOVERY
         CVT   DSECT=YES,LIST=NO   MVS CVT DSECT
         IEESMCA                   MVS SMCA DSECT
         DROP  R8                  DROP ERA ADDRESSABILITY
         DROP  R12                 DROP ROUTINE ADDRESSABILITY
         LTORG ,                   DECLARE LITERAL ORIGIN
         $MODEND
         END   ,                   END OF HASX4Ø
```

Sample CA-View SARSTCUX exit follows:

```
SARSTCUX TITLE 'SARSTCUX - SARSTC USER EXIT'
*
************************************************************************
*                                                                      *
* *****         I M P O R T A N T   I N F O R M A T I O N    *****    *
*                                                                      *
*   WHEN SAR REJECTS A JOB FROM THE ARCHIVE, THE JOB IS REQUEUED       *
*   FOR OUTPUT UNDER THE NAME OF THE SAR STARTED TASK. THIS MEANS      *
*   THAT ALL REJECTED OUTPUT HAS THE SAME NAME AS THE SAR STARTED      *
*   TASK, AND HAS LOST ITS ORIGINAL JOB NAME. TAKE HEED.               *
*                                                                      *
************************************************************************
SARSTCUX CSECT
         EBCMODE   ,                SETS MODE TO 31/ANY BY DEFAULT
************************************************************************
* SARSTCUX -                                                           *
*        SAR ARCHIVAL STARTED TASK USER EXIT. THE EXIT IS INVOKED      *
*        WHEN A SYSOUT DATASET IS SELECTED FOR ARCHIVAL, BEFORE        *
*        EACH SYSOUT RECORD IS ARCHIVED, AND AFTER ALL SYSOUT HAS      *
*        BEEN PROCESSED FOR THE DATASET.  THE EXIT MAY BE USED         *
*        TO ALTER ATTRIBUTES IN THE GCR FOR THE DATASET AND TO         *
*        IDENTIFY EXCEPTIONAL CONDITIONS.                              *
*                                                                      *
* INPUTS -                                                             *
*        REG Ø - Ø, NEW SYSOUT DATASET SELECTED.                       *
*                4, SYSOUT RECORD PASSED TO EXIT FOR EXCEPTIONAL       *
*                   CONDITION ANALYSIS.                                *
*                8, NO MORE RECORDS FOR DATASET.                       *
```

```
*        REG 1 - ADDRESS OF PARAMETER LIST (MAPPED BY SARSPL).        *
*        REG13 - ADDRESS OF REGISTER SAVE AREA                        *
*        REG14 - RETURN ADDRESS                                       *
*        REG15 - ENTRY POINT ADDRESS.                                 *
*                                                                     *
* OUTPUTS (WHEN REG Ø = Ø ON INPUT) -                                 *
*        REG Ø - CONTENTS ARE IGNORED.                                *
*        REG 1 - CONTENTS ARE IGNORED.                                *
*        REG15 - CONTENTS ARE IGNORED.                                *
*                                                                     *
*        NOTES -                                                      *
*           1. THE GCR FIELDS MAY BE CHANGED AS DESIRED.              *
*                                                                     *
* OUTPUTS (WHEN REG Ø = 4 ON INPUT) -                                 *
*        REG Ø - ADDRESS OF WTO MESSAGE TO ISSUE.                     *
*        REG 1 - LENGTH OF WTO MESAGE TO ISSUE OR ZERO IF NO          *
*                MESSAGE IS TO BE ISSUED.                             *
*        REG15 - Ø, DO NOT CHANGE THE PRINT STATUS OF THE SYSOUT      *
*                   GROUP.                                            *
*                4, THE SYSOUT GROUP IS TO BE PRINTED PROVIDED        *
*                   INITIALIZATION OPTION XPRINT=YES IS IN EFFECT.    *
*                                                                     *
*        NOTES -                                                      *
*           1. GCRXCODE MAY BE CHANGED TO CONTAIN A SHORT             *
*              EXCEPTION MESSAGE.                                     *
*           2. GCRUSER MAY BE CHANGED IF DESIRED.                     *
*           3. THE CONTENTS OF THE SYSOUT RECORD MAY BE CHANGED,      *
*              AND ITS LENGTH CHANGED AS DESIRED.  THE MAXIMUM        *
*              LENGTH FOR THE RECORD IS 2Ø6 BYTES.  SETTING THE       *
*              RECORD LENGTH TO ZERO WILL CAUSE SAR TO DISCARD        *
*              THE RECORD.                                            *
*                                                                     *
* OUTPUTS (WHEN REG Ø = 8 ON INPUT) -                                 *
*        REG Ø - CONTENTS ARE IGNORED.                                *
*        REG 1 - CONTENTS ARE IGNORED.                                *
*        REG15 - CONTENTS ARE IGNORED.                                *
*                                                                     *
*        NOTES -                                                      *
*           1. GCRXCODE MAY BE CHANGED TO CONTAIN A SHORT             *
*              EXCEPTION MESSAGE.                                     *
*           2. GCRUSER MAY BE CHANGED IF DESIRED.                     *
*                                                                     *
* MODIFICATION NOTES: -                                               *
*        SPECIAL EXIT TO PRINT AND NOT ARCHIVE SYSOUTS THAT ARE       *
*        MORE THAN A CERTAIN AMOUNT OF LINES. THE FOLLOWING           *
*        MODIFICATIONS ARE NECESSARY TO IMPLEMENT THIS EXIT:          *
*                                                                     *
*        THIS MODIFICATION IS FOR USE UNDER JES2 ONLY.                *
*                                                                     *
*            1. INDICATE THE MAXIMUM AMOUNT OF LINES THAT ARE TO BE   *
*               ARCHIVED IN THE SETA STATEMENT FOR &MAXLNES (5ØØØØ    *
*               IS USED AS AN EXAMPLE).                               *
*            2. INDICATE THE PRINT CLASS FOR SYSOUTS GREATER THAN     *
```

```
*               THE MAXIMUM LINES IN THE SETC STATEMENT FOR &PRTCLS   *
*               (CLASS "P" IS USED AS AN EXAMPLE), AND THE INITIAL-   *
*               IZATION PARAMETERS "PRTCLSL" AND "NARCCLSL" MUST BE   *
*               SET TO THE SAME CLASS. THE "CLSL" INITIALIZATION      *
*               PARAMETER MUST NOT INCLUDE THIS SYSOUT CLASS, NOR     *
*               MAY THE SYSTEM EXTENSIONS HAVE THIS CLASS IN THE      *
*               "SYSCLASS" PARM.                                      *
* ATTRIBUTES -                                                        *
*        AUTHORIZED.                                                  *
**********************************************************************
         MACRO
&LABEL   DATADEF &NAME
         LCLA  &COUNT
&COUNT   SETA  K'&NAME-2
         AIF   (&COUNT LE 8).CHK1
         MNOTE 8,'NAME GREATER THAN 8 CHARACTERS'
         MEXIT
.CHK1    AIF   (&COUNT NE Ø).GEN
&LABEL   DC    AL1(Ø)               END OF TABLE INDICATOR
         AGO   .DONE
.GEN     ANOP
&LABEL   DC    AL1(&COUNT),CL&COUNT.&NAME LENGTH, VALUE
.DONE    MEND
         LCLA  &MAXLNES
         LCLC  &PRTCLS
&PRTCLS  SETC  'Y'                  PRINT CLASS
&MAXLNES SETA  1ØØØØ                MAX ARCHIVAL LINES ( NONE FOR NOW )
         SAVE  (14,12),,SARSTCUX-&SYSDATE
         LR    R12,R15               LOAD BASE REGISTER
         USING SARSTCUX,R12
         LR    R4,R1                 SAVE A(PARM LIST)
         USING SPL,R4                ADDRESSABILITY FOR PLIST
         LTR   RØ,RØ                 VERIFY NEW DATASET SELECTED
         BNZ   RETURN                NO, THEN RETURN
         L     R1Ø,SPLGCR            LOAD ADDRESS OF GROUP CNTL RECORD
         USING GCR,R1Ø
         CLC   =CL8'SYSLOG',GCRJNAM IS THIS JOB A SYSLOG PRINTING
         BNE   CHKLNES               NO, THEN GO CHECK MAX LINES
         CLC   =C'SYS',GCRUCS        CHECK IF UCS FIELD HAS SMF ID
* ABOVE DEPENDS ON CONVENTION OF SMF SYSIDS STARTING WITH 'SYS'
         BNE   RETURN                NO, JUST GO RETURN
         MVC   GCRID,BLANKS          CLEAR JOB ID FIELD
         MVC   GCRID,GCRJNAM         MOVE 'SYSLOG' AS JOB ID PREFIX
         MVC   GCRID+6(1),GCRUCS+3   MOVE LAST CHAR OF UCS AS ID SUFFIX
* THE ABOVE DEPENDS ON JES2 EXIT4Ø SETTING THE UCS FIELD FOR SYSLOGS
* WITH THE SMF SYSID (E.G. SYSA/SYSB/SYSM/ETC.) AND ARBITRARILY USING
* THE LAST BYTE OF THE SMF SYSID AS THE JOB ID SUFFIX
         B     RETURN                AND RETURN
CHKLNES  L     R3,SPLSSCM            LOAD ADDRESS OF SDB
         MODESET EXTKEY=ZERO,SAVEKEY=(2),WORKREG=2 GET KEY ZERO
         MVC   DENYMSG+34(8),SDBDDNM-SDB(R3) SAVE DDNAME IN ERROR MSG
         L     R3,SDBPDDB-SDB(,R3)   LOAD PDDB ADDRESS?
         L     R3,PDBRECCT-PDB(,R3) LOAD LINE COUNT FOR DATASET?
```

```
         MODESET KEYADDR=(2)        GET BACK PROBLEM KEY
         CLC   GCRJNAM(3),=CL3'TSO' IS THIS A TSO USER'S JOB?
         BE    CHECKNAM             IF SO CHECK AUTHORIZATION
         C     R3,=F'&MAXLNES'      IS SYSOUT DATASET SIZE > &MAXLNES
         BNH   RETURN               NO, THEN ALLOW ARCHIVE
CHECKNAM EQU   *
         LA    R8,EXCLUDE           POINT TO EXCLUSION LIST
         SR    R9,R9                CLEAR R9 FOR ICM
LOOP     ICM   R9,1,Ø(R8)           GET COMPARE LENGTH
         BZ    NAUGHTY              AT ZERO LENGTH GIVE UP SEARCH
         BCTR  R9,RØ                DECREMENT FOR EXECUTE
         EX    R9,COMPNAME          CHECK FOR MATCH
         BE    RETURN               IF IT MATCHES, ALLOW ARCHIVING
         LA    R8,2(R9,R8)          POINT TO NEXT ENTRY IN TABLE
         B     LOOP                 IF NO MATCH CHECK NEXT ENTRY
*
COMPNAME CLC   GCRJNAM(Ø),1(R8)     INSTRUCTION TO BE EXECUTED IN LOOP
*
NAUGHTY  MVI   GCRCLASS,C'&PRTCLS'  ELSE SET SYSOUT CLASS TO PRINT
         MVC   DENYMSG+16(8),GCRJNAM PUT JOB NAME INTO MESSAGE
         MVC   DENYMSG+25(8),GCRJID PUT JOB ID INTO MESSAGE
DENYMSG  WTO   'SSXØØ1I ******** ******** ******** WAS DENIED ACCESS TOX
                THE SAR ARCHIVE',ROUTCDE=11
         B     RETURN               GO RETURN
RETURN   SR    R15,R15              ZERO OUT RETURN REGISTERS
         LR    RØ,R15
         LR    R1,R15
         L     R14,12(,R13)         RESTORE REGISTER 14
         RETURN (2,12),RC=(15)      RESTORE REGISTERS AND RETURN
BLANKS   DC     CL16' '
EXCLUDE  DATADEF 'TSOSP1'           ALLOW TSOSP1   JOBS OVER LINE COUNT
         DATADEF 'TSOSP2'           ALLOW TSOSP2
         DATADEF 'TSOSP3'           ALLOW TSOSP3
         DATADEF 'TSOSP4'           ALLOW TSOSP4
         DATADEF '******'           SPARE ENTRY FOR ZAPPING
         DATADEF '******'           SPARE ENTRY FOR ZAPPING
         DATADEF ''                 END OF TABLE
*
         LTORG
         PRINT NOGEN
         IOSDIOBE                   REQUIRED BY $SDB
         IOSDIEDB                   REQUIRED BY $SDB
         SARSPL
         SARMCR
         SARGCR
SARSTCUX CSECT
         $TAB
         $SDB
         $PDDB
         $HASPEQU
         END
```

Cache status management

INTRODUCTION

Maintaining the cache storage on 3990 control units is a very simple job, but how many times do you have to look and search if the DASD-fast-write option is set on or if the non-volatile storage is on, or even if there is pinned data on that subsystem? Probably there are regular daily jobs or 'message capture' products that are triggered by the SIM (if it is produced) on the operator console, but even then, the operator has to find you to test the condition and set the cache command on.

In other, more simple, situations, the 3990 technicians ask you to turn the cache on or off for the control unit maintenance.

The work that is common to all the above situations was listing the 3990 subsystem and in some way copying the output to a member in a PDS (manipulating it with all kinds of EDIT commands). Until you are satisfied with the results, you never have the accurate subsystem layout and always have to check it against the 'LISTDATA' command.

Well, here is a simple REXX routine with the following steps and functions:

- LIST the cache status into a sequential file.
- CHECK against the 3990 control unit on DASD-fast-write/ cache-fast-write/NVS status.
- CREATE SYSIN on two sequential files with all the relevant SETCACHE commands for later use.
- Notify about rinned-data in the control unit.
- Simple DASD capacity calculation separated by VOLSER names.

Notes:

1 The only parameter required for this job is the first or any of the subsystems DASD VOLSERs.

2 It is assumed that there is a naming convention in the VOLSER name (the first four characters in the VOLSER).

3 The calculation is made on 2.83GB DASD capacity (3390-3 model).

4 The calculation is made for *on-line* devices only!

Complete batch job:

```
//SØ38XXX  JOB (SS38,B1,3Ø,S),LIST-CACHE-STATUS,
//         NOTIFY=SØ38,MSGCLASS=T,REGION=5ØØØK,CLASS=S
//LISTD   EXEC PGM=IDCAMS
//SYSPRINT DD DSN=SØ38.TEXT3,DISP=SHR
//SYSIN    DD *
    LISTDATA VOLUME(MVSØØ1) UNIT(339Ø)  DSTATUS
    LISTDATA VOLUME(SPLØØ3) UNIT(339Ø)  DSTATUS
/*
//*
//REXX     EXEC ISPBATCH
//SYSPROC  DD DSN=SYS2.CLIST,DISP=SHR
//SYSTSIN  DD *
  PROFILE NOPREFIX
  ISPSTART CMD(CACHERX2) BDISPMAX(99999999)
/*
//
```

CACHSTAT

```
/* ———————————————————— REXX ———————————————————— */
/*                                                  */
/* This REXX routine does the following tasks:      */
/*                                                  */
/* 1. Checks the cache status in the 399Ø from      */
/*    'LISTDATA VOLUME(xxxxxx) UNIT(339Ø)  DSTATUS' */
/*    command, generates and SUBMITS - to the needed*/
/*    devices - the appropriate jobstream           */
/*    'SETCACHE VOLUME(xxxxxx)' .                   */
/*                                                  */
/*    The EditMacro eliminates the unneeded lines from*/
/*    the LISTDATA output.                          */
/*                                                  */
/* 2. Dynamically generate CACHE Subsystem ON/OFF for*/
/*    the required 399Ø subsystem as member in PDS for*/
/*    later use.                                    */
/*    THE CACHE ON input cards is allocated in:     */
/*        'SØ38.LST.CACHON' dataset.                */
/*    THE CACHE Off input cards is allocated in:    */
/*        'SØ38.LST.CACHOFF' dataset.               */
/*                                                  */
/* 3. Notify if there is PINNED DATA on the CU for  */
/*    each DASD.                                    */
```

```
/*                                                          */
/* 4. Simple DASD and Gigabyte calculation for the          */
/*    required control unit (ONLY ON-LINE DASD are          */
/*    taken for calculation!!)                              */
/*    The calculation takes for granted that the first      */
/*    four characters in the VOLSER are in naming           */
/*    convention standard.                                  */
/*                                                          */
/* ———————————————————————————————————————————————————— */
TRACE NO
ADDRESS ISPEXEC "EDIT DATASET('SØ38.TEXT3') MACRO(CACHEED1)"
ADDRESS TSO 'ALLOC FILE(IN) DA(SØ38.TEXT3) SHR'
ADDRESS TSO 'ALLOC FILE(OUT) DA(SØ38.LST.SETC) SHR'
ADDRESS TSO 'ALLOC FILE(O1) DA(SØ38.LST.CACHON) SHR'
ADDRESS TSO 'ALLOC FILE(O2) DA(SØ38.LST.CACHOFF) SHR'
I = Ø
T = Ø
N = 5
NUM = Ø
TOT = Ø
DSDT = 'XXXX'
DEV1 = Ø
LOOP1:
"EXECIO 1 DISKR IN "
IF RC > Ø THEN SIGNAL EXIT
  PULL LINE
   PARSE VAR LINE DEVN VOL CCA DDC CACH DFW DUP PIN NVS
   TOT = TOT + 1
   IF SUBSTR(VOL,1,4) > DSDT THEN DO
      SAY 'There are ' NUM * 2.83 || 'GB IN '|| DEV1 DSDT || ' DASDs'
      NUM = Ø
      DEV1 = Ø
   END
         DSDT = SUBSTR(VOL,1,4)
         NUM = NUM + 1
         DEV1 = DEV1 + 1
/*———————————————————————————————————————————————————— */
/*                                                          */
/* Prepare input for later process for CACHE ON/OFF         */
/*                                                          */
/*———————————————————————————————————————————————————— */
O1. =
O1.1 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) SUBSYSTEM ON ;'
O1.1 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) CFW ON ; '
O1.2 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) DFW ON ; '
O1.3 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) NVS ON ; '
  O2. =
  O2.1 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) SUBSYSTEM OFF ;'
  O2.1 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) CFW OFF ; '
  O2.2 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) DFW OFF ; '
```

```
  O2.3 = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) NVS OFF ; '
    "EXECIO * DISKW O1 (STEM O1."
    IF RC > Ø THEN DO
              SAY 'ERROR WRITING ON DATASET : 'O1
              SIGNAL EXIT
              END
    "EXECIO * DISKW O2 (STEM O2."
    IF RC > Ø THEN DO
              SAY 'ERROR WRITING ON DATASET : 'O2
              SIGNAL EXIT
              END
CACHE:
 SELECT
   WHEN CACH = 'NO ' THEN DO
     IF T = Ø THEN DO
       OUT. =
       OUT.1 = '//SØ38CONI JOB (SS38,B1,Ø5),CACHE-ON-339Ø,CLASS=J,'
       OUT.2 = '//        NOTIFY=SØ38,MSGCLASS=T,REGION=5ØØØK'
       OUT.3 = '//SETC   EXEC PGM=IDCAMS'
       OUT.4 = '//SYSPRINT DD SYSOUT=*'
       OUT.5 = '//SYSIN    DD *'
       T = 1             /* flag for 1st time */
       END               /* if */
     N = N + 1
     OUT.N = ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) DEVICE  ON'
     END                 /* when    */
DFW:
   WHEN DFW  = 'NO ' THEN DO
     IF T = Ø THEN DO
       OUT. =
       OUT.1 = '//SØ38CONI JOB (SS38,B1,Ø5),CACHE-ON-339Ø,CLASS=J,'
       OUT.2 = '//        NOTIFY=SØ38,MSGCLASS=T,REGION=5ØØØK'
       OUT.3 = '//SETC   EXEC PGM=IDCAMS'
       OUT.4 = '//SYSPRINT DD SYSOUT=*'
       OUT.5 = '//SYSIN    DD *'
       T = 1             /* flag for 1st time */
     END                 /* if */
    N = N + 1
    OUT.N =  ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) DFW ON'
    N = N + 1
    OUT.N =  ' SETCACHE VOLUME(' || VOL || ') UNIT(339Ø) NVS ON'
   END                   /* end    */
PIN:
  WHEN PIN  = 'YES' THEN
     SAY 'THERE IS PINED DATA ON: ' || VOL
   OTHERWISE SIGNAL LOOP1
   END                   /* select */
SIGNAL LOOP1
EXIT:
    "EXECIO * DISKW OUT (STEM OUT."
```

```
    IF RC > Ø THEN DO
              SAY 'ERROR WRITING ON DATASET : 'OUT
              SIGNAL EXIT
              END
  "EXECIO Ø DISKR IN (FINIS"
  ADDRESS TSO "FREE F(IN)"
  "EXECIO Ø DISKW OUT (FINIS"
  ADDRESS TSO "FREE F(OUT)"
  "EXECIO Ø DISKW O1 (FINIS"
  ADDRESS TSO "FREE F(O1)"
  "EXECIO Ø DISKW O2 (FINIS"
  ADDRESS TSO "FREE F(O2)"
SAY '================================================'
  SAY 'YOU HAVE ' || TOT * 2.83 || 'GB IN ' || TOT || ' ONLINE DEVICES'
SUBMIT:
IF T ¨= Ø THEN   /* checks if there is some changes to do */
ADDRESS TSO 'SUB SØ38.LST.SETC'
ELSE
  EXIT
```

CASHEED1 EDIT MACRO

This macro eliminates all the unneeded data from the output produced by the LISTDATA command:

```
/* REXX */
ADDRESS ISREDIT "MACRO"
ADDRESS ISREDIT "C 'Ø' ' ' 1 ALL"
ADDRESS ISREDIT "X '      ' 2 ALL"
ADDRESS ISREDIT "X 'IDC' ALL"
ADDRESS ISREDIT "X 'LISTDATA' ALL"
ADDRESS ISREDIT "X 'DEV' ALL"
ADDRESS ISREDIT "DEL X ALL"
ADDRESS ISREDIT "SORT 9"
ADDRESS ISREDIT "SAVE"
ADDRESS ISREDIT "END"
ADDRESS ISREDIT "MEND"
RETURN
```

OUTPUT

Cache on/off commands:

```
 SETCACHE VOLUME(MVS2Ø2) UNIT(339Ø) CFW OFF ;
 SETCACHE VOLUME(MVS2Ø2) UNIT(339Ø) DFW OFF ;
 SETCACHE VOLUME(MVS2Ø2) UNIT(339Ø) NVS OFF ;
 SETCACHE VOLUME(CTDØØØ) UNIT(339Ø) CFW OFF ;
 SETCACHE VOLUME(CTDØØØ) UNIT(339Ø) DFW OFF ;
```

```
SETCACHE VOLUME(CTD000) UNIT(3390) NVS OFF ;
SETCACHE VOLUME(TEMP01) UNIT(3390) CFW OFF ;
SETCACHE VOLUME(TEMP01) UNIT(3390) DFW OFF ;
SETCACHE VOLUME(TEMP01) UNIT(3390) NVS OFF ;
```

Sysout calculations for DASD capacity:

```
READY
  PROFILE NOPREFIX
READY
  ISPSTART CMD(CACHERX2) BDISPMAX(99999999)
There are  8.49GB IN 3 CTD0 DASDs
There are  19.81GB IN 7 DB2C DASDs
There are  48.11GB IN 17 DB2P DASDs
There are  16.98GB IN 6 DB2T DASDs
There are  56.60GB IN 20 HDSA DASDs
There are  5.66GB IN 2 MVSD DASDs
There are  25.47GB IN 9 MVSS DASDs
There are  2.83GB IN 1 MVSU DASDs
There are  56.60GB IN 20 MVS0 DASDs
There are  14.15GB IN 5 MVS1 DASDs
There are  14.15GB IN 5 MVS2 DASDs
There are  19.81GB IN 7 MVS3 DASDs
There are  8.49GB IN 3 MVS4 DASDs
There are  2.83GB IN 1 MVS5 DASDs
There are  82.07GB IN 29 PERM DASDs
There are  8.49GB IN 3 SPL0 DASDs
There are  8.49GB IN 3 TEMP DASDs
There are  22.64GB IN 8 TEST DASDs
=====================================================
YOU HAVE 430.16GB IN 152 ONLINE DEVICES
 S038.SPFLOG1.LIST has been kept.
READY
END
```

Oded Tagger
Systems Programmer (Israel)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from the Xephon Web site. The URL is: www.xephon.com/contnote.html.

DASD tuning

INTRODUCTION

With the advent of 'storage processors' (eg, IBM RVA, HDS 7700E, EMC Symmetrix, etc) conventional DASD tuning methodology has to be changed. This article provides a methodology for identifying and curing DASD performance problems.

To understand the new world of storage processors, a brief explanation of DASD subsystem evolution is desirable, since the changing world in some cases requires different adjustments (ie tuning). Tuning requires an understanding of DASD measurement data; with evolution, DASD response time components acquired different meanings. A method of DASD tuning in the 'old' and 'new' environment follows.

DASD SUBSYSTEM EVOLUTION

There are three major phases of evolution:

- *Ancient uncached DASD world.* This world did not know about caches. Much of DASD tuning folklore and understanding originates from this environment. Performance was characterized by a DASD response time of about 20-30ms. This world does not exist anymore, except in isolated places.
- *Old DASD world.* Characterized by small (<= 64MB) cache control units. Average DASD response times of around 15-20ms become the norm. These cache controllers (eg, a 3990-3) are '1-stage' storage processors, ie for cache misses data is transferred directly from disk to channel at disk speed.
- *New DASD world.* Characterized by the appearance of large caches (>=256 MB) and the existence of new 'two-stage' proprietary storage processors (eg, IBM/STK RVA, EMC Symmetrix, HDS 7700, IBM/STK RSA, IBM RAMAC, Amdahl Spectris). Data in these boxes that is requested by the CPU is always transferred from cache at channel speed.

TWO-STAGE STORAGE PROCESSORS

For cache misses, in the first stage of activity data from disk is transferred to cache; in the second stage data is transferred from cache to the channel, at channel speed, as if they were cache hits. Also, all writes go to the cache first, and they are destaged from there subsequently. Large caches of 1000 MB or more are typical, resulting in high (90+%) hit ratios. Consequently, average response times are typically below 10ms (or even 7ms), since response times for cache hits are very low (long DB2 chains represent an exception). Typically, ESCON channels are used with nominal transfer rates of 17-18 MB/s, but protocol times have been fairly large (around 1ms) and only recently pushed down to the sub-millisecond arena. Most of these storage processors support logical (also called 'functional', 'flex' or 'hiper') volumes – these are the ones recognized by MVS. Typically, a logical volume is mapped to many physical volumes. Conversely, physical volumes are not dedicated to a single logical volume any more (many-to-many mapping).

DASD RESPONSE TIME

The basic equation for DASD performance (as measured by DASD response time) is familiar. The equation is universally valid, but the meaning and the importance of its elements are always changing with developments in technology. Equation (1) states that the response time (RT) is the sum of four components:

```
(1)  RT = IOSQ + DISC + CONN + PEND
```

We can analyse the four components to understand what determines their magnitude. RMF calculates IOSQ. The disk subsystem 'measures' the value of the other three components, reporting their value to the channel subsystem. Extensive experience indicates that DASD performance time tends to be good when response time does not exceed 1.5 times the sum of DISC and CONN times (eg queueing is not significant).

```
(1a)  RT < 1.5 x (DISC + CONN)
```

Connect time (CONN)

CONN is the average time for the control unit during an I/O operation while connected to the channel for the purpose of transferring data to the channel from the disk subsystem. Equation (2) states that:

```
Connect time = data transfer time + protocol time.
(2)  CONN = XFER + PR
```

The protocol time is a characteristic of the disk subsystem. Essentially it represents the 'overhead' associated with the I/O operation, caused (primarily) by the execution of microcode in the storage processors or control units. The complexity of the microcode and the microprocessors used determine its magnitude.

In the 'old world' of control units, protocol times of around 1ms were typical when parallel channels were used.

In the 'new world' of storage processors with ESCON channels protocol times started out higher than before (2-3ms); but values are now being reduced to 1ms and below. Equation (3) states that 'transfer time = block size/transfer rate':

```
(3)  XFER = BLK / XRT
```

In the new world, channel speed (for example, ESCON speed of 17 MB/s) determines data transfer time when transferring data to or from cache, which is always true for two-stage controllers.

Disconnect time (DISC)

The disconnect time occurs for cache misses only. Hence, it usually indicates the effectiveness of cacheing (for random retrievals).

In the new world, DISC represents delays internal to the storage processors. These may be caused by mechanical or electronic queueing faults. If a request is not satisfied from cache and data must be transferred into cache from disk, a DISC time of typically about 15-25ms is encountered. This DISC time is added to the CONN time encountered while transferring the data from cache into the CPU. A typical (weighted average) DISC time is about 2ms.

Service time

```
(4)  STa = CONN + DISC
```

The sum of CONN and DISC is considered to be the service time, STa.

IOSQ time

IOSQ occurs when a second or subsequent request arises from the same MVS system to a logical device (UCB) already considered busy by MVS. If IOSQ is high (and other components are not), then perhaps the volume should be split. This was typically real physical device contention in the old world, while it is typically logical device contention in the new world.

PEND time

PEND time occurs for path busy situations only in the new world. (Physical devices are not visible to MVS.) A typical value is less than 1ms under normal conditions.

THE NEED FOR TUNING

DASD tuning becomes necessary when DASD performance is inadequate. It is essential to recognize tuning needs when an application (eg CICS) response time or throughput is 'bad' because of high DASD response time. 'Bad' DASD response time can be recognized by decomposition of the application response time (also called degradation analysis) or by regression analysis. Also, a performance analyst may find high DASD response time by examining DASD measurement reports and/or high DASD activity rates and/or high DASD utilization and/or high DASD response time volumes (defined later). Note that finding the problem is not sufficient, the performance analyst has to act on the identified problem. In determining 'bad' response times and analysing them, expectations stated such as 'PEND time is typically less than 1ms' are used to help decision making.

WHAT CONSTITUTES DASD TUNING?

It is possible to improve DASD performance by:

- *Acquiring more hardware* – more or faster DASD actuators, more

or faster channels, more or faster control units, more cache or NVS.

- *Moving data* – this usually means moving some datasets from a badly performing (heavily used, sluggish) volume (or control unit) to a relatively unused volume (or control unit). Note that in the new world, since volumes are logical, the delays are not (usually) caused by physical contention, but by contention in a logical volume. Hence, the logical volume requires splitting. Occasionally, however, the new world requires changing the logical volume to physical volume mapping.
- *Changing application I/O usage* – for example, use larger block sizes. Note, that this becomes very important in the new world, where protocol times of 1ms are comparable to transfer times of 1ms for a 16Kb block. Thus, small block sizes encounter heavy overheads.

Implicit assumptions in tuning

Measurements usually represent average values, not instantaneous ones. (The latter is very difficult to handle.) Installation workloads are repetitive. Thus, bad transaction response times or throughput values occurring at 2pm yesterday are likely to occur at the same time today and tomorrow. Consequently, one uses RMF reports of yesterday to correct today's problem and thereby avoid tomorrow's problem.

Frequent new world tuning situations

MVS measurements are related to logical, not physical, devices. The actual data is usually distributed over an array of physical devices. Measurements internal to the storage processor are usually not (yet) available and often no internal tuning knobs are available. (The internal algorithms used are proprietary and frequently unknown.) Thus, there is an occasional need for analysis of the type that requires estimating physical device and path usage in the storage processor in order to solve problems internal to the storage processor. As an alternative, the manufacturers may provide proprietary data reduction tools which report internal physical device and path usage. The use of such tools (such as IXFP for IBM RVA, EMC Manage for Symmetrix, GRAPHTRACK for HDS 7700) is often desirable.

Symptoms

MVS (logical) response times exceeding 7ms (individually or on average).

High average response time, typically (greater than 7ms) on a storage processor, as shown for example for an LCU in RMF. This usually indicates an overloaded storage processor. Contention can be caused by to few paths, too few (or insufficiently powerful) microprocessors, inadequate cache sizes, back store overload, or microcode problems.

Solutions

Reduce the storage processor load by moving some data and hence some I/O activity to another storage processor, or by adding more cache (this reduces protocol times and back store load).

If more than one storage processor is used in an installation, load skew will occur, thus, the second and subsequent storage processors will support fewer I/Os. Load balancing across storage processors may be required.

Microcode problems may occur in special situations such as paging or PDSs or load libraries being inadequately handled.

One big (and fairly new) problem with the new storage processors is the limited number of paths available. (Caution: information stated here changes rapidly.) The number of paths limits throughput. Pth numbers are:

- 4 paths – 3990/RAMAC, RVA maximum I/O rate is around 1200 I/Os per second; a better plan is to support 700-900 I/Os per second.
- 8 paths – Symmetrix 3, RVA2T, 7700. Maximum may be 2400 I/Os per second; a better plan would be to support 1400-1600 I/Os per second.
- 16 paths – RSA, Symmetrix 4 and Spectris. Maximum I/O rate is about 3500 I/Os per second; a better plan is to support 2400-2800 I/Os per second.
- 32 paths – 7700E can have 32 paths. Maximum I/O rate is about

6500 I/Os per second; a better plan is to support 4500-5000 I/Os per second.

Symptoms

High (individual) logical volume response time.

Solutions

If IOSQ time is high (by itself, without an accompanying excessive DISC time), ie there is a contention on the logical volume, it must be split, which means that either datasets have to be moved off the volume or, if there is only a single dataset on the volume, the dataset must be split. This problem is essentially unchanged from the old world. Most storage processors allow logical volume definitions smaller than the emulated 3390-3s. The need to place multiple datasets on a single (logical) volume does not exist any more as it did when volumes were also physical and leaving unused space meant wasting physical storage.

If DISC time is high, that usually means the data for the logical volume is not retrieved from cache. This is most often because of inadequate cache sizes (or lack of record level caching). The solution is to obtain more cache. This can be accomplished by moving the dataset (logical volume) in question to another storage processor with more cache, actually purchasing more cache, or possibly removing some other dataset that uses cache heavily to another place in order to relieve cache contention.

High DISC time sometimes means that the storage processor has internal contention problems. This may be because multiple SCSI volumes exist on the same string (an internal path contention problem). This particular phenomenon appeared sometimes on EMC Symmetrix 5500s, because there is only a single path used for accessing up to eight physical volumes.

The phenomenon is likely to occur primarily with sequential access to multiple physical volumes that happen to be on the same string. The remedy is to alter the logical (hiper) volume to physical volume mapping. This usually requires intervention by the supplier customer engineer.

High DISC time may sometimes be caused by actual physical device contention.

This particular problem can occur on RAID-5 implementations, if care is not taken in mapping logical volumes to physical volumes in such a way that several heavily used volumes requiring back store access (low cache hit ratio, or heavy writing access) are sharing the same RAID-5 array. This could occur, for example, with RAMAC if the volumes in a drawer are heavily used, or in Symmetrix when multiple heavily used hiper volumes are mapped to the same physical volume. Usually, in either case, sequential access or heavy write content causes the problem. The remedy can be different logical to physical volume mapping usually requiring intervention by the supplier customer engineer.

Other events that can cause similar phenomena:

- The storage processor does not recognize sequential patterns (eg RAMAC prior to March 1996 not recognizing keyed sequential VSAM datasets).
- Microcode problems, for example, early Iceberg problems in handling PDSs.
- Putting very large temporary datasets on RVA. RVA writes these out, somewhat unnecessarily exercising the back store.
- Just cache-unfriendly data (might be solved by using solid state devices).

APPROACH TO TUNING

As a first method of attack, one could examine RMF DASD reports for potential problems. Thus, one may scan all the volumes in the DASD report to find ‘problem volumes’. This approach could work, but nowadays many installations have terabytes’ worth of data, ie hundreds if not thousands of volumes. To look at (or print out) all those reports is very wasteful.

Hence it is desirable to narrow the search to only volumes ‘worth’ looking at. These we will call ‘interesting DASD’ (the term was introduced in an IBM product, now discontinued). One becomes

concerned with (interested in) volumes when their response time is high. What is considered high of course depends on the particular installation. The installation may specify that any response time, RTa, exceeding some value (say 7ms) is high, or may define 'threshold response times' for categories of volumes. When these volumes have been identified, and sorting can be used to find them, further examination of these volumes can easily lead to tuning solutions. This approach represents a more refined second method of attack (fewer volumes have to be examined).

A moment's reflection reveals, however, that one should not be concerned with high response time, RTa, when the corresponding I/O rate, Sa, is low. (In particular, many volumes in an installation are hardly used, perhaps the usual '20-80' rule applies, that is 20% of the volumes generate 80% of the load.) This argument leads to the thought that the impact of a volume's performance to overall performance of the system (and of the DASD subsystem) can best be assessed by using a 'response time x I/O rate' product, called now 'I/O intensity' (this term is now extensively used in a number of products), previously also called 'response time volume' or 'DASD MPL'. In other words, the 'interesting DASD' could be found among those with high I/O intensity. This approach is a third cut at identifying the interesting DASD. Hence the method calls for selecting interesting DASD as those with the highest I/O intensity.

However, not all high I/O intensity volumes are interesting. A volume may have high I/O intensity owing to a high I/O rate, but it is not interesting unless its response time exceeds some threshold (for example 7ms). Thus, a volume may encounter Sa = 200 I/O operations a second, but if the response time of the volume is RTa = 2ms (owing to good caching), then this is *not* an interesting DASD volume, even though it has a high I/O intensity. Similarly, often we see volumes with very high response time (for example, RTa = 200ms), but the I/O rate is very low (Sa = 0.001 I/Os). This phenomenon occurs because of a monitor (RMF) sampling inadequacy, but clearly this volume is again *not* interesting.

Thus, we conclude that it may be worth while filtering out (exclude from examination) the volumes with response times below a threshold

(say RTa < 7ms), and those with low I/O rates (say Sa < 1 I/O per second). When this double filter is used, the pool of potentially interesting volumes is drastically reduced. Next, the reduced pool is sorted and examined for the volumes with the highest I/O intensity. The volumes so identified then probably qualify as the most interesting DASD, ie those that have the largest impact on the system performance and therefore the ones where tuning action should be performed. This fourth refined approach is the one recommended.

Once interesting DASD have been identified, further examination reveals the 'dominant' response time component leading to the high RTa. Once the component (eg, IOSQ or DISC) is identified then further analysis (and possible remedial action) becomes easier to identify.

CONCLUSIONS

A method for reducing excessive measured DASD data has been presented. This method involves ordering DASD according to their I/O intensity numbers after appropriate filters have been used to exclude volumes with spuriously high values. Once analysis is focused on only a handful of devices, selection of the dominant response time component for reduction should lead to a remedial action. Once the 'bad' response time component is identified, the selection of volume splitting or volume movement, or increased cache sizes, readily lend themselves as appropriate tuning solutions.

Thomas Beretvas
Beretvas Performance Consultants (USA)

Dynamically allocating files to system utilities

THE PROBLEM

One of the problems with using JCL and programs for which we have only the object module is that we cannot make dynamic decisions on dataset names at run-time. For instance, if we wish to run a batch utility and write the output to a dataset that contains the date as part of its name, there is no simple way to do this. If we have the source for the utility, we can change it to dynamically allocate a dataset with a name decided upon at run-time. This article provides some hints and tips on how to do it for those utilities for which we do not have the source.

One of the first things that comes to mind is to have an allocation program in a prior step. This step could accept parameters and allocate the file and then pass it to the next step that will use it. For example, let's say ALLOCPGM is the name of our program and that it will dynamically allocate a file to the parameter DDname and then pass it to the next step which runs the system utility which needs the file with the date and time as part of the name. Let's say the program also has the logic to be parameter-driven in its decision to make the date and time part of the dynamically allocated dataset name:

```
//STEP1    EXEC  PGM=ALLOCPGM,
//      PARM='DSN=ABC.DEF,DATE=Y,TIME=Y,DDNAME=PASSFILE'
//*
//STEP2    EXEC  PGM=utility
//OUTDD      DD  DSN=*.STEP1.PASSFILE,DISP=(OLD,KEEP),…
```

Unfortunately this will not work, for several reasons. First of all we will get a JCL error, because PASSFILE is not a DD-card in the first step; it was dynamically allocated. Even if we get creative and put a 'basic' DD-card into STEP1 and then modify the JFCB dynamically it will still not work: the JCL interpreter will not go and 'search' for the file name – referback only works for dataset names that are actually coded, and by coding the name in STEP1 we defeat the whole purpose of making up the name at run-time.

This type of dataset may be required by applications that run daily, like system logs, SMF dumps, etc. We will demonstrate a way to overcome

this problem and, more specifically, do it along the lines of the SMF dump program, IFASMFDP, because this brings some additional complications to the table. So, just to clearly state our intention: we wish to dump our SMF data to datasets with the name SMF.DUMP.Dyymmdd.Thhmmss. By having the dataset name built dynamically at the moment of execution, we get an additional benefit in the specific case of IFASMFDP. Most sites use dataset names that are GDG-based, even if it is unique for each SMF dump dataset. This often means that only one SMF dump task can run at any time. In high-activity production systems this could lead to a loss of SMF data if the dump programs cannot keep up with the generation of SMF data records. By dynamically building the dataset name at the moment the SMF dump task starts, we have unique dataset names. This means that, at least theoretically, any number of SMF dump tasks can run at the same time. If we are concerned that more than one SMF dump switch can occur at the same time, we can even make the fraction of a second a part of the name, for example SMF.DUMP.Dyymmdd.

Thhmmssx with x the 1/10th of the second.

A SOLUTION

A solution to this problem is to have a single program that does the following:

- Reads parameters and determines whether a dataset with the date and/or time as part of the name has to be allocated.
- Gets the date and time and then allocates the file dynamically.
- Invokes the utility.
- Once the utility completes, simply takes the return code, loads it into register 15 and branches back to MVS.

This is exactly what the author of this article did and it became a very useful allocation utility. The name of the program to invoke can also be accepted as a parameter. Another useful consideration is whether the utility requires the caller to be in Supervisor State. An example of parameters that the program may accept include:

```
//STEP1 EXEC PGM=ALLOCPGM,
//   PARM='DDNAME=ddname,UTILITY=utility,TIME=x,DATE=y,SUPSTATE=z,
//              DSNAME=hlq1.hlq2,PARAM=zzzz'
```

where

- DDNAME is the name of the DDname to allocate the file to.
- UTILITY is the name of the program to invoke once the allocation has been done.
- TIME=Y/N – with Y indicating we want the time as part of the dataset name.
- DATE=Y/N – with Y indicating we want the date as part of the dataset name.
- SUPSTATE=Y/N – indicates whether we should switch to Supervisor State before we call the utility.
- DSNAME= – contains one or more high-level qualifiers that are used to form the dataset name.
- PARAM= – contains the value that would have been passed to the utility if it had been run directly from JCL. This is required because system utilities often accept parameters from JCL.

The date and time have to be prefixed by a character because a dataset qualifier cannot start with numeric values. So, if we have:

```
//STEP1 EXEC PGM=ALLOCPGM,
//   'DDNAME=UTILITY=IFASMFDP,TIME=Y,DATE=Y,SUPSTATE=N,
//              DSNAME=SMF.DUMP'
```

we can quite easily write a program that will allocate a file by the name of SMF.DUMP.D991215.T1728144 and then call IFASMFDP in problem state. This becomes a fairly generic program we can use for any such utility.

None of the techniques required so far is really difficult and an abundance of examples in previous *MVS Update* articles describe how to accept parameters, change program state, dynamically allocate files, and LINK to routines.

We will now look at some interesting experiences the author had when this utility was used to dump SMF data. The program that was called

once the file was allocated to DUMPOUT was obviously IFASMFDP. Strangely enough it produced an error in cleaning the dump dataset if the program was called in supervisor state, so we had to stay in problem state. ALLOCPGM had to be called from an APF authorized library though.

Once we had the program working, most of our SMF dump problems were forever solved (well, so it seemed for a while anyway).

The next problem we encountered was dealing with x37 abends. IFASMFDP 'helps' by actually intercepting the abend. It gives an error message and then terminates with an RC=12. Not much help – the systems programmer would still get a page in the middle of the night because the SMF dump task failed. The last hurdle to overcome was to make the step recover from the x37 abend. SMS and other products can help in that regard nowadays, but sooner or later the time will come when there simply is not any space available on the system. We then decided to build enough intelligence into ALLOCPGM to switch to tape should the need arise. (To specifically cater for the exceptions IFASMFDP delivers we have a check in the program to see if UTILITY=IFASMFDP.) This then introduced a further parameter:

```
ALTUNIT=CART
```

which indicated that the data should be dumped to tape following an x37 abend. 'Telling' IFASMFDP to not intercept the x37 abend is not an easy task – there is no parameter to do this. We decided to do the following:

- Allocate the file.
- Load IFASMFDP into storage.
- Search IFASMFDP for the instructions that set the ESTAE and 'zap' them out in storage.
- Then invoke IFASMFDP.

One thing we discovered was that IFASMFDP left some storage lying around after the x37 abend. To overcome this, it was decided to ATTACH IFASMFDP rather than LINK to it. If it gets the x37 abend, we free the dataset and detach the subtask. MVS then cleans up the

storage and we are back to where we started. We then updated the dynamic allocation text unit, re-allocated the dataset and called, IFASMFDP again, this time dumping to tape.

Another benefit of ATTACHing IFASMFDP rather than LINKing to it is that we do not have to set up an ESTAE routine. If a program is invoked via a LINK and it then abends, the entire step will fail if there is no ESTAE routine to intercept the abend. When a program is ATTACHed, the completion code of the program is in the ECB that gets posted when the subtask terminates. These few lines of code show how the task is attached and how the completion code is analysed. Register 3 contains the address of the area where we have loaded IFASMFDP and register 2 contains the address of the field with the parameters we wish to pass to IFASMFDP:

```
PREPATCH EQU   *
         LA    R4,SUBTECB          Point to the subtask's ECB
         XC    SUBTECB,SUBTECB
         ATTACH EPLOC=(3),PARAM=((2)),ECB=(4),VL=1
         ST    R1,TASKTCB          Preserve the task's TCB address
         WAIT  ECB=SUBTECB         Wait for subtask to complete
         LA    R1,TASKTCB          Load the subtask's TCB address
         DETACH (1)                DETACH the subtask
         XR    R15,R15
         ICM   R15,7,SUBTECB+1     Pick up subtask's return code
         LTR   R15,R15             Zeros?
         BZ    SUCCESS             Yes, terminate normally
CODEANAL L     R2,SUBTECB          Pick up the ECB contents
         SRA   R2,12               Move 12 bytes to the right
         CLM   R2,1,=X'37'         Did we have an X37 type abend?
         BNE   PERCOLAT            No, go percolate the abend
         OI    X37FLAG,Yes         Yes, set flag
         B     RETRY               Go switch to tape
   etc…
```

We have not been called again to attend to a failing SMF dump task. Because the tape dataset is also catalogued it makes no difference to post-processing jobs. With the recent versions of OS/390, datasets on different media can be concatenated, so even if some of the dump datasets are on disk and others are on tape we do not have a problem. By virtue of the fact that the dump dataset names contain the date and time, we never have to sort SMF data – as long as we sort the dataset names for the post processing input.

The last part of our article is to show you how we managed to get rid of the ESTAE in IFASMFDP. We could obviously have zapped the module on disk, but this causes a maintenance problem – every time the module changes, the zap has to be reworked. We decided to stick to the module supplied by IBM and to scan it in storage in order to locate the ESTAE instructions. For this we had to do the following:

- Load the module to get its length. Note that MVS loads the module into a page-protected area where it cannot be altered, even in supervisor state.
- Delete the module after we have saved the length.
- Allocate enough storage to contain the module.
- Do a *directed* load into our allocated area. For the directed load we need to do the LOAD with a DCB parameter, meaning that we first had to dynamically allocate SYS1.LINKLIB.
- Scan the module for the ESTAE SVCs (X'0A3C'). Note that there are two occurrences of this SVC: one for when the ESTAE is set and one for when it is deleted.
- Give a message that we are removing the ESTAE SVCs.
- Replace both ESTAE SVCs with NOP instructions (X'0700').

This is a section of the program that removes the ESTAE SVCs from IFASMFDP:

```
*********************************************************************
* This routine loads IFASMFDP and zaps out the ESTAE SVC entries
*********************************************************************
IFASMFDP BAKR  R14,Ø
         LOAD  EP=IFASMFDP,ERRET=NOSMF
         LTR   R15,R15
         BNZ   NOSMF
         LR    R2,R1               Length of the module
         N     R2,=X'ØØØØFFFF'     Turn the first half of the word off
         MH    R2,=H'8'            length is in double words
         ST    R2,MODLENG
         STORAGE OBTAIN,LENGTH=(2),LOC=BELOW,BNDRY=DBLWD
         LTR   R15,R15
         BNZ   NOSMF
         LR    R2,R1               Where module will be loaded
         ST    R2,MODSTART
```

```
         DELETE EP=IFASMFDP        Delete loaded copy
         LTR   R15,R15
         BZ    OPENLINK
         LR    R2,R15
         WTO   'ALLOCPGM(E): -Module not deleted',ROUTCDE=11
         LR    R15,R2
         B     NOSMF
OPENLINK BAS   R14,ALOCLINK        Dynamically allocate SYS1LINKLIB
                                   (Not shown here, normal DYNALLOC)
         OPEN  LINKLIB
         LOAD  EP=IFASMFDP,ADDR=(R2),DCB=LINKLIB,ERRET=NOSMF
         LTR   R15,R15             Directed load to private storage
         BNZ   NOSMF
         ST    RØ,MODEP            IFASMFDP's entry point
         L     R4,MODLENG
         BCTR  R4,Ø                Reduce length by 1
         L     R2,MODSTART
         XR    R5,R5               Count # of ESTAE macros
ESTAELP  EQU   *                   Look for ESTAE SVC
         CLC   Ø(2,R2),=X'ØA3C'     (SVC 6Ø)
         BNE   NEXTCHAR
INTEGRTY LA    R5,1(R5)            Bump up zap-counter (must be 2)
         LR    R6,R2               Where we found the ESTAE
         SH    R6,=H'4'            Move 4 bytes back
         CLC   Ø(2,R6),=X'41ØØ'    Must contain "LA RØ," instruction
         BNE   ESTAEERR            Suspect, needs inspection
         MVC   Ø(2,R2),=X'Ø7ØØ'    Replace with NOP
         LA    R2,1(R2)
         BCTR  R4,Ø                Reduce loop counter by 1
NEXTCHAR LA    R2,1(R2)            Bump up "from" pointer
         BCT   R4,ESTAELP          Do for each byte
         CH    R5,=H'2'            Must be exactly 2 ESTAE macros
         BE    ESTAEWTO            There are 2
ESTAEERR WTO   'ALLOCPGM(E): -Unexpected ESTAE format in IFASMFDP,ESTAX
               E macros cannot be reliably removed',ROUTCDE=11
         ABEND ØØØ9
ESTAEWTO WTO   'ALLOCPGM(I): -IFASMFDP ESTAE removed',ROUTCDE=11
         B     IFASMFDX
IFASMFDX PR
NOSMF    ABEND ØØ1Ø,DUMP
```

Tommy Afbeen
Brandenburg Consulting (South Africa)

Testing the catalog search interface

INTRODUCTION

Below is a PL/I program called TSTCSI which invokes the IBM catalog search interface module IGGCSI00. IBM provides sample programs in the *DFSMS/MVS Managing Catalogs* book (form SC26-4914) to show the invocation of IGGCSI00 in the Assembler and REXX environments, but for no other languages.

This sample allows the user to pass certain parameters to it, which are then used by the IGGCSI00 interface to query catalogs and return the requested information. The information available to be passed to TSTCSI includes the dataset filter to be used (CSIFILTK), the optional specific catalog name to search (CSICATNM), the size of the work area to be used by IGGCSI00 to return information (CSIUSRLN), an option to perform tracing (TRCLVL), and an option flag to indicate whether only one catalog should be searched (CSIS1CAT, which would either be blank to search all catalogs or Y to search only the catalog specified in CSICATNM). A small Assembler routine is also included, called HVPC2X, which is used to convert the return and reason code fields returned by IGGCSI00 to a printable hexadecimal format.

TSTCSI

```
//TSHVRA JOB (),'TSTCSI',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
//PLI    EXEC IEL1CLG,
//         REGION=5000K,
//         PARM.PLI='OFFSET,INCLUDE,NODECK,LIST,AGGREGATE,ATTRIBUTES',
//         PARM.LKED='XREF,LIST,RENT,AMODE=31,RMODE=ANY',
// PARM.GO='/CSIUSRLN(32000) CSIFILTK(TSHVR.BROL.**) TRCLVL(5000)      X
//              CSICATNM(),CSIS1CAT( )'
//PLI.SYSLIB DD DSN=TSHVR.SOURCE.TEST,DISP=SHR
//           DD DSN=TSHVR.SOURCE.PROD,DISP=SHR
//PLI.SYSIN    DD *
* PROCESS LANGLVL(OS,SPROG);
* PROCESS SYSTEM(MVS);
 /* TSTCSI CATALOG SEARCH INTERFACE CSI TEST  IGGCSI00              */
 /* COMPLETE LIST OF FIELDS YOU CAN USE ARE IN:                     */
 /*  MOD:SYS4A.AOSA0(IGG0CLFE) LMOD:SYS4.LINKLIB(IGG0CLX0          */
 TSTCSI: PROC(PARMS) OPTIONS(MAIN NOEXECOPS REENTRANT) ;
 /* FUNCTION:                                                      */
```

```
/*                                                                  */
/* INSTALLATION..                                                   */
/* INPUT..                                                          */
/* OUTPUT..                                                         */
/* MAINTENANCE MODIFICATIONS CHANGES BEGIN */
/* MAINTENANCE MODIFICATIONS CHANGES END   */
/* USAGE NOTES BEGIN:*/
/* USAGE NOTES END---*/
/* COMPILE/LINK NOTES:*/
/* ABENDS:*/
/* DOCUMENTATION LITERATURE:*/
/* SC26-4914-Ø3  DFSMS/MVS V1R4 MANAGING CATALOGS   DGT1C1Ø4  */
/* CHANGING VALUES IN CATALOGS:                                     */
/*  SEE SYS4.HELP(ALTER) FOR CCSID                                  */
/****************************/
/* TEST MOGELIJKHEDEN      */
/****************************/
/**********************************/
/* PREPROCESSOR PRECOMPILER MACRO */
/**********************************/
/****************************/
/* RETURN CODES            */
/****************************/
DCL MYRC BIN FIXED(31) INIT(Ø);
/****************************/
/* CONSTANTS               */
/****************************/
DCL   $TRUE  BIT(1) STATIC INIT('1'B);
DCL   $FALSE BIT(1) STATIC INIT('Ø'B);
DCL   $ERROR   CHAR(8) STATIC INIT('*ERROR*');
DCL   $DEBUG CHAR(8)  STATIC INIT('*DEBUG*');
DCL   $INFO  CHAR(8)  STATIC INIT('*INFO *');
DCL   $PROG CHAR(8) STATIC INIT('TSTCSI');
DCL   $HOMELIB CHAR(44) STATIC INIT('TSHVR.SOURCE.TEST');
/**/
DCL $FENTERR  STATIC CHAR(4) INIT('C6E2Ø464'X);
                       /*FIELD ENTRY ERROR CODE FROM CAT*/
DCL $CTLGINV  STATIC CHAR(4) INIT('C1C878Ø4'X);/*INVALID CTLG?*/
DCL $CSINOENT STATIC BIT(1) INIT('1'B);

/*CF=CHAR FIXED CV=CHAR VARYING BF15=BINARY*/
DCL $CF CHAR(4) STATIC INIT('CF');
DCL $CV CHAR(4) STATIC INIT('CV');
DCL $BF15 CHAR(4) STATIC INIT('BF15');
DCL $FLDLINES(13) STATIC CHAR(16) INIT(
'ACCOUNT CF  ØØ32',
'CCSID   BF15ØØØ2',
'DATACLAS****ØØØØ',
'DSCRDT2 ****ØØØØ',
'FILEDATA****ØØØØ',
'FILESEQ ****ØØØØ',
'NVSMATTR****ØØØØ',
```

```
'OWNERID ****0000',
'SECFLAGS****0000',
'STORCLASCV  0000',
'UDATASIZ****0000',
'USERAREC****0000',
'VOLSER  CF  0006');
DCL 1 $FLDSTRUC(13) BASED(ADDR($FLDLINES)) UNALIGNED,
    2 $FLDN CHAR(8),
    2 $FLDT CHAR(4),
    2 $FLDL PIC'9999';

/***************************/
/* EXTERNAL ENTRIES  ENTRY */
/***************************/
DCL IGGCSI00 EXTERNAL  ENTRY OPTIONS(INTER,ASM,RETCODE);
/* HVPC2X:CONVERT CHAR TO HEX */
DCL HVPC2X             ENTRY OPTIONS(INTER,ASM,RETCODE);

/***************************/
/* PASSED PARM PARMS PRM   */
/***************************/
DCL PARMS CHAR(*) VARYING;
/***************************/
/* BUILTIN                 */
/***************************/
/*EXPLICIT DECLARATIONS OF PL/I BUILTIN FUNCTIONS*/
 DCL (ABS,ADDR,ALLOCATION,BINARY,BIT,CEIL,CHAR,CSTG,
      DATE,DATETIME,DECIMAL,DIM,ENTRYADDR,
      COMPLETION,STATUS,
      LBOUND,HBOUND,HIGH,INDEX,
      LENGTH,LOW,MAX,MIN,
      MOD,NULL,ONCHAR,ONCODE,ONLOC,ONSOURCE,
      PLIDUMP,PLIRETC,PLIRETV,REPEAT,ROUND,SAMEKEY,
      STG,STRING,SUBSTR,TIME,TRANSLATE,TRUNC,UNSPEC,VERIFY) BUILTIN;
DCL (POINTERVALUE,BINARYVALUE,POINTERADD,SYSNULL) BUILTIN;
/***************************/
/* FILES                   */
/***************************/
DCL SYSPRINT FILE STREAM OUTPUT PRINT ;
/***************************/
/* WORK        VARIABLES   */
/***************************/
DCL TRCLVL BIN FIXED(31);
DCL DTTM CHAR(17) INIT('');
DCL KARS CHAR(2048) BASED;
DCL KARSV CHAR(2048) VARYING BASED;
DCL BF15 BIN FIXED(15) BASED;
DCL LOPER BIN FIXED(15);
/*IGG VARIABLES*/
DCL 1 IGGWA UNALIGNED,
 2 CSIRWORK,
  3 CSIUSRLN BIN FIXED(31),
```

```
  3 CSIREQLN BIN FIXED(31),
  3 CSIUSDLN BIN FIXED(31),
  3 CSINUMFD BIN FIXED(15),
 2 CSIRCAT,
  3 CSICFLG        ,
    4 CSINTICF BIT(1),
    4 CSINOENT BIT(1),
    4 CSINTCMP BIT(1),
    4 CSICERR  BIT(1),
    4 CSICERRP BIT(1),
    4 RESERVED BIT(3),
  3 CSICTYPE CHAR(1),
  3 CSICNAME CHAR(44),
  3 CDS,
   4 CSIRETM  CHAR(2),
   4 CSIRETR  CHAR(1),
   4 CSIRETC  CHAR(1),
 2 ENTRIES  CHAR(32767);
/* IGGCDS RETURN AREA CODES */
DCL Ø1 IGGCDS UNALIGNED,
 Ø2 MODID CHAR(2),
 Ø2 RSNCD BIT(8),
 Ø2 RTNCD BIT(8);
DCL IGGCDS_CHAR CHAR(4) BASED(ADDR(IGGCDS));
/* SYS4.MACLIB(IGGCSINP) */
/* SELECTION_CRITERIA_LIST */
DECLARE
 1 CSIFIELD     UNALIGNED,
  2 CSIFILTK    CHAR(44),      /* FILTER   KEY                     */
  2 CSICATNM    CHAR(44),      /* CATALOG NAME OR BLANKS           */
  2 CSIRESNM    CHAR(44),      /* RESUME NAME OR BLANKS            */
  2 CSIDTYPS            ,      /* ENTRY TYPES                      */
    3 CSISTYPS(16) CHAR (1),   /* SUBSCRIPT CSIDTYPS               */
  2 CSIOPTS,                   /* CSI OPTIONS                      */
   3 CSICLDI   CHAR(1) INIT('Y'),/*RETURN D&I IF C A MATCH Y OR BLNK*/
   3 CSIRESUM  CHAR(1) INIT(' '), /*RESUME FLAG Y OR BLANK*/
   3 CSIS1CAT  CHAR(1),       /* SEARCH CATALOG      Y OR BLANK   */
   3 CSIRESRV  CHAR(1),       /* RESERVED                         */
  2 CSINUMEN  BIN FIXED(15),  /* NUMBER OF ENTRIES FOLLOWING      */
  /* THERE IS A LIMIT OF 1ØØ FIELD NAMES PER INVOCATION OF CSI */
  2 CSIFLDNM(1ØØ) CHAR(8);
/**/
DCL ENTRY@ POINTER;
/* 1 + 1 + 44  + 4 + CSITOTLN - 4 ??? */
DCL 1 ENTRY UNALIGNED BASED(ENTRY@),
 2 CSIEFLAG,
  3 CSIPMENT BIT(1),/*1=PRIMARY ENTRY,Ø=PRECEDING ENTRY*/
  3 CSIENTER BIT(1),
  3 CSIEDATA BIT(1),
  3 RESERVED BIT(5),
 2  CSIETYPE CHAR(1),
 /*********
```

```
 A NON-VSAM DATASET
 B GENERATION DATA
 C CLUSTER
 D DATA COMPONENT
 G ALTERNATE INDEX
 H GENERATION DATASET
 I INDEX COMPONENT
 R PATH
 X ALIAS
 U USER CATALOG
 L ATL LIBRARY ENTRY
 W ATL VOLUME ENTRY
 ***************/
 2 CSIENAME CHAR(44),
 2 CSIERETN CHAR(4);
 /* FIRST LEN FIELD FOLLOWS  */
 /* CSILENF1 BIN FIXED(15) */
/**/
DCL 1 ENTRY_CDS BASED(ADDR(ENTRY.CSIERETN)),
   2 CSIRETM  CHAR(2),
   2 CSIRETR  CHAR(1),
   2 CSIRETC  CHAR(1);
DCL 1 ENTRY_DATA BASED(ADDR(ENTRY.CSIERETN)),
  2  CSITOTLN BIN FIXED(15),
  2  RESERVED CHAR(2);
DCL TOTLN BIN FIXED(31);
DCL IGGWA_END@ POINTER;
DCL CTLGNM CHAR(44) INIT('');
DCL FLDLN@ POINTER;
DCL FLD@   POINTER;
DCL FLD_LOPER BIN FIXED(15);
DCL 1 OUTREC,
 2 TYPE  CHAR(1),
 2 NAME CHAR(44),
 2 CTLGNM CHAR(44),
 2 ACCOUNT CHAR(32);
/***************************/
/* CONDITIONS              */
/***************************/
 ON ERROR BEGIN;
   ON ERROR SYSTEM;
   DTTM=DATETIME();
   PUT SKIP EDIT(DTTM,$PROG,$ERROR,ONCODE,'RAISED IN',ONLOC)
               (3(X(1),A),X(1),F(4),X(1),A,X(1),A);
   IF ONCODE=9 THEN DO; /* SIGNAL ERROR */
    PUT SKIP EDIT('CHECK PRECEDING MESSAGES!')
                (X(1),A);
   END;
 END;/*ON ERROR*/
/***************************/
/*       MAIN              */
/***************************/
```

```
IGGWA='';
CALL PARMS_GET;
/**/
FETCH IGGCSIØØ;
/**/
IGGCDS_CHAR='ØØØØØØØØ'X;
CSISTYPS='';
CSICLDI='Y';
CSIRESUM='';
DO LOPER=1 TO DIM($FLDLINES,1);
 CSIFLDNM(LOPER)=$FLDSTRUC.$FLDN(LOPER);
END;
CSINUMEN=DIM($FLDLINES,1);
DO UNTIL(CSIRESUM ¬= 'Y');
 /**/
 DTTM=DATETIME();
 PUT SKIP EDIT(DTTM,$PROG,$INFO,'CALLING IGGCSIØØ')
               (A,X(1));
 /**/
 CALL IGGCSIØØ(IGGCDS,CSIFIELD,IGGWA);
 MYRC=PLIRETV();
 IF MYRC¬=Ø THEN DO;
  PUT SKIP DATA(MYRC);
  CALL IGGCDS_LIST(IGGCDS_CHAR);
  PUT SKIP DATA(CSIFIELD);
  SIGNAL ERROR;
 END;
 ELSE DO;
  IF IGGCDS_CHAR¬='ØØØØØØØØ'X THEN DO;
   PUT SKIP LIST('IGGCDS IS NOT ALL NULL!');
   CALL IGGCDS_LIST(IGGCDS_CHAR);
   SIGNAL ERROR;
  END;
  IF IGGWA.CSIRWORK.CSINUMFD ¬= CSINUMEN + 1  THEN DO;
   PUT SKIP LIST('REMARKABLE FIELD:');
   PUT SKIP DATA(IGGWA.CSIRWORK.CSINUMFD);
   SIGNAL ERROR;
  END;
  IF CTLGNM ¬= CSICNAME THEN DO;
   CTLGNM=CSICNAME;
   PUT SKIP EDIT('NEW CTLG:',CTLGNM)(A);
  END;
  IF IGGWA.CSIRWORK.CSIREQLN >=CSIUSRLN  THEN DO;
   PUT SKIP LIST('CSIUSRLN IS TOO SHORT!');
   PUT SKIP LIST(' CSIUSRLN SHOULD BE AT LEAST CSIREQLN!');
   PUT SKIP LIST(' IF NOT ABENDED, PROGRAM WILL LOOP!');
   PUT SKIP DATA(CSIRWORK);
   PUT SKIP DATA(CSIRCAT);
   SIGNAL ERROR;
  END;
  IF IGGWA.CSIRCAT.CSICFLG.CSINOENT THEN DO;
   PUT SKIP LIST('NO ENTRIES');
```

```
END;/*IF IGGWA.CSIRCAT.CSICFLG.CSINOENT*/
ELSE DO;
 IF CSIRESUM='Y' THEN DO;
  IF CSICATNM ¬= CSICNAME  THEN DO;
   PUT SKIP LIST('REMARKABLE SITUATION:');
   PUT SKIP LIST(' SWITCH TO OTHER CATALOG');
  END;
 END;
 IF TRCLVL > 4Ø96 THEN DO;
  PUT SKIP LIST('IGGWA:');
  MYRC=HVPDMPX(ADDR(IGGWA),BINARY(CSIUSDLN,31),SYSPRINT);
 END;
 IGGWA_END@=POINTERADD(ADDR(IGGWA),IGGWA.CSIRWORK.CSIUSDLN);
 ENTRY@=ADDR(IGGWA.ENTRIES);
 DO WHILE(ENTRY@ < IGGWA_END@);
  IF ¬CSIENTER THEN DO;
   IF ENTRY.CSIETYPE='Ø' THEN DO;
    TOTLN=4;/*CTLG*/
    CTLGNM=ENTRY.CSIENAME;
    PUT SKIP EDIT('NEW CTLG:',CTLGNM)(A);
   END;/*IF ENTRY.CSIETYPE='Ø'*/
   ELSE DO; /*IF ENTRY.CSIETYPE¬='Ø'*/
    TOTLN=ENTRY_DATA.CSITOTLN;
    IF TRCLVL > 4Ø96 THEN
     PUT SKIP EDIT('ENTRY_DATA.CSITOTLN=',
                   ENTRY_DATA.CSITOTLN)(A,F(5));
    OUTREC.TYPE=ENTRY.CSIETYPE;
    OUTREC.NAME=ENTRY.CSIENAME;
    OUTREC.CTLGNM=CTLGNM;
    /**/
    FLDLN@=POINTERADD(ENTRY@,CSTG(ENTRY));/*5Ø*/
    FLD@  =POINTERADD(FLDLN@,(CSINUMFD-1)*2);
    DO FLD_LOPER=1 TO  (CSINUMFD-1);
     IF FLDLN@->BF15<=Ø THEN DO;
      /*USERAREC FLDLEN=-1 FOR ALIAS*/
      IF FLDLN@->BF15<Ø THEN DO;
       IF $FLDSTRUC.$FLDN(FLD_LOPER)^='USERAREC' THEN DO;
        PUT SKIP LIST('REMARKABLE:');
        PUT SKIP LIST(' FLDLN@->BF15<=Ø');
        PUT SKIP EDIT(' FIELD=',FLD_LOPER,$FLDSTRUC.$FLDN(FLD_LOPER),
        ' FLDLEN=',FLDLN@->BF15)
        (A,F(4),X(1),A,A,F(6));
       END;
      END;
     END;/*IF FLDLN@->BF15<=Ø*/
     /*IF FLDLN@->BF15>Ø*/
     ELSE                    DO;
      IF FLD_LOPER=1 THEN OUTREC.ACCOUNT=
       SUBSTR(FLD@->KARS,1,FLDLN@->BF15);
      IF TRCLVL > 128  THEN DO;
      /**/
      IF $FLDSTRUC.$FLDT(FLD_LOPER)=$CV  THEN DO;
```

```
          PUT SKIP EDIT(
          '-FIELD ',$FLDSTRUC.$FLDN(FLD_LOPER),':',FLD@->KARSV)(A);
         END;/*$CV*/
         ELSE IF $FLDSTRUC.$FLDT(FLD_LOPER)=$CF  THEN DO;
          PUT SKIP EDIT(
          '-FIELD ',$FLDSTRUC.$FLDN(FLD_LOPER),':',
          SUBSTR(FLD@->KARS,1,FLDLN@->BF15))(A);
         END;/*$CF*/
         ELSE IF $FLDSTRUC.$FLDT(FLD_LOPER)=$BF15 THEN DO;
          PUT SKIP EDIT(
          '-FIELD ',$FLDSTRUC.$FLDN(FLD_LOPER),':',
          FLD@->BF15)(3(A),F(5));
         END;/*$BF15*/
         ELSE DO;
          PUT SKIP EDIT('FIELD=',FLD_LOPER,$FLDSTRUC.$FLDN(FLD_LOPER),
          ' FLDLEN=',FLDLN@->BF15)
          (A,F(4),X(1),A,A,F(6));
          PUT SKIP EDIT('*',SUBSTR(FLD@->KARS,1,FLDLN@->BF15),'*')(A);
         END;/*ELSE*/
         END;/*IF TRCLVL > 128*/
         FLD@=POINTERADD(FLD@,FLDLN@->BF15);
        END;/*IF FLDLN@->BF15>Ø*/
        FLDLN@=POINTERADD(FLDLN@,Ø2);
       END;/*DO FLD_LOPER=1 TO  (CSINUMFD-1)*/
       /**/
       PUT SKIP EDIT('*',OUTREC)(A);
      END;/*IF ENTRY.CSIETYPE^='Ø'*/
     END;/* NOT CSIENTER*/
     ELSE DO;
      TOTLN=4;
      PUT SKIP LIST('ENTRY WITH CSIENTER:');
      PUT SKIP LIST(ENTRY);
      PUT SKIP LIST('IGGWA:');
      PUT SKIP LIST(
       SUBSTR(ADDR(IGGWA)->KARS,1,CSIUSDLN));
      IF ENTRY.CSIETYPE='Ø' THEN ;
      ELSE SIGNAL ERROR;
     END; /*CSIENTER*/
     ENTRY@=POINTERADD(ENTRY@,CSTG(ENTRY)+TOTLN-4);
    END;/*DO WHILE(ENTRY@ < IGGWA_END@)*/
   END;/*IF IGGWA.CSIRCAT.CSICFLG.CSINOENT*/
  END;/*MYRC=Ø ON CALL IGGCSIØØ*/
END;/*DO UNTIL(CSIRESUM ¬= 'Y'*/
/***************************/
/* END                     */
/***************************/
L_RETURN:
ON ERROR SYSTEM;  /* AVOID ERROR LOOP */
CALL PLIRETC(MYRC);
RETURN;
/***************************/
/* SUBROUTINES             */
```

```
/**************************/
PARMS_GET:PROC;
  DCL (X,Y) BIN FIXED;
  DCL CSIUSRLN_PIC PIC'999999';
  DCL TRCLVL_PIC   PIC'999999';
  DCL $CSIUSRLN CHAR(9) INIT('CSIUSRLN(');
  DCL $CSIFILTK CHAR(9) INIT('CSIFILTK(');
  DCL $CSICATNM CHAR(9) INIT('CSICATNM(');
  DCL $CSIS1CAT CHAR(9) INIT('CSIS1CAT(');
  DCL $TRCLVL   CHAR(7) INIT('TRCLVL(');
  CSIUSRLN=1Ø24;
  X=INDEX(PARMS,$CSIUSRLN);
  IF X>Ø THEN DO;
   X=X+LENGTH($CSIUSRLN);
   DO Y=X TO LENGTH(PARMS);
    IF SUBSTR(PARMS,Y,1)=')' THEN LEAVE;
   END;
   /* Y = 1 TOO FAR */
   CSIUSRLN_PIC=SUBSTR(PARMS,X,Y-X);
   CSIUSRLN=CSIUSRLN_PIC;
  END;/*$CSIUSRLN*/
  PUT SKIP DATA(CSIUSRLN);
  CSIFILTK='**';
  X=INDEX(PARMS,$CSIFILTK);
  IF X>Ø THEN DO;
   X=X+LENGTH($CSIFILTK);
   DO Y=X TO LENGTH(PARMS);
    IF SUBSTR(PARMS,Y,1)=')' THEN LEAVE;
   END;
   /* Y = 1 TOO FAR */
   CSIFILTK=SUBSTR(PARMS,X,Y-X);
  END;/*$CSIFILTK*/
  PUT SKIP DATA(CSIFILTK);
  TRCLVL=Ø;
  X=INDEX(PARMS,$TRCLVL);
  IF X>Ø THEN DO;
   X=X+LENGTH($TRCLVL);
   DO Y=X TO LENGTH(PARMS);
    IF SUBSTR(PARMS,Y,1)=')' THEN LEAVE;
   END;
   /* Y = 1 TOO FAR */
   TRCLVL_PIC=SUBSTR(PARMS,X,Y-X);
   TRCLVL=TRCLVL_PIC;
  END;/*$TRCLVL*/
  PUT SKIP DATA(TRCLVL);
  CSICATNM='';/*BEGIN WITH MCAT*/
  X=INDEX(PARMS,$CSICATNM);
  IF X>Ø THEN DO;
   X=X+LENGTH($CSICATNM);
   DO Y=X TO LENGTH(PARMS);
    IF SUBSTR(PARMS,Y,1)=')' THEN LEAVE;
   END;
```

```
   /* Y = 1 TOO FAR */
   CSICATNM=SUBSTR(PARMS,X,Y-X);
  END;/*$CSICATNM*/
  PUT SKIP DATA(CSICATNM);
  CSIS1CAT='';/*BEGIN WITH MCAT*/
  X=INDEX(PARMS,$CSIS1CAT);
  IF X>Ø THEN DO;
   X=X+LENGTH($CSIS1CAT);
   DO Y=X TO LENGTH(PARMS);
    IF SUBSTR(PARMS,Y,1)=')' THEN LEAVE;
   END;
   /* Y = 1 TOO FAR */
   CSIS1CAT=SUBSTR(PARMS,X,Y-X);
  END;/*$CSIS1CAT*/
  PUT SKIP DATA(CSIS1CAT);
END PARMS_GET;
/**/
IGGCDS_LIST:PROC(IGGCDS_CHAR);
 DCL IGGCDS_CHAR CHAR(4);

 DCL Ø1 IGGCDS UNALIGNED BASED(ADDR(IGGCDS_CHAR)),
  Ø2 MODID CHAR(2),
  Ø2 RSNCD BIT(8),
  Ø2 RTNCD BIT(8);
 DCL CHAR8 CHAR(8);

 CALL HVPC2X(IGGCDS_CHAR,CHAR8,CSTG(IGGCDS_CHAR));
 PUT SKIP EDIT('IGGCDS HEX:',CHAR8)(A);
END IGGCDS_LIST;
/**/
/*LANGUAGE(PLI) SYSTEM(MVS) DB2(NO) MAIN(NO) */
/*HVPDMPX DUMP HEX */
HVPDMPX:PROC(MSGPTR,MSGLEN,MSGFILE) OPTIONS(REENTRANT)
       RETURNS(BIN FIXED(31));
  DCL MSGPTR POINTER;
  DCL MSGLEN BIN FIXED(31);
  DCL MSGFILE FILE ;

  DCL HVPC2X    ENTRY OPTIONS(INTER,ASM,RETCODE);

  DCL MYRC   BIN FIXED(31) INIT(Ø);
  DCL HEXBUFFER CHAR(32);
  DCL HEXLOPER BIN FIXED(31);
  DCL HEXAANTAL BIN FIXED(31);
  DCL OFFSET_HEX CHAR(8);
  DCL OVERLAYC9999 CHAR(9999) BASED;

   HEXLOPER=1;
   DO WHILE(HEXLOPER<=MSGLEN);
    HEXAANTAL=MSGLEN-HEXLOPER+1;
    IF HEXAANTAL>16 THEN HEXAANTAL=16;
    CALL HVPC2X(SUBSTR(MSGPTR->OVERLAYC9999,HEXLOPER,HEXAANTAL),
```

```
                HEXBUFFER,
                HEXAANTAL);
     CALL HVPC2X(HEXLOPER-1,
                OFFSET_HEX,
                BINARY(4,31));
     PUT FILE(MSGFILE)
         SKIP EDIT(OFFSET_HEX,SUBSTR(HEXBUFFER,1,HEXAANTAL*2),
                   SUBSTR(MSGPTR->OVERLAYC9999,HEXLOPER,HEXAANTAL))
                   (A(8),X(1),A(32),X(1),A(16));
     HEXLOPER=HEXLOPER+HEXAANTAL;
    END;/*DO WHILE(HEXLOPER<=MSGLEN)*/
  HVPDMPX_END:
  RETURN(MYRC);
 END HVPDMPX;
 END TSTCSI ;
/*
//LKED.TOOLSMOD DD DISP=SHR,DSN=TSHVR.OBJMOD.MVS.TEST
//              DD DISP=SHR,DSN=TSHVR.OBJMOD.MVS.PROD
//LKED.SYSIN   DD *
 INCLUDE TOOLSMOD(HVPC2X)
/*
```

HVPC2X

```
* HVPC2X : CONVERT TO HEX
*
* HOW TO CALL:
* ASM:
*        CALL  HVPC2X,(GA@,ERRMSG+GWAHEX-INFMSG1,                    *
*              $FW4),VL,MF=(E,CALLLIST3)
* PLI:
*  DCL HVPC2X ENTRY OPTIONS(INTER,ASM,RETCODE);
*   CALL HVPC2X(EIBFN,FN_HEX,CSTG(EIBFN));
*               IN,OUT,INLEN
*   CALL HVPC2X(AMBLP,PHEX,BINARY(4,31));
*
* DOEL  : AANTAL BYTES CONVERTEREN NAAR HEXADECIMALE VOORSTELLING
* INPUT : RØ1->PARAMETERLIST
* OUTPUT: HEXADECIMALE VOORSTELLING
*
* REGISTER USAGE   :
* R13   : ADRESS OF SAVE AREA FROM CALLER
* R14   : RETURN ADRESS TO CALLER
* R15   : ENTRY ADRESS OF CALLED HVPC2X / RETURN-CODE
* EQU'S
         COPY HVREGS     REGISTER EQUATES
DFZØ1Ø4Ø
R_PARM   EQU RØ1
R_BASE   EQU RØ2
R_LEN    EQU RØ3           =  HIGH(LEN/4)
```

```
WORKØ1   EQU RØ4
WORKØ2   EQU RØ5
R_NIBBLE_IX EQU RØ6
R_MOD4    EQU RØ7      R_MOD4= MOD(LEN/4)
R_MASK    EQU RØ8
R_INPUT  EQU RØ9
WORKØ3   EQU R1Ø
R_OUTP_IX EQU R11
*
*
DFZØ372Ø
PARMLIST DSECT
INPUT@    DS A        @ TO A
OUTPUT@   DS A        @ TO A
LENGTE@   DS A        @ TO F
NEXT      DS ØA
*
HVPC2X CSECT
HVPC2X AMODE 31
HVPC2X RMODE ANY
         SAVE   (14,12),,HVPC2X.&SYSTIME..&SYSDATE
DFZØ442Ø
         BALR   R_BASE,Ø                    ESTABLISH BASE
DFZØ444Ø
         USING  *,R_BASE
DFZØ446Ø
*
         USING PARMLIST,R_PARM
L_NEXT   EQU   *
         ICM   R_OUTP_IX,B'1111',OUTPUT@
         ICM   R_INPUT,B'1111',INPUT@
         ICM   R_LEN,B'1111',LENGTE@ ADRES
         ICM   R_LEN,B'1111',Ø(R_LEN) LENGTE ZELF
*
         IC    R_MASK,MASK4   MOVED OUT OF LOOP TO HERE
         LR    R_MOD4,R_LEN
         SLL   R_MOD4,3Ø(Ø)
         SRL   R_MOD4,3Ø(Ø) ONTHOUDT 2 LAATSTE BITS
         SRL   R_LEN,2(Ø)      R_LEN=R_LEN/4
         LTR   R_MOD4,R_MOD4
         BZ    L_MAINLOOP MULTIPLE OF 4
         LA    R_LEN,1(R_LEN)  R_LEN = HIGH(LEN/4)
L_MAINLOOP EQU *
*        R_LEN=HIGH(LEN/4)
**       IC    R_MASK,MASK4   MOVED OUT OF LOOP
         LA    R_NIBBLE_IX,8   4 BYTES 8 NIBBLES
         CH    R_LEN,=H'1' LAATSTE RONDE
         BNE   CONT1
         LTR   R_MOD4,R_MOD4
         BZ    CONT1
         LR    R_NIBBLE_IX,R_MOD4
         SLL   R_NIBBLE_IX,1(Ø)  1 NIBBLE = 2 NIBBLES OUTPUT
```

```
         LA    WORKØ3,MASK1
         SH    R_MOD4,=H'1'
         IC    R_MASK,Ø(R_MOD4,WORKØ3) POINTER NAAR GOEDE MASKER
CONT1    EQU   *
         XR    WORKØ2,WORKØ2
         EX    R_MASK,ICM
L_LOOP_1_4 EQU *    MAX 4 NIBBLES TO MAX 8 CHARS
         XR    WORKØ1,WORKØ1
*        IC    WORKØ1,C_ØF
         SLDL  WORKØ1,4(Ø)      4 BITS FROM WORKØ2 TO WORKØ1
         CH    WORKØ1,=H'9'
         BH    L_A_F
         LA    WORKØ3,X'FØ'
         B     L_STORE
L_A_F    EQU   *
         SH    WORKØ1,=H'9'
         LA    WORKØ3,X'CØ'
L_STORE  EQU   *
         OR    WORKØ1,WORKØ3
*        WORKØ1 CONTAINS NOW B'1111????'   B'11ØØ????'
         STC   WORKØ1,Ø(R_OUTP_IX)
         LA    R_OUTP_IX,1(Ø,R_OUTP_IX)
         BCT   R_NIBBLE_IX,L_LOOP_1_4
         LA    R_INPUT,4(R_INPUT)
         BCT   R_LEN,L_MAINLOOP
*
         L     WORKØ1,LENGTE@
         LTR   WORKØ1,WORKØ1
         BM    RETURN              LAST PARM
         LA    R_PARM,NEXT
         B     L_NEXT
RETURN   EQU   *
DFZ4326Ø
         RETURN (14,12),RC=Ø
DFZ4332Ø
*  EXECUTED INSTRUCTION
ICM      ICM   WORKØ2,B'ØØØØ',Ø(R_INPUT)
         DS    ØF
MASK1    DC    B'ØØØØ1ØØØ'   ORED WITH INSTRUCTION
MASK2    DC    B'ØØØØ11ØØ'   ORED WITH INSTRUCTION
MASK3    DC    B'ØØØØ111Ø'   ORED WITH INSTRUCTION
MASK4    DC    B'ØØØØ1111'   ORED WITH INSTRUCTION
         LTORG
         END
```

Herman Vierendeels
Systems Programmer (Belgium)

Using STRING and UNSTRING in COBOL

INTRODUCTION

Did you ever want to concatenate several alphanumeric fields together to create a composite field? STRING will save you a lot of actions and redefinition. It will also save programming loops if the data in question varies in length (ever try to concatenate first and last name fields to get it to print out correctly?).

THE OLD WAY

In the COBOL I world, we would have accomplished the task with code similar to this:

```
DATA DIVISION.
     Ø5 FIRST-NAME             PIC X(2Ø).
     Ø5 FN-TBL REDEFINES FIRST-NAME.
       1Ø FIRSTN-CHAR     OCCURS 2Ø TIMES
                               PIC X.

     Ø5 LAST-NAME              PIC X(2Ø).
     Ø5 LN-TBL REDEFINES LAST-NAME.
       1Ø LASTN-CHAR      OCCURS 2Ø TIMES
                               PIC X.

     Ø5 FULL-NAME              PIC X(3Ø).
     Ø5 FULLN-TBL REDEFINES FIRST-NAME.
       1Ø FULLNAME-CHAR   OCCURS 2Ø TIMES
                              PIC X.
     Ø5 FN-SUB                 PIC 99.
     Ø5 WORK-SUB               PIC 99.

PROCEDURE DIVISION.

NAME-ROUTINE.
     MOVE 1          TO FN.
     PERFORM MOVE-FIRST-NAME
                     VARYING WORK-SUB FROM 1 BY 1
                     UNTIL WORK-SUB > 2Ø
                     OR FIRSTN-CHAR (WORK-SUB) = SPACE.
     MOVE SPACE TO FULLNAME-CHAR (FN-SUB).
     ADD 1           TO FN-SUB.
     PERFORM MOVE-LAST-NAME
                     VARYING WORK-SUB FROM 1 BY 1
                     UNTIL WORK-SUB > 2Ø
                     OR LASTN-CHAR (WORK-SUB) = SPACE.
```

```
MOVE-FIRST-NAME.
    MOVE FIRSTN-CHAR (WORK-SUB) TO FULLNAME-CHAR (FN-SUB).
    ADD 1           TO FN-SUB.

MOVE-LAST-NAME.
    MOVE LASTN-CHAR (WORK-SUB) TO FULLNAME-CHAR (FN-SUB).
    ADD 1           TO FN-SUB.
```

In the COBOL II environment, the job is simpler:

```
DATA DIVISION.
    Ø5 FIRST-NAME             PIC X(2Ø).
    Ø5 LAST-NAME              PIC X(2Ø).
    Ø5 FULL-NAME              PIC X(3Ø).
PROCEDURE DIVISION.

NAME-ROUTINE.
            ** initialize the receiving field **
    MOVE SPACES      TO FULL-NAME.
            ** move the parts of the name into the receiving field **
    STRING FIRST-NAME DELIMITED BY SPACE
            SPACE
            LAST-NAME DELIMITED BY SIZE
                 INTO FULL-NAME
```

This will concatenate the first name, a space, and the last name into the full name. We could have delimited the last name by space also. For example:

```
if FIRST-NAME contains "ALLAN   "
and LAST-NAME contains "KALAR   "
then FULL-NAME will contain "ALLAN KALAR   "
```

Another example:

```
STRING  A  B  C  DELIMITED BY SIZE  INTO  D.
```

The general form of the command is:

```
STRING identifier-1 DELIMITED identifier-2
       literal-1    BY        literal-2
                              SIZE
                                  INTO identifier-3
                                  {{WITH} POINTER identifier-4}
                                  {{ON} OVERFLOW imperative-statement-1}
                                  {NOT {ON} OVERFLOW imperative-statement-2}
                                  {END-STRING}
```

Identifiers 1 to 3 must be USAGE DISPLAY. Numeric items must be integers and the PICTURE cannot contain a ‘P’.

The sending fields will be concatenated together INTO identifier-3. The scan will start from the left-most byte of the sending field (yes, you can use REFERENCE MODIFICATION; it will be evaluated once at the beginning of the cycle) and will stop when the DELIMITED phrase is satisfied. Then the next sending field will be laid down into identifier-3 behind the first, until all sending fields are moved in, or identifier-3 runs out of room. No space filling is provided, so remember to initialize identifier-3 first.

What if we wanted to take words out of a field instead of combining them? As you might expect, there is an UNSTRING command. Be aware that it can exhibit different behaviour when the CMPR2 compiler option is in effect.

Using our previous example, we can take a name apart and store its components in separate fields.

```
MOVE SPACES         TO FIRST-NAME LAST-NAME.
     UNSTRING FULL-NAME DELIMITED BY SPACE
                        INTO FIRST-NAME
                             LAST-NAME.
```

The delimiter (SPACE) will be put into the receiving field along with the text preceding it, room permitting. Using our example of "ALLAN KALAR", the resulting fields would look like this:

```
FIRST-NAME:  "ALLAN  "
LAST-NAME:   "KALAR  "
```

If we had specified a DELIMITER field, the space would have been put in a separate field instead.

First we need to define the fields:

```
DATA DIVISION.
    Ø5 FIRST-NAME            PIC X(2Ø).
    Ø5 LAST-NAME             PIC X(2Ø).
    Ø5 FULL-NAME             PIC X(3Ø).
    Ø5 FN-D                  PIC X.
    O5 LN-D                  PIC X.
```

Then our code would look like this:

```
     MOVE SPACES         TO FIRST-NAME LAST-NAME.
     UNSTRING FULL-NAME DELIMITED BY SPACE
                    INTO FIRST-NAME  DELIMITER FN-D
                         LAST-NAME   DELIMITER LN-D.
```

The space following each name would be placed in FN-D and LN-D, rather than in FIRST-NAME and LAST-NAME – not really a requirement of this exercise, but it might be useful in other situations where a delimiter is something other than a space, or you are extracting numbers out of an alphanumeric field.

The general format of the command is:

```
UNSTRING identifier-1
         {DELIMITED {BY} {ALL} — identifier-2
         literal-1
                    {OR  {ALL} — identifier-3 —  }
                                 literal-2
         }

         INTO identifier-4 {DELIMITER  identifier-5}

         {END-UNSTRING}
```

This is a simplified version of the format. The complete write-up is in the COBOL manual.

Allan Kalar
Systems Programmer (USA)

Cursor-sensitive ISPF (part 2)

This month we continue our look at the cursor-sensitive 'DS' command.

```
VGET (ZUSC) PROFILE                          /*   Get scroll amt.          */
                                             /*————————————————————————————*/
IF (&ZUSC = ' ') &ZUSC = 'CSR '              /* set default to 'CSR'       */
                                             /*————————————————————————————*/
VGET (DSLACT)                                /* get action from DS         */
IF (&DSLACT ¬= &Z)                           /* if action found            */
    IF (&DSLACT ¬= L)                        /* no action for list         */
        &DSSTART = TRUNC(&ZDATA,161)         /* data b4 1st d'set line     */
        &DUMMY  =  TRUNC(&ZDATA,162)         /*                            */
        &DSREST = .TRAIL                     /* rest of data               */
        &ZDATA  = '&DSSTART.&DSLACT.&DSREST' /* insert action              */
    &DSLACT = &Z                             /* reset to avoid repeat      */
    VPUT DSLACT                              /*                            */
ELSE                                         /* no option now              */
    IF (&DSSTART ¬= &Z)                      /* if option was inserted     */
        .RESP = END                          /* .. simulate END key        */
                                             /*————————————————————————————*/
```

```
)PROC
    ..
    ..
    ..
```

DS EXEC

```
/*============================>>  REXX  <<================================*/
/*                                                                       */
/*  This EXEC performs actions on datasets starting at the cursor        */
/*  position. The actions are listed below.                              */
/*                                                                       */
/*  THE FORMAT OF THE COMMAND:                                           */
/*     a) ===> DS action                                                 */
/*     - place the cursor anywhere on a dsname (or volser or msg-id)     */
/*  OR                                                                   */
/*     b) ===> DS action dsname (volume)                                 */
/*     - and it will use the supplied dsname (and ignore the cursor)     */
/*                                                                       */
/*  THE FOLLOWING PARAMETERS CAN BE SPECIFIED:                           */
/*                                                                       */
/* DEF       : display the default ACTION                                */
/* DEF action: define a default ACTION                                   */
/* NODEF     : remove default ACTION, then default is last-used action*/
/* HELP      : shows a tutorial-panel, that explains the function and */
/*                 parameters of the DS command.                         */
/* A         : dataset allocation information (invoking ISPF 3.2)        */
/* B         : BROWSE the dataset                                        */
/* BOOK      : search BookManager (using the bookshelf set by SHELF)     */
/* C         : CATALOG dataset                                           */
/* CMD       : browse CLIST or EXEC                                      */
/* CO        : COPY dataset or member(s)                                 */
/* D         : DELETE the dataset or member                              */
/* DD        : list DDNAMES (using ISRDDN)                               */
/* DI        : list dataset MEMBERS (invoking ISPF 3.1)                  */
/* E         : EDIT the dataset                                          */
/* H         : show HELP panel (same as action: HELP)                    */
/* I         : list dataset INFORMATION (using LISTDSI or %LVSAM)        */
/* L         : LIST datasets (invoking ISPF 3.4 via %DSLIST EXEC)        */
/* LC        : browse CATALOG information from TSO LISTCAT output        */
/* M         : list dataset MEMBERS (invoking ISPF 3.4 via %DSLIST)      */
/* MEM       : set DS member action to BROWSE, EDIT or VIEW              */
/* MO        : MOVE dataset member(s)                                    */
/* MSG       : browse ISPF message definition member                     */
/* PNL       : browse ISPF panel definition member                       */
/* Q         : show MVS enqueues (uses program DA$ENQS)                  */
/* R         : RENAME dataset                                            */
/* RS        : Reset Statistics of library member(s)                     */
/* S         : message showing SHORT dataset information (uses LISTD) */
/* SHELF     : set default bookshelf for BOOK action                     */
/* SKL       : browse ISPF skeleton                                      */
```

```
/* ST        : invoke StarTools                                          */
/* U         : UNCATALOG dataset                                         */
/* V         : VIEW the dataset                                          */
/* VOL       : list all datasets on VOLUME volser (invoking ISPF 3.4)    */
/* X (parm)  : EXECUTE (CLIST or EXEC), optionally passing a parm        */
/* Z         : COMPRESS library                                          */
/* ?         : show HELP panel (same as action: HELP)                    */
/*                                                                       */
/*  If the supplied (action) is not one of the above, it will try to     */
/*  invoke a TSO command: action 'dsname' .                              */
/*                                                                       */
/* EXECS     : CSRWORD / DSLIST / LVSAM / LCAT / BOOKSRCH / FINDMEM      */
/*                                                                       */
/* PANELS    : DSDEL / DSERR / DSHELP / LVSAMP / LVSAMH                  */
/*           : DSIEO / DSIES / DSILE / DSIP / DSIPO / DSIHELP            */
/*             ISRUDLP / ISRUDSLØ   (modified versions of IBM panels)    */
/*                                                                       */
/*=======================================================================*/
   Address ISPEXEC                          /* cmds default to ISPEXEC */
   "CONTROL ERRORS RETURN"                  /* handle ISPF error codes */
   "CONTROL DISPLAY SAVE"                   /* save display environment*/

   plib = 'prefix.DS.ISPPLIB'               /* panel library for DS    */
   Call LIBDEF_PLIB(plib)                   /* allocate via a LIBDEF   */

   Parse Upper ARG action DSN VOL . ,       /* get invocation parm(s)  */
                  1 parms                   /* save the whole string   */

   Call GET_DSVARS                          /* saved vars from table   */

   Call PROCESS_DEFAULTS                    /* process default actions */

   Call SAVE_LAST_ACT                       /* save last-used action   */

   If WordPos(action,'HELP H ? SHELF MEM DEF NODEF') = Ø Then Do
      Call GET_DSN                          /* get DSN & VOL values    */

      Call PROCESS_ACTION                   /* process the action      */
      End

   Call FINIS                               /* cleanup and EXIT        */
   Exit

/*=======================================================================*/
/*               Allocate DS panel library via LIBDEF                    */
/*-----------------------------------------------------------------------*/
LIBDEF_PLIB: Arg plib
   x = FIND_DSN(plib)                       /* check that it exists    */
   If x = Ø Then Do                         /* .. otherwise -> rc=2Ø   */
      "LIBDEF ISPPLIB DATASET ID('"plib"') STACK"
```

```
      libdef_rc = rc                        /* if rc=16 any underlying */
      End                                   /* .. LIBDEF will be lost  */
   Return

/*====================================================================*/
/*          Open table and get the stored variables for DS            */
/* - This is used instead of a normal PROFILE pool since DS runs in   */
/*   many different applids and inconsistent defaults could confuse.  */
/*--------------------------------------------------------------------*/
GET_DSVARS:
  "TBOPEN DSVARS LIBRARY(ISPPROF) WRITE SHARE"
  tbopen_rc = rc
  Select
    When tbopen_rc = Ø Then Do
      "TBTOP DSVARS"                        /* ensure cursor position */
      "TBSKIP DSVARS"                       /* get saved variables    */
      End
    When tbopen_rc = 8 Then
      Call NEW_DSVARS                       /* create new DSVARS table*/
    When tbopen_rc = 12 Then Do
       /*  table in use    */
       End
    Otherwise
       /*  severe error    */
    End
  If DSACTL= 'DSACTL' ! DSDFLT = 'DSDFLT',
   ! DSMEM = 'DSMEM' Then                  /* old version of table   */
      Call NEW_DSVARS                       /* create new DSVARS table*/
  Return tbopen_rc

/*====================================================================*/
/*        Create new table with (default) stored variables for DS     */
/*--------------------------------------------------------------------*/
NEW_DSVARS:
  "TBEND DSVARS"                            /* ensure no open table   */
  "TBCREATE DSVARS LIBRARY(ISPPROF) WRITE REPLACE SHARE",
     "NAMES(DSACTL DSDFLT BKSHELF DSMEM)"
  DSACTL  = 'I'                             /* display dataset info   */
  DSDFLT  = ''                              /* no permanent default   */
  BKSHELF = 'ALL'                           /* ALL shelves, no search */
  DSMEM   = 'BROWSE'                        /* BROWSE/EDIT members    */
  "TBADD DSVARS"
  "TBSAVE DSVARS LIBRARY(ISPPROF)"          /* save default values    */
  tbopen_rc = rc
  Return

/*====================================================================*/
/*               Process any DEFAULT actions (or HELP)                */
/*--------------------------------------------------------------------*/
PROCESS_DEFAULTS:        /* these are not saved as 'last-used action' */
   Select
     /*------*/
```

```
/* HELP */
/*------*/
When action = 'HELP' | action = 'H' | action = '?' Then Do
   Call RESIZE                          /* ensure no Pop-up window */
   "DISPLAY PANEL(DSHELP)"              /* HELP panel for DS       */
   End
/*-----------------------*/
/* BOOKMANAGER BOOKSHELF */
/*-----------------------*/
When action = 'SHELF' Then Do           /* SET or DISPLAY DEFAULT  */
   If DSN = '' Then Do                  /* no shelf parm supplied  */
      Call GET_SCREEN_TEXT              /* get ISPF screen buffer  */
      Call EXTRACT_DSNAME               /* get DSN (shelf name)    */
      End
   If DSN = '' Then
      msg = 'BookManager BOOK SHELF is:' BKSHELF
   Else Do
      BKSHELF = DSN
      "TBPUT DSVARS"                                /* update table */
      msg = 'BookManager BOOK SHELF set to:' BKSHELF
      End
   Call LONGMSG                         /* create ISPF message     */
   End

/*----------------*/
/* MEMBER DEFAULT */
/*----------------*/
When action = 'MEM' Then Do             /* SET MEMBER DEFAULT      */
   If DSN = '' Then
      msg = 'DS member displays via:' DSMEM
   Else Do
      Select
        When Left(DSN,1) = 'E' Then DSMEM = 'EDIT'
        When Left(DSN,1) = 'V' Then DSMEM = 'VIEW'
        Otherwise DSMEM = 'BROWSE'
        End
      msg = 'DS member displays set to:' DSMEM
      "TBPUT DSVARS"                                /* update table */
      End
   Call LONGMSG                         /* create ISPF message     */
   End
/*----------------*/
/* DEFAULT ACTION */
/*----------------*/
When action = 'DEF' Then Do             /* SET PERMANENT DEFAULT   */
   If DSN <> '' Then Do
      DSDFLT = DSN
      msg = 'Default DS action set to:' DSDFLT
      "TBPUT DSVARS"                                /* update table */
      End
   Else Do                              /* DISPLAY CURRENT DEFAULT */
      If DSDFLT = '' Then
```

```
            msg = 'Default DS action is:' DSACTL '(last-used action)'
         Else
            msg = 'Default DS action is:' DSDFLT
         End
      Call LONGMSG                        /* create ISPF message     */
      End
   When action = 'NODEF' Then Do          /* REMOVE DEFAULT ACTION   */
      DSDFLT = ''
      "TBPUT DSVARS"                                 /* update table */
      msg = 'Default DS action set to:' DSACTL '(last-used action)'
      Call LONGMSG                        /* create ISPF message     */
      End
   /*--------------------*/
   /* USE DEFAULT ACTION */
   /*--------------------*/
   When action = '' Then Do               /* SETUP DEFAULT ACTION    */
      If DSDFLT <> ''
         Then action = DSDFLT             /* permanent default action*/
         Else action = DSACTL             /* last-used action        */
      End
   Otherwise NOP
   End
Return

/*=====================================================================*/
/*               Save last-used action                                 */
/*---------------------------------------------------------------------*/
SAVE_LAST_ACT:
   e_list = 'HELP H ? SHELF MEM DEF NODEF X'   /* actions not saved   */
   If action <> '' & WordPos(action,e_list) = Ø Then Do
      DSACTL = action
      "TBPUT DSVARS"                           /* update table        */
      End
   "TBCLOSE DSVARS LIBRARY(ISPPROF)"           /* finished with table */
   Return

/*=====================================================================*/
/*               Get a value for DSN                                   */
/*---------------------------------------------------------------------*/
GET_DSN:
   /*----------------------------------*/
   /* GET DSNAME FROM COMMAND PARAMETER */
   /*----------------------------------*/
   dsn_quotes = Ø
   If DSN <> '' Then Do                    /* DSN parm supplied       */
      If Pos("'",dsn) > Ø Then Do
         dsn_quotes = 1
         dsn =STRIP(dsn,'B',"'")           /* remove quotes           */
         End
      End
   If action = 'BOOK' Then
      Parse Var parms . DSN    /* search string can be multiple words */
```

```
   If action = 'X' Then DSN = ''      /* to get dsname from csr pos'n */
   /*-----------------------------------------*/
   /* GET DSNAME FROM CURSOR POSITION ON SCREEN */
   /*-----------------------------------------*/
   If DSN = '' Then Do                       /* no DSN parm supplied    */
      Call GET_SCREEN_TEXT                   /* get ISPF screen buffer  */
      Call EXTRACT_DSNAME                    /* get DSN                 */
      End
   If action = 'X' Then Do
      If DSN = ''                            /* no DSN from csr pos'n   */
         Then Parse Upper Var parms . DSN VOL   /* use original parms */
         Else Parse Upper Var parms . VOL
      End
   /*-------------------------------------*/
   /* IF NO DSN VALUE - SHOW ERROR PANEL */
   /*-------------------------------------*/
   If DSN = '' & parms <> '',             /* still no dsname found   */
    & Wordpos(action,'DD BOOK') = Ø Then Do
      msgid = 'ISPP19Ø'                   /* "Enter required field"  */
      Call ERROR_PANEL(msgid)             /* display ERROR panel     */
      If result > Ø Then
         Call FINIS                       /* EXIT */
      End
   If DSN = '' & parms = '' Then Do       /* no dsname or action     */
      Call RESIZE
      "DISPLAY PANEL(DSHELP)"             /* display HELP info       */
      Call FINIS                          /* EXIT */
      End
   Return

/*=====================================================================*/
/*               Process the action                                    */
/*---------------------------------------------------------------------*/
PROCESS_ACTION:
 If Sysvar("SYSTSOE") < 2Ø5Ø       /* SMSINFO avail from TSO/E 2.5 on */
    Then smsinfo = ''              /* ... it is used in LISTDSI cmd   */

 exc_list = 'VOL L LC BOOK SHELF DD MSG PNL SKL CMD'
 Do num = 1 to 999                         /* try up to 999 times   */
    /*--------------------------------*/
    /* CHECK THAT THE DATASET EXISTS */
    /*--------------------------------*/
    If DSN <> '' & WordPos(action,exc_list) = Ø Then Do
       x = FIND_DSN(DSN)                /* look for DSN as specified */
       If x > 4 Then Do                 /* If not found .....        */
          If ¬DSN_quotes Then Do
             prefx = Sysvar(SYSPREF)
             If prefx = '' Then
                prefx = Userid()
             FULLDSN = prefx"."DSN           /* add prefix to DSN     */
             x = FIND_DSN(FULLDSN 'NOMSG')   /* look again for it     */
             End
```

```
      If x > 4 Then Do                  /* If still not found .. */
         Call ERROR_PANEL('LONGMSG')    /* display ERROR panel   */
         If result = Ø Then Iterate num          /* try again    */
                       Else Call FINIS           /* EXIT from DS */
         End
      Else DSN = FULLDSN             /* dataset found by LISTDSI */
      End
   End

/*-------------------------------*/
/* PROCESS ACTION ON THE DATASET */
/*-------------------------------*/
Select
  /*---------------------*/  /* The user sees dataset information */
  /* DATASET INFORMATION */  /* followed by ISRUDA2S panel which  */
  /*---------------------*/  /* can be used to allocate new d'set.*/
  When action = 'A' Then Do
     DSNAME = "'"DSN"'"
     VOLUME = VOL
     "VPUT (DSNAME VOLUME)"           /* pass args to pgm ISRUDA */
     "CONTROL NONDISPL ENTER"     /* simulate ENTER on next panel */
     "SELECT PGM(ISRUDA) PARM(ISRUDA2)",/* invoke normal ISPF 3.1 */
        "SUSPEND"                     /* ensure no Pop-up window */
     End

  /*-------------------------------*/
  /* DATASET BROWSE, EDIT or VIEW */
  /*-------------------------------*/
  When action = 'B' | action = 'E' | action = 'V' Then Do
     If Right(DSN,3) = '(*)' Then          /* '*' not supported   */
        DSN = Left(DSN,Length(DSN)-3)
     If Pos('(',DSN) > Ø Then Do           /* MEMBER specified    */
        ZERRMSG = ''              /* clear any previous error msg */
        If action = 'B' Then Do
           Call RESIZE                        /* no Pop-up window */
           act = 'BROWSE'
           End
        If action = 'E' Then act = 'EDIT'
        If action = 'V' Then act = 'VIEW'
        act "DATASET('"DSN"') VOLUME("VOL")"
        If ZERRMSG <> '' Then Do
           Call ERROR_PANEL('ZERRMSG')   /* display ERROR panel   */
           If result = Ø Then Iterate num         /* try again    */
           End
        End
     Else                              /* NO MEMBER specified     */
        Call DLIST                           /* invoke ISPF 3.4   */
     End

  /*-----------------------------*/
  /* MEMBER BROWSE, EDIT or VIEW */
  /*-----------------------------*/
```

```
When Wordpos(action,'CMD MSG PNL SKL') > Ø Then Do
   Call MEMBER_ACTION
   If result = Ø Then
      Iterate num            /* recursively invoke action B,E,V */
   End

/*--------------------*/
/* BOOKMANAGER SEARCH */
/*--------------------*/
When action = 'BOOK' Then Do
   s_name = Left(Word(dsn,1),8) /* up to 8 chars from srch strng */
   "SELECT PGM(ISPSTRT)",                        /* swap screen   */
      "PARM(CMD(%BOOKSRCH" bkshelf dsn")",       /* invoke search */
           "SCRNAME("s_name") )"                 /* screen name   */
   End

/*--------------*/
/* COPY or MOVE */
/*--------------*/
When action = 'CO' ! action = 'MO' Then Do
   If Pos('(',DSN) > Ø Then Do  /* member specified             */
      "CONTROL NONDISPL ENTER"  /* simulate ENTER on next panel */
      ZOPT = Left(action,1)     /* set option for panel ISRUMC1 */
      DSNAME = "'"DSN"'"
      VOLUME = VOL
      "VPUT (ZOPT DSNAME VOLUME)"    /* pass args to pgm ISRUMC */
      "SELECT PGM(ISRUMC) SCRNAME(MCOPY)", /* invoke normal 3.3 */
         "SUSPEND"                   /* ensure no Pop-up window */
      End
   Else                                  /* no member specified */
      Call DLIST                          /* invoke ISPF 3.4   */
   End

/*--------*/
/* DELETE */
/*--------*/
When action = 'D' Then Do
   If Pos('(',DSN) > Ø Then Do            /* member specified   */
      Call DSNINFO
      "ADDPOP"
      "DISPLAY PANEL(DSDEL)"
      DelRC = RC
      "REMPOP"
      If DelRC = Ø Then Do
         Address TSO "DELETE '"DSN"'"
         DelRC = RC
         Select
            When DelRC = Ø Then Do
               MSG = "'"DSN"' was Deleted"
               Call LONGMSG
```

```
               End
            Otherwise
               MSG = "Error on DELETE of '"DSN"'"
               Call LONGMSG
            End
         End
      Else Do
         MSG ="Delete of '"DSN"' was cancelled"
         Call LONGMSG
         End
      End
   Else                                  /* no member specified */
      Call DLIST                         /* invoke ISPF 3.4     */
   End

/*-------------*/
/* DDNAME LIST */          /* using Doug Nadel's program ISRDDN */
/*-------------*/
When action = 'DD' Then Do
   Call RESIZE                              /* no Pop-up window */
   DSN = Translate(DSN,'  ','*%')
   cmdstack = 'TSO ISRDDN'd'0' DSN          /* usually  d = ';' */
   If VOL <> '' Then
      cmdstack = cmdstack||d'MEMBER' VOL
   "DISPLAY PANEL(ISRıPRIM) COMMAND(cmdstack)"
   End

/*----------------------*/
/* DATASET INFORMATION */
/*----------------------*/
When action = 'DI' Then Do
   action = 'I'
   Call DLIST                             /* invoke ISPF 3.4     */
   End

/*----------------------*/
/* DATASET INFORMATION */
/*----------------------*/
When action = 'I' Then Do
   Call RESIZE                              /* no Pop-up window */
   Call DSNDSI DSN              /* using LISTDSI (& LVSAM?)     */
   If result = Ø Then Iterate num                      /* EXIT */
   End

/*---------------*/             /* This block makes the dataset */
/* LIST DATASETS */             /* list, then shows ISRUDLP pnl */
/*---------------*/             /* after leaving the list.      */
When action = 'L' Then
   Address TSO "%DSLIST" "'"DSN"'" VOL

/*----------------*/   /* This block uses a dataset list and    */
/* DSLIST ACTIONS */   /* enters the action without the user    */
```

```
    /*----------------*/   /* ever seeing panel ISRUDLP or ISRDULSØ.*/
    When action = 'C' ,                     /* CATALOG dataset     */
       ! action = 'F' ,                     /* FREE unused space   */
       ! action = 'L' ,    /* <-- not active  - LIST datasets       */
       ! action = 'M' ,                     /* MEMBER list         */
       ! action = 'R' ,                     /* RENAME dataset      */
       ! action = 'RS',                     /* Reset Statistics    */
       ! action = 'U' ,                     /* UNCATALOG dataset   */
       ! action = 'Z' ,                     /* COMPRESS dataset    */
    Then
       Call DLIST                           /* invoke ISPF 3.4     */

    /*--------------*/
    /* LIST CATALOG */
    /*--------------*/
    When action = 'LC' Then
       "SELECT CMD(%LCAT '"DSN"') SUSPEND"

    /*----------------------*/          /* This block is not active */
    /* LIST DATASET MEMBERS */          /* while action M is in the */
    /*----------------------*/          /* DSLIST ACTIONS block.    */
    When action = 'M' Then Do
       DSNAME = "'"DSN"'"
       VOLUME = VOL
       "VPUT (DSNAME VOLUME)"            /* pass args to pgm ISRUDA */
       "CONTROL NONDISPL ENTER"     /* simulate ENTER on next panel */
       "SELECT PGM(ISRUDA) PARM(ISRUDA1)",      /* invoke normal 3.1 */
          "SUSPEND"                      /* ensure no Pop-up window */
       End

    /*---------------------------*/
    /* SHORT DATASET INFORMATION */
    /*---------------------------*/
    When action = 'S' Then Do
       Call DSNINFO
       MSG = "RECFM:"Recfm "   Lrecl:"LRECL"    ",
                "Blksize:"BLKSIZE"   Dsorg:"DSORG"    Vol:"VOL
       Call LONGMSG
       End

/*  /*----------*/                                                  */
/*  /* ENQUEUES */            * using Dave Alcock's WHOHAS program */
/*  /*----------*/                                                  */
/*  When action = 'Q' Then Do   (showing 3 alternative invocations) */
/*  /* Queue "DA$ENQS '"DSN"'" */                                   */
/*  /* "SELECT PGM(DA$ENQS) PARM(''"DSN"'') MODE(FSCR)" */          */
/*     Address TSO "CALL 'load.library(DA$ENQS)' ''"DSN"''"       */
/*     End                                                         */
/*                                                                 */
/*  /*-----------*/                                               */
```

```
/*  /* STARTOOLS */                                    * from Serena */
/*  /*-----------*/                                                    */
/*  When action = 'ST' Then Do                                        */
/*     Select                                                         */
/*       When DSN = '' Then                                           */
/*          "SELECT PANEL(PDSiPRIM) SUSPEND"                          */
/*       When VOL = '' Then                                           */
/*          "SELECT CMD(PDS '"DSN"') SUSPEND"                         */
/*       Otherwise                                                    */
/*          "SELECT CMD(PDS '"DSN"' VOL("VOL") ) SUSPEND"             */
/*       End                                                          */
/*     End                                                            */

    /*---------------------*/
    /* VOLUME DATASET LIST */
    /*---------------------*/
    When action = 'VOL' Then Do
       If VOL = '' Then
          Address TSO "%DSLIST V" DSN
       Else
          Address TSO "%DSLIST V" VOL
       End

    /*---------*/   /* Note:  Address TSO "EXEC '"DSN"' '"parm"'"   */
    /* EXECUTE */   /*    would work OK with Rexx, but NOT with any */
    /*---------*/   /*    CLIST using GLOBAL symbolic parameters.   */
    When action = 'X' Then Do
       parm = VOL                        /* VOL holds any parameters */
       Queue "EXEC '"DSN"' '"parm"'"    /* <-- OK for Rexx or CLIST */
       End

    /*------------------*/
    /* COMPRESS DATASET */    /* <---- this is currently NOT ACTIVE */
    /*------------------*/
    When action = 'Z' Then
       Address TSO "PDSE '"DSN"' COMPRESS"

    /*----------------------*/
    /* MISCELLANEOUS ACTION */
    /*----------------------*/
    Otherwise
    /* Address TSO action "'"DSN"'" VOL */
       "SELECT CMD("action "'"DSN"'" VOL" ) SUSPEND MODE(FSCR)"
       If rc > Ø Then Do
          MSG =  'RC='rc 'from TSO command:' action "'"DSN"'" VOL
          Call ERROR_PANEL('LONGMSG')      /* display ERROR panel   */
          If result = Ø Then Iterate num            /* try again    */
          End
     End                                           /* End Select    */
  Return
  End                                              /* End num loop  */
```

```
 Return

/*====================================================================*/
/*               Set-up ISPF message                                  */
*/
/*--------------------------------------------------------------------*/
LONGMSG:
   ZERRHM   = 'DSHELP'
   ZERRSM   = ''
   ZERRALRM = 'YES'
   ZERRLM   = MSG
   ZERRTYPE = 'WARNING'
   "SETMSG MSG(ISRZØØ3)"              /* standard ISPF dummy message */
   Return

/*====================================================================*/
/*               Display DS Error panel                               */
/*--------------------------------------------------------------------*/
ERROR_PANEL:
   Arg message
   If message = 'LONGMSG' Then        /* use msg text from DS exec   */
      Call LONGMSG
   If message = 'ZERRMSG' Then        /* use standard ISPF error msg */
      "SETMSG MSG("ZERRMSG")"
   "ADDPOP"                           /* panel in Pop-Up Window      */
   "DISPLAY PANEL(DSERR)"             /* display DS ERROR panel      */
   dis_rc = rc                        /* rc=Ø <ENTER>, or rc=4 <END> */
   "REMPOP"
   Return dis_rc

/*====================================================================*/
/*               Remove any Pop-up window                             */
/*--------------------------------------------------------------------*/
RESIZE:
   "TBOPEN ISPSPROF SHARE"
   "TBTOP ISPSPROF"
   "TBSKIP ISPSPROF"                   /* get command delimiter ZDEL */
   "TBEND ISPSPROF"
   d = ZDEL
   "CONTROL NONDISPL END"
   cmdstack = 'RESIZE'd'END'           /* ISPF commands to invoke    */
   "DISPLAY PANEL(DSBLANK) COMMAND(cmdstack)"
   Return

/*====================================================================*/
/*               Find dataset                                         */
/*--------------------------------------------------------------------*/
FIND_DSN:
   Arg dsname setmsg
   Parse Var dsname dsname '(' .            /* remove any membername */

   If VOL <> '' & VOL <> 'VOL' & action <> 'X'
```

```
      Then loc ='VOLUME('VOL')'              /* volume information    */
      Else loc =''                           /* no volume specified   */

   If action = 'I' | action = '?' Then
      List_rc = LISTDSI("'"dsname"'" loc "RECALL DIRECTORY" smsinfo)
   Else
      List_rc = LISTDSI("'"dsname"'" loc "RECALL")

   If List_rc = 16 & setmsg <> 'NOMSG' Then Do
      Select
    When SYSREASON = Ø  Then msg = ''
    When SYSREASON = 1  Then msg = 'LISTDSI parsing error'
    When SYSREASON = 2  Then msg = 'Dynamic allocation error'
    When SYSREASON = 3  Then msg = 'Invalid dataset type'
    When SYSREASON = 4  Then msg = 'Error determining UNIT name'
    When SYSREASON = 5  Then msg = 'Dataset not catalogUed'
    When SYSREASON = 8  Then msg = 'Dataset not on disk'
    When SYSREASON = 11 Then msg = 'Not authorized to read directory'
    When SYSREASON = 12 Then msg = 'VSAM dataset'
    When SYSREASON = 13 Then msg = 'Dataset could not be opened'
    When SYSREASON = 18 Then msg = 'Partial data set information'
    When SYSREASON = 19 Then msg = 'Dataset on multiple volumes'
    When SYSREASON = 21 Then msg = 'Catalog error locating dataset'
    When SYSREASON = 22 Then msg = 'Volume not mounted'
    When SYSREASON = 23 Then msg = 'Permanent I/O error on volume'
    When SYSREASON = 24 Then msg = 'Dataset not found'
    When SYSREASON = 25 Then msg = 'Dataset migrated to non-DASD device'
    When SYSREASON = 3Ø Then msg = 'Dataset not SMS managed'
/*==========================>>  REXX  <<==============================*/
/*                                                                    */
/*  This EXEC performs actions on datasets starting at the cursor     */
/*  position. The actions are listed below.                           */
/*                                                                    */
/*  THE FORMAT OF THE COMMAND:                                        */
/*     a) ===> DS action                                              */
/*     - place the cursor anywhere on a dsname (or volser or msg-id) */
/*  OR                                                                */
/*     b) ===> DS action dsname (volume)                              */
/*     - and it will use the supplied dsname (and ignore the cursor)  */
/*                                                                    */
/*  THE FOLLOWING PARAMETERS CAN BE SPECIFIED:                        */
/*                                                                    */
/* DEF       : display the default ACTION                           */
/* DEF action: define a default ACTION                              */
/* NODEF     : remove default ACTION, then default is last-used action*/
/* HELP      : shows a tutorial-panel, that explains the function and */
/*                 parameters of the DS command.                      */
/* A         : dataset allocation information (invoking ISPF 3.2)   */
/* B         : BROWSE the dataset                                   */
/* BOOK      : search BookManager (using the bookshelf set by SHELF)  */
```

```
/* C         : CATALOG dataset                                          */
/* CMD       : browse CLIST or EXEC                                     */
/* CO        : COPY dataset or member(s)                                */
/* D         : DELETE the dataset or member                             */
/* DD        : list DDNAMES (using ISRDDN)                              */
/* DI        : list dataset MEMBERS (invoking ISPF 3.1)                 */
/* E         : EDIT the dataset                                         */
/* H         : show HELP panel (same as action: HELP)                   */
/* I         : list dataset INFORMATION (using LISTDSI or %LVSAM)       */
/* L         : LIST datasets (invoking ISPF 3.4 via %DSLIST exec)       */
/* LC        : browse CATALOG information from TSO LISTCAT output       */
/* M         : list dataset MEMBERS (invoking ISPF 3.4 via %DSLIST)     */
/* MEM       : set DS member action to BROWSE, EDIT or VIEW             */
/* MO        : MOVE dataset member(s)                                   */
/* MSG       : browse ISPF message definition member                    */
/* PNL       : browse ISPF panel definition member                      */
/* Q         : show MVS enqueues (uses program DA$ENQS)                 */
/* R         : RENAME dataset                                           */
/* RS        : Reset Statistics of library member(s)                    */
/* S         : message showing SHORT dataset information (uses LISTD)   */
/* SHELF     : set default bookshelf for BOOK action                    */
/* SKL       : browse ISPF skeleton                                     */
/* ST        : invoke StarTools                                         */
/* U         : UNCATALOG dataset                                        */
/* V         : VIEW the dataset                                         */
/* VOL       : list all datasets on VOLUME volser (invoking ISPF 3.4)   */
/* X (parm)  : EXECUTE (clist or exec), optionally passing a parm       */
/* Z         : COMPRESS library                                         */
/* ?         : show HELP panel (same as action: HELP)                   */
/*                                                                      */
/*  If the supplied (action) is not one of the above, it will try to    */
/*  invoke a TSO command: action 'dsname' .                             */
/*                                                                      */
/* EXECS     : CSRWORD / DSLIST / LVSAM / LCAT / BOOKSRCH / FINDMEM     */
/*                                                                      */
/* PANELS    : DSDEL / DSERR / DSHELP / LVSAMP / LVSAMH                 */
/*           : DSIEO / DSIES / DSILE / DSIP / DSIPO / DSIHELP           */
/*             ISRUDLP / ISRUDSLØ   (modified versions of IBM panels)   */
/*                                                                      */
/*======================================================================*/
   Address ISPEXEC                          /* cmds default to ISPEXEC */
   "CONTROL ERRORS RETURN"                  /* handle ISPF error codes */
   "CONTROL DISPLAY SAVE"                   /* save display environment*/

   plib = 'prefix.DS.ISPPLIB'               /* panel library for DS    */
   Call LIBDEF_PLIB(plib)                   /* allocate via a LIBDEF   */

   Parse Upper ARG action DSN VOL . ,       /* get invocation parm(s)  */
                  1 parms                   /* save the whole string   */

   Call GET_DSVARS                          /* saved vars from table   */
```

```
   Call PROCESS_DEFAULTS                  /* process default actions */

   Call SAVE_LAST_ACT                     /* save last-used action   */

   If WordPos(action,'HELP H ? SHELF MEM DEF NODEF') = Ø Then Do
      Call GET_DSN                        /* get DSN and VOL values  */

      Call PROCESS_ACTION                 /* process the action      */
      End

   Call FINIS                             /* clean-up and EXIT       */
   Exit

/*====================================================================*/
/*               Allocate DS panel library via LIBDEF                 */
/*--------------------------------------------------------------------*/
LIBDEF_PLIB: Arg plib
   x = FIND_DSN(plib)                     /* check that it exists    */
   If x = Ø Then Do                       /* .. otherwise -> rc=2Ø   */
      "LIBDEF ISPPLIB DATASET ID('"plib"') STACK"
      libdef_rc = rc                      /* if rc=16 any underlying */
      End                                 /* .. LIBDEF will be lost  */
   Return

/*====================================================================*/
/*          Open table and get the stored variables for DS            */
/* - This is used instead of a normal PROFILE pool since DS runs in   */
/*   many different applids and inconsistent defaults could confuse.  */
/*--------------------------------------------------------------------*/
GET_DSVARS:
  "TBOPEN DSVARS LIBRARY(ISPPROF) WRITE SHARE"
  tbopen_rc = rc
  Select
    When tbopen_rc = Ø Then Do
      "TBTOP DSVARS"                      /* ensure cursor position */
      "TBSKIP DSVARS"                     /* get saved variables    */
      End
    When tbopen_rc = 8 Then
      Call NEW_DSVARS                     /* create new DSVARS table*/
    When tbopen_rc = 12 Then Do
       /*  table in use    */
       End
    Otherwise
       /*  severe error    */
    End
  If DSACTL= 'DSACTL' ! DSDFLT = 'DSDFLT',
   ! DSMEM = 'DSMEM' Then                 /* old version of table   */
      Call NEW_DSVARS                     /* create new DSVARS table*/
  Return tbopen_rc
```

```
/*====================================================================*/
/*        Create new table with (default) stored variables for DS     */
/*--------------------------------------------------------------------*/
NEW_DSVARS:
  "TBEND DSVARS"                              /* ensure no open table   */
  "TBCREATE DSVARS LIBRARY(ISPPROF) WRITE REPLACE SHARE",
     "NAMES(DSACTL DSDFLT BKSHELF DSMEM)"
  DSACTL  = 'I'                               /* display dataset info   */
  DSDFLT  = ''                                /* no permanent default   */
  BKSHELF = 'ALL'                             /* ALL shelves, no search */
  DSMEM   = 'BROWSE'                          /* BROWSE/EDIT members    */
  "TBADD DSVARS"
  "TBSAVE DSVARS LIBRARY(ISPPROF)"            /* save default values    */
  tbopen_rc = rc
  Return

/*====================================================================*/
/*               Process any DEFAULT actions (or HELP)                */
/*--------------------------------------------------------------------*/
PROCESS_DEFAULTS:        /* these are not saved as 'last-used action' */
   Select
     /*------*/
     /* HELP */
     /*------*/
     When action = 'HELP' ! action = 'H' ! action = '?' Then Do
        Call RESIZE                           /* ensure no Pop-up window */
        "DISPLAY PANEL(DSHELP)"               /* HELP panel for DS       */
        End
     /*-----------------------*/
     /* BOOKMANAGER BOOKSHELF */
     /*-----------------------*/
     When action = 'SHELF' Then Do            /* SET or DISPLAY DEFAULT  */
        If DSN = '' Then Do                   /* no shelf parm supplied  */
           Call GET_SCREEN_TEXT               /* get ISPF screen buffer  */
           Call EXTRACT_DSNAME                /* get DSN (shelf name)    */
           End
        If DSN = '' Then
           msg = 'BookManager BOOK SHELF is:' BKSHELF
        Else Do
           BKSHELF = DSN
           "TBPUT DSVARS"                                 /* update table */
           msg = 'BookManager BOOK SHELF set to:' BKSHELF
           End
        Call LONGMSG                          /* create ISPF message     */
        End

     /*----------------*/
     /* MEMBER DEFAULT */
     /*----------------*/
     When action = 'MEM' Then Do              /* SET MEMBER DEFAULT      */
        If DSN = '' Then
```

```
            msg = 'DS member displays via:' DSMEM
         Else Do
            Select
              When Left(DSN,1) = 'E' Then DSMEM = 'EDIT'
              When Left(DSN,1) = 'V' Then DSMEM = 'VIEW'
              Otherwise DSMEM = 'BROWSE'
              End
            msg = 'DS member displays set to:' DSMEM
            "TBPUT DSVARS"                              /* update table */
            End
         Call LONGMSG                        /* create ISPF message     */
         End
      /*-----------------*/
      /* DEFAULT ACTION */
      /*-----------------*/
      When action = 'DEF' Then Do            /* SET PERMANENT DEFAULT   */
         If DSN <> '' Then Do
            DSDFLT = DSN
            msg = 'Default DS action set to:' DSDFLT
            "TBPUT DSVARS"                              /* update table */
            End
         Else Do                            /* DISPLAY CURRENT DEFAULT */
            If DSDFLT = '' Then
               msg = 'Default DS action is:' DSACTL '(last-used action)'
            Else
               msg = 'Default DS action is:' DSDFLT
            End
         Call LONGMSG                        /* create ISPF message     */
         End
      When action = 'NODEF' Then Do          /* REMOVE DEFAULT ACTION   */
         DSDFLT = ''
         "TBPUT DSVARS"                                 /* update table */
         msg = 'Default DS action set to:' DSACTL '(last-used action)'
         Call LONGMSG                        /* create ISPF message     */
         End
      /*--------------------*/
      /* USE DEFAULT ACTION */
      /*--------------------*/
      When action = '' Then Do               /* SETUP DEFAULT ACTION    */
         If DSDFLT <> ''
            Then action = DSDFLT             /* permanent default action*/
            Else action = DSACTL             /* last-used action        */
         End
      Otherwise NOP
      End
   Return

/*====================================================================*/
/*               Save last-used action                                */
/*--------------------------------------------------------------------*/
SAVE_LAST_ACT:
   e_list = 'HELP H ? SHELF MEM DEF NODEF X'   /* actions not saved   */
```

```
   If action <> '' & WordPos(action,e_list) = Ø Then Do
      DSACTL = action
      "TBPUT DSVARS"                            /* update table        */
      End
   "TBCLOSE DSVARS LIBRARY(ISPPROF)"            /* finished with table */
   Return

/*====================================================================*/
/*               Get a value for DSN                                  */
/*--------------------------------------------------------------------*/
GET_DSN:
   /*-----------------------------------*/
   /* GET DSNAME FROM COMMAND PARAMETER */
   /*-----------------------------------*/
   dsn_quotes = Ø
   If DSN <> '' Then Do                     /* DSN parm supplied       */
      If Pos("'",dsn) > Ø Then Do
         dsn_quotes = 1
         dsn =STRIP(dsn,'B',"'")            /* remove quotes           */
         End
      End
   If action = 'BOOK' Then
      Parse Var parms . DSN    /* search string can be multiple words */
   If action = 'X' Then DSN = ''      /* to get dsname from csr pos'n */
   /*-----------------------------------------*/
   /* GET DSNAME FROM CURSOR POSITION ON SCREEN */
   /*-----------------------------------------*/
   If DSN = '' Then Do                      /* no DSN parm supplied    */
      Call GET_SCREEN_TEXT                  /* get ISPF screen buffer  */
      Call EXTRACT_DSNAME                   /* get DSN                 */
      End
   If action = 'X' Then Do
      If DSN = ''                           /* no DSN from csr pos'n   */
         Then Parse Upper Var parms . DSN VOL   /* use original parms */
         Else Parse Upper Var parms . VOL
      End
   /*-----------------------------------*/
   /* IF NO DSN VALUE - SHOW ERROR PANEL */
   /*-----------------------------------*/
   If DSN = '' & parms <> '',               /* still no dsname found   */
    & Wordpos(action,'DD BOOK') = Ø Then Do
      msgid = 'ISPP19Ø'                     /* "Enter required field"  */
      Call ERROR_PANEL(msgid)               /* display ERROR panel     */
      If result > Ø Then
         Call FINIS                         /* EXIT */
      End
   If DSN = '' & parms = '' Then Do         /* no dsname or action     */
      Call RESIZE
      "DISPLAY PANEL(DSHELP)"               /* display HELP info       */
      Call FINIS                            /* EXIT */
      End
   Return
```

```
/*====================================================================*/
/*               Process the action                                   */
/*--------------------------------------------------------------------*/
PROCESS_ACTION:
 If Sysvar("SYSTSOE") < 2050       /* SMSINFO avail from TSO/E 2.5 on */
    Then smsinfo = ''              /* ... it is used in LISTDSI cmd   */

 exc_list = 'VOL L LC BOOK SHELF DD MSG PNL SKL CMD'
 Do num = 1 to 999                          /* try up to 999 times   */
    /*--------------------------------*/
    /* CHECK THAT THE DATASET EXISTS */
    /*--------------------------------*/
    If DSN <> '' & WordPos(action,exc_list) = 0 Then Do
       x = FIND_DSN(DSN)                /* look for DSN as specified */
       If x >
```

DSLIST EXEC

```
/*============================== REXX ================================*/
/*  DSLIST - DATASET LIST (same as ISPF option 3.4)                   */
/*                                                                    */
/*     a) This is invoked directly by the DS EXEC.                    */
/*                                                                    */
/*     b) This is usually defined in an ISPF command table            */
/*          DL : 'SELECT CMD(%DSLIST &ZPARM) NEWAPPL'                 */
/*                                                                    */
/*        The user enters one of the following :                      */
/*          a) DL             - basic 3.4 panel                       */
/*          b) DL dsmask      - list of "dsmask" datasets             */
/*          c) DL V vol       - all datasets on "vol"                 */
/*          d) DL dsmask vol  - "dsmask" datasets on "vol"            */
/*                                                                    */
/*====================================================================*/
 Address ISPEXEC
 "CONTROL ERRORS RETURN"
 Parse Upper Arg dsn vol .             /* get invocation argument(s) */

 If dsn <> '' Then Do
    "CONTROL NONDISPL ENTER"         /* simulate ENTER on next panel */

    If vol = '' Then ZDLPVL = ''
    Else Do
       ZDLPVL = Left(vol,6)       /* ensure volser only 6 bytes long */
       If dsn = 'V' Then          /* show all datasets on the volume */
          ZDLDSNLV = ''           /* ....hence dsname mask is blank  */
       End
    "VPUT ZDLPVL"                          /* VPUT the volume serial */

    If dsn <> 'V' Then Do
       If Left(dsn,1) = "'" Then
```

```
         dsn = Strip(dsn,,"'")             /* strip quotes from dsn */
      Else If Sysvar(SYSPREF) <> '' Then
         dsn = Sysvar(SYSPREF)!!'.'!!dsn   /* add TSO prefix to dsn */
      mempos = Pos('(',dsn)
      If mempos > Ø Then    /* remove member or relative GDG number */
         dsn = Left(dsn,mempos-1)
      ZDLDSNLV = dsn
      End

   "VPUT ZDLDSNLV"                          /* VPUT the dsname mask */
   End

"SELECT PGM(ISRUDL) PARM(ISRUDLP)"            /* invoke normal 3.4 */
Return
```

FINDMEM EXEC

```
 /*======================>> REXX FUNCTION <<============================*/
/*                                                                      */
/*  FINDMEM    : Find a member in a specified ddname, as ISPF would,    */
/*               (ie check ALTLIB and LIBDEF where appropriate).        */
/*                                                                      */
/*               FINDMEM is invoked by the DS exec.                     */
/*                                                                      */
/*  Function   : It returns the DSNAME of the first library which       */
/*               contains that member.  It also returns the VOLUME      */
/*               if it is the first library of the concatenation.       */
/*                                                                      */
/*               Note that the member name can be exact or a mask       */
/*               using the % or * characters.                           */
/*                                                                      */
/*               This works correctly if datasets are allocated via     */
/*               the TSO ALLOCATE command.  If a DDname is allocated    */
/*               via JCL in the TSO procedure there can be an error,    */
/*               when an uncatalogUed dataset is not the first in the
*/
/*               concatenation.                                         */
/*                                                                      */
/*  Invocation : This should be called as a REXX external function      */
/*               eg    lib = FINDMEM(member ddname)                     */
/*                     If lib = member 'not found in' DDname            */
/*                        Then .......                                  */
/*                     If lib = 'DDname:' ddname 'not found'            */
/*                        Then .......                                  */
/*                     vol = Word(lib,2)    /* volser or blank */       */
/*                     lib = Word(lib,1)    /* library name */          */
/*                                                                      */
/*  Notes      : The first library is checked by attempting to BROWSE   */
/*               the member with a VOLUME parameter.  That is done in   */
/*               case it is not catalogued on that volume (or at all).*/
```

```
/*               All following libraries are also checked via BROWSE, */
/*               but with no VOLUME parameter.                         */
/*               ALLOCATE command can allocate only one uncatalogued  */
/*               library, or a concatenation of cataloged libraries.  */
/*                                                                     */
/*               SYSDSN is not used since it can't check uncatalogued */
/*               datasets, and it will not accept a member name mask. */
/*                                                                     */
/*               This code has many exit points: it exits as soon as  */
/*               it finds the member in a library.                     */
/*                                                                     */
/*=====================================================================*/
  Address ISPEXEC                          /* commands -> ISPEXEC */
  "CONTROL ERRORS RETURN"                  /* handle return codes */
  Arg member file

/*--------------------------------*/
/* Look for REXX or CLIST                                 */
/*--------------------------------*/
  If file = 'SYSEXEC' ! file = 'SYSPROC' Then Do
     Address TSO "ALTLIB DISPLAY QUIET"
     "VGET IKJADM SHARED"                  /* no of message lines */
     Do n = 3 to IKJADM
        "VGET IKJADM"n "SHARED"            /* get a message line  */
        Interpret 'msgline = IKJADM'n
        Parse Var msgline . '=' ddfind     /* extract the ddname  */
        If cmdfile = 'CMDFILE'
           Then cmdfile = ddfind
           Else cmdfile = cmdfile 'or' ddfind
        Call SEARCH_FILE(ddfind)           /* search libraries    */
        End
     EXIT member 'not found in' cmdfile    /* RETURN: ERROR MSG   */
     End

/*----------------------------*/
/* Not REXX or CLIST, so check for an ISPF LIBDEF */
/*----------------------------*/
  "QLIBDEF" file "TYPE(libtype) ID(libid)"
  If rc = 0 Then Do                        /* a LIBDEF is active  */
```

Editor's note: this article will be continued in the next edition.

Ron Brown
Systems Progammer (Germany)

MVS news

EMC has announced support for IBM's Geographically Dispersed Parallel Sysplex (GDPS) clustering with its Symmetrix Remote Data Facility (SRDF) business continuity software.

The enhanced version of SRDF targets very large System/390 environments in which GDPS is deployed and data runs across multiple EMC Symmetrix systems. SRDF already provides business continuity facilities for applications on smaller databases running on individual Symmetrix boxes.

SRDF automatically duplicates production data from a primary site to a secondary site. If the primary site becomes inoperable, SRDF enables rapid failover to the secondary site. Support for GDPS also enables use of the EMC TimeFinder point-in-time copy software at the remote GDPS site. In this configuration, users can create TimeFinder Business Continuity Volumes at the secondary site and use those volumes for software development, batch processing and remote restart protection.

For further information contact:

EMC Corp, 35 Parkwood Drive,
Hopkinton, MA, 01748-9103, USA.
Tel: (508) 435 1000
Fax: (508) 497 6915

EMC Computer Systems (UK), Windmill Court, Millfield Lane, Lower Kingswood, Tadworth, Surrey, KT20 6TS, UK.
Tel: (0870) 608 7777
Fax: (0870) 608 7788

http://www.emc.com

* * *

IBM has launched Version 2.1.0 of its Application Testing Collection for MVS/ESA and OS/390 (ATC), which is a set of productivity tools aimed at application development and maintenance testing in a variety of MVS environments, such as batch, CICS, IMS, DB2, and TSO/ISPF.

ATC comprises five tools:

The Coverage Assistant supports COBOL, PL/I, or Assembler programs and measures coverage while the programs are executing and generates reports of the source statements and conditions executed.

Distillation Assistant is used to create distilled testbeds, reducing the test case input files for COBOL and PL/I programs, potentially by 10 to 100, to a minimum number of records that will cause equivalent test case coverage.

Source Audit Assistant processes an IBM SuperC code comparison (for example pre and post changes) and provides reports on unexpected differences.

Unit Test Assistant allows the user to warp data (increment, decrement, or set data values) dynamically at defined points in the program logic.

Automated Regression Testing Tool helps automate the creation of batch and on-line test data and reports changes in program behaviour between a captured baseline and post-baseline runs of that program.

Contact your local IBM representative for further information.

* * *