136

MVS update

January 1998

In this issue

MVS Update

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

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

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

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
Dr Jaime Kaminski

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

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

PDS global edit using SELCOPY

The need to globally edit a string across all the members of a PDS is a fairly common one. For example, it may be necessary to change all the occurrences of a dataset name within the jobs that reference that dataset. Many sites will have written their own routine for doing this (possibly using library management facilities) or may even have a package to do it. Whether or not you have such a facility, you may still be interested in the following SELCOPY global update dialog. While preparing to carry out an in-house SELCOPY course, and as a result of going through the manual to make sure I was fully up-to-date with the SELCOPY language, I came across an interesting feature that I'd not noticed before. SELCOPY has an 'update in place' feature for PDSs, thereby making it possible to create an update system that would not suffer from potential space problems. Also, with its relatively recent REXX interface, it meant it would be straight-forward to create an ISPF-based dialog to exploit the update in place facility. As a result, I created the following code, which permits a string for string replacement, and includes a global find feature.

To initiate the dialog, simply specify TSO SELCHAN1 on your command line while in ISPF and you will be presented with the first panel. This will allow the specification of the PDS, where the change is to take place, and will also allow the specification of the change/find strings. If a change is required, there is a further option to specify whether the change is to take place for all occurrences of the string within a member, or just for the first. Specification of the string should ideally be in quotes to ensure an exact match (no quotes will cause SELCOPY to ignore whether the data is upper or lower case). There is also one extra entry on the panel to allow the user to specify whether they want the SELCOPY processing report to be printed. This may be useful to analyse the number of changes which took place. Note that, should an error be detected by SELCOPY, this report will be automatically produced.

Once the change or find has taken place, a table will be displayed showing all the members that have been modified or found. Placing a character at the side of a member name will allow the user direct access to view the member and see what happened. For any further information regarding the use of this dialog, please refer to the help panels SELCHAH1 and SELCHAH2 below.

Firstly the panels:

SELCHAP1 – the invocation panel.

```
)ATTR DEFAULT(%+_)
                %   TYPE(TEXT) INTENS(HIGH)
                +   TYPE(TEXT) INTENS(LOW)
                TYPE(INPUT) INTENS(HIGH) CAPS(OFF) JUST(LEFT)
                £   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY WINDOW(70,10)
+Specify PDS ===!dsname                                     +
+
+
+Find    ===>_change                                          +
+Replace ===>_to                                              +
+
+First or All ===£type +
+List  option ===£deb+ Set this to YES if selcopy output required+
+Press%ENTER+to action the change         +
+otherwise press%END+to abort the change     +
)INIT
.HELP = selchah1                /* INSERT NAME OF TUTORIAL PANEL  */
)PROC
VER (&DSNAME,NB)
VER (&DSNAME,DSNAME)
VER (&CHANGE,NB)
VER (&TYPE,LIST,FIRST,ALL)
VER (&DEB,LIST,YES)
&REPLY=.RESP
)END
```

SELCHAH1 – the help panel associated with SELCHAP1.

```
)BODY
`-------------- HELP PANEL FOR FIND/CHANGE UTILITY ---------------------
+
+This dialog front ends SELCOPY to carry out either a search or replace
+for the strings specified in the CHANGE and TO labels. If the TO option
+is left blank, then only a search takes place.
+Upon completion of the operation a table of which members were modified
+or which contain the string is shown. By specifying any character at
+the side of the member name it will be presented for viewing so that
+the results of the change can be seen, or the member can be
+individually scanned using ISPF.
+
+Note that the method of entering variable for change is SELCOPY
+standard. If hex is required use X'??' format, and if character is
```

```
+required, enter the data in quotes for an exact match/replacement.
+If quotes are not used then the match will take place irrespective of
+case and the replacement will be in upper case. Not using quotes can
+also cause some difficulties for SELCOPY, and is therefore not
+recommended. In the event of a SELCOPY error, the dialog should display
+the SELCOPY listing to permit diagnosis of the problem. However it is
+also possible to retrieve the listing to get information by specifying
+list YES. You may want to do this if you want an idea of the number of
+changes made by the dialog (for example).
)PROC
.help=isp00004
)END
```

SELCHAP2 – The member table display panel.

```
)Attr Default(%+_)
! type(output) intens(high) caps(on ) just(left )
£ type(input) intens(low) caps(on ) just(asis )
)Body Window(70,15)
%Command ===>_zcmd                %Scroll ===>_amt +
+
Member
)Model
£t!z       +
)Init
.HELP = selchah2
.ZVARS = '(member)'
&amt = PAGE
)reinit
&t=''
)PROC
&REPLY= .RESP
)End
```

SELCHAH2 – the help panel associated with SELCHAP2.

```
)BODY
`-------------- HELP PANEL FOR FIND/CHANGE UTILITY ---------------------
+
+This panel provides a member list of all the members that contain the
+change string. If a TO string was also specified then, these members
+have also been altered accordingly.
+If you wish to view any or all of these members, simply place a
+character alongside the relevant table entry.
)PROC
.help=isp00004
)END
```

SELCHAN1 – the controlling code.

```
/* REXX */
/* This dialog is designed to provide a global update feature     */
/* for PDS libraries based around SELCOPY                         */
/* */
start_point:
MEMBER.=''       /* re-initialise the member array */
/* */
/* display the initial panel */
/* */
zwinttl='PDS GLOBAL FIND/REPLACE'
ADDRESS ISPEXEC
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(SELCHAP1)'
'REMPOP'
/* */
/* If the user pressed PF3 end the dialog                          */
/* If they specified a replace field but didn't say what type then */
/*    reply with a suitable error message.                         */
/* If they didn't specify replace, but did specify a type then also*/
/*    reply with a suitable error message                          */
/* */
IF reply='END' THEN EXIT
IF SYSDSN(dsname)¬'OK' THEN DO
ZEDSMSG='Dataset not found'
ZEDLMSG=dsname 'not in catalog'
'SETMSG MSG(ISRZØØ1)'
SIGNAL start_point
  END
IF to¬'' & type='' THEN DO
               ZEDSMSG='What type of replace?'
               ZEDLMSG='is required. First or All'
               'SETMSG MSG(ISRZØØ1)'
               SIGNAL start_point
               END
            IF to='' & type¬'' THEN DO
               ZEDSMSG='You have specified' type
               ZEDLMSG='Did you mean to do a change?'
               'SETMSG MSG(ISRZØØ1)'
               SIGNAL start_point
               END
            /* */
            /* End of initial error messages. Now prepare the selcopy */
            /* and ensure all necessary files available              */
            /* */
            ADDRESS TSO
            call msg(off)
            'FREE FI(SYSPRINT,SYSIN,SPLODGE)'
"ALLOC FI(SYSPRINT) DELETE",
       "DSORG(PS) REC(F B) LR(133) BLK(133ØØ)",
       "SPACE(2,2) TRACKS"
X=LISTDSI('SYSPRINT' 'FILE')
PRINT_DSN=SYSDSNAME
PRINT_VOLUME=SYSVOLUME
"ALLOC FI(SYSIN) DELETE ",
       "DSORG(PS) REC(F B) LR(8Ø) BLK(312Ø)",
       "SPACE(1,2) TRACKS"
/* */
/* If to isn't specified then its only a find so need to avoid */
/* specifying update on the read                               */
/*/
IF to¬'' THEN,
     QUEUE " READ F=SPLODGE DSN='"DSNAME"' DIRDATA UPD W 5ØØØØ"
ELSE QUEUE " READ F=SPLODGE DSN='"DSNAME"' DIRDATA W 5ØØØØ"
QUEUE " IF EOF"
QUEUE " OR DIR"
/* */
/* the following bit of selcopy code is designed to cope with */
/* variable length numbers for the creation of rexx array     */
/* variables.                                                 */
/* */
QUEUE " THENIF POS 46ØØØ EXACT 'Y'"
QUEUE "    THEN ADD 1 TO 3 AT 49ØØØ"
QUEUE "    THEN CVPC 3 AT 49ØØØ TO 5 AT 491ØØ FORMAT ZZZZ9"
QUEUE "    THENIF POS 49Ø99,491Ø4 NE ' ' PTR @ARAY"
QUEUE "        THEN MOVE @ARAY,491Ø4 TO 42ØØ7"
QUEUE "        THENIF POS 42ØØØ,42Ø13 EXACT=' ' PTR @LEN"
QUEUE "        THEN XV SET @LEN-42ØØØ AT 42ØØØ FROM 8 AT 45ØØØ"
QUEUE " IF DIR"
QUEUE " THEN MOVE 8 FROM 1 TO 45ØØØ * PRESERVE MEMBER NAME"
QUEUE " THEN POS 42ØØØ MOD 'MEMBER.'"
QUEUE " THEN POS 46ØØØ MOD 'N'      * SET CHANGE FLAG"
QUEUE " THEN GG"
/* */
/* incount works by member so for a file start need to use */
/* stopaft to ensure the REXX variable counter is set      */
/* only once.                                              */
QUEUE " IF INCOUNT EQ 1 | THEN POS 49ØØØ MOD X'ØØØØØC' S 1"
/* */
/* will need to set a pointer per record to minimize searching */
/* */
QUEUE " @MINE=1"
/* */
/* If a full replace required then we need to loop */
/* */
IF type='ALL' THEN QUEUE "LOOPER"
QUEUE " IF @MINE GT L | THEN GG"
QUEUE " IF POS @MINE,L EQ" change "PTR @MINE"
/* */
/* Only need the following code if a change is to take place */
/* */
```

```
IF to¬'' THEN DO
     QUEUE "    THEN POS @MINE MOD "to
     QUEUE "    THEN UPDATE SPLODGE" /* no update required?*/
     END
/* */
QUEUE "    THEN @MINE=@MINE+1"
QUEUE "    THEN POS 46000 MOD 'Y'"
/* */
/* If a full replace required ensure the loop */
/* */
IF type='ALL' THEN QUEUE "THEN GOTO LOOPER"
ELSE QUEUE "THEN FLAG EOMEMB" /* Speed up first processing */
/* */
QUEUE ''
'EXECIO * DISKW SYSIN (FINIS'
ADDRESS TSO
/* */
/* err is used to keep the selcopy rc if necessary */
/* */
err=''
/* */
'SELCOPY'
/* */
/* RC up to 16 is ok otherwise error */
/* */
if rc>16 then err='YES'
ADDRESS ISPEXEC
/* */
/* If an error or a listing requested */
/* */
IF deb ='YES' | err='YES' THEN DO
    'VIEW DATASET('PRINT_DSN') VOLUME('PRINT_VOLUME')'
     END
/* */
/* If there has been a selcopy error send back a message */
/* */
IF err='YES' THEN DO
     ZEDSMSG='Selcopy code error'
     ZEDLMSG='Probably unsuitable find/replace data'
     'SETMSG MSG(ISRZ001)'
     SIGNAL start_point
     END
/* */
/* If no array members then nothing found or changed */
/* */
IF MEMBER.1='' THEN DO /* no members altered? */
     ZEDSMSG='No members altered'
     ZEDLMSG=change 'not found in any member'
     'SETMSG MSG(ISRZ001)'
     SIGNAL start_point
     END
```

```
/* */
/* now create the table. Max entries 99999 */
/* */
'TBCREATE MEMCHA NAMES(MEMBER) NOWRITE REPLACE'
DO x=1 TO 99999
IF MEMBER.x='' THEN LEAVE
member=STRIP(MEMBER.x)
'TBADD MEMCHA'
END
'TBTOP MEMCHA'
'ADDPOP ROW(1) COLUMN(9)'
'TBDISPL MEMCHA PANEL(SELCHAP2)'
/* */
/* loop around possible selections from the table and provide a view */
/* of the member. Note if PF3 is pressed we return to front panel.   */
/* */
in_point:
IF reply='END' THEN SIGNAL leave_point
IF ztdsels=0 THEN 'TBDISPL MEMCHA'
IF ztdsels = 1 THEN DO
         'VIEW DATASET('dsname'('member'))'
         'TBDISPL MEMCHA'
END
IF ztdsels > 1 THEN DO UNTIL ztdsels=1
          member=STRIP(member)
          'VIEW DATASET('dsname'('member'))'
          'TBDISPL MEMCHA'
END
SIGNAL in_point
leave_point:
'ISPEXEC REMPOP'
SIGNAL start_point
```

C A Jacques
Systems Programmer (UK)

Displaying devices in TSO

To retrieve information about a system device, DASD, tape, etc, you normally have to go to a system console (or SDSF log) and issue the MVS command DISPLAY U. You must also have an idea of the unit address you are looking for, since MVS by default retrieves only 16 addresses at a time.

To get around this, I have written a program that scans the entire unit chain, by means of the UCBSCAN Assembler macro. The program module can be called from a TSO prompt. The results are presented by means of a full-screen 3270 datastream (no ISPF involved). You can specify the type of device you want or the starting address, but by default you browse through the entire chain for a specific device type.

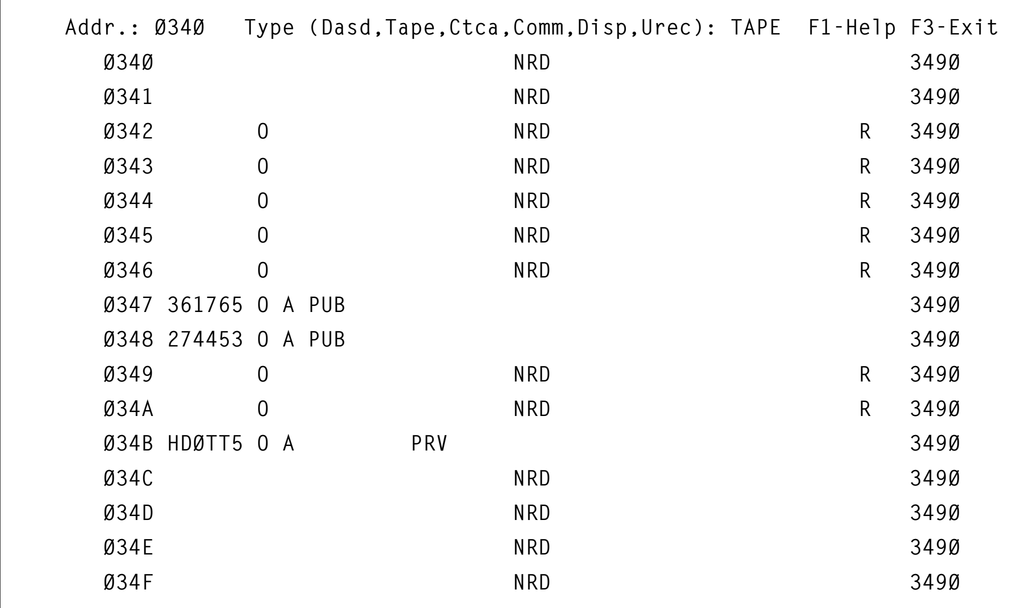
In Figure 1, there is an example of the program's output screen. By default, the device type is DASD, and the device address is the lowest for that class. Hitting 'Enter' will browse you through the chain, in a circular fashion. In the Addr.: field you can input an address to jump directly to it. In the 'TYPE' field, choose the device type you want. Any invalid type defaults to DASD.

The HELP screen is shown in Figure 3, and is available by pressing F1/F13. As far as possible, the mnemonics used are similar to those of the DU command. For a more detailed description of each mnemonic/ device condition and why they occur, see the *MVS Data Areas* manual for the mapping macros referred in the beginning of the program, or see the comments written in the macros themselves.

For more details on how to create 3270 data streams, see *MVS Update*

```
Addr.: Ø19Ø   Type (Dasd,Tape,Ctca,Comm,Disp,Urec): DASD  F1-Help F3-Exit
   Ø19Ø DS1C9Ø O A         PRV RSD
   Ø191 DS1C91 O A         PRV RSD
   Ø192                        RSD NRD
   Ø193                        RSD NRD
   Ø194                        RSD NRD
   Ø2ØØ DS22ØØ O A         PRV RSD
   Ø2Ø1 DS2AØ3 O A         PRV RSD
   Ø2Ø2 DS2AØ4 O A         PRV RSD
   Ø2Ø3                        RSD
   Ø2Ø4                        RSD
   Ø2Ø5                        RSD
   Ø2Ø6                        RSD
   Ø2Ø7 DS2BØ7 O A         PRV RSD
   Ø2Ø8 DS2BØ8 O A         PRV RSD
   Ø2Ø9                        RSD
   Ø2ØA DS2BX2 O A     STO     RSD
   Ø2ØB                        RSD
   Ø2ØC                        RSD
```

Figure 1: The DASD screen



```
Addr.: Ø34Ø   Type (Dasd,Tape,Ctca,Comm,Disp,Urec): TAPE  F1-Help F3-Exit
   Ø34Ø                            NRD                            349Ø
   Ø341                            NRD                            349Ø
   Ø342        0                   NRD                        R   349Ø
   Ø343        0                   NRD                        R   349Ø
   Ø344        0                   NRD                        R   349Ø
   Ø345        0                   NRD                        R   349Ø
   Ø346        0                   NRD                        R   349Ø
   Ø347 361765 0 A PUB                                            349Ø
   Ø348 274453 0 A PUB                                            349Ø
   Ø349        0                   NRD                        R   349Ø
   Ø34A        0                   NRD                        R   349Ø
   Ø34B HDØTT5 0 A         PRV                                    349Ø
   Ø34C                            NRD                            349Ø
   Ø34D                            NRD                            349Ø
   Ø34E                            NRD                            349Ø
   Ø34F                            NRD                            349Ø
```

Figure 2: The TAPE screen

124, page 50, or GA23-0059, *IBM 3270 Information Display System Data Stream Programmer's Reference*.

SCANUNIT SOURCE CODE

```
*********************************************************************
*         ScanUnit - Scan Unit Control Blocks for the following     *
*         device classes: DASD, Tape, Communications, Unit Record,  *
*         Channel-to-Channel and Graphics (display). Supports       *
*         4-digit and dynamically allocated units.                  *
*         Mapping macros: IEFUCBOB for the common segment,          *
*         IECDUCBG, IECDUCBT, IECDUCBC, IECDUCBD, IECDUCBE for the  *
*         device-specific segments (SYSL.MACLIB and SYSL.MOGDEN).   *
*********************************************************************
SCANUNIT CSECT
SCANUNIT Amode 31
SCANUNIT Rmode 24                  Must be 24 because of TPUT macro
         SAVE  (14,12)             Standard initial stuff
         LR    R12,R15
         USING SCANUNIT,R12
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     FOLLOW
```

```
          A         Allocated
          AP        Attention pending
          BOX       Boxed. Forced offline due to error
          BLP       Bypass Label Processing (Tape)
          H         Hot I/O, device boxed or not recovered yet
          IP        Intercept condition pending
          IR        Intervention required message issued
          MP        Mount message pending, but message not issued
          LABELNS   Label not Standard (Tape)
          NP        No operational paths
          NRD       Not Ready
          NS        No subchannel connected
          O         Online
          PRV       Private
          PS        Pending sense operation
          PUB       Public
          R         Intercept condition requires ERP processing
          RSD       Permanently resident
          STO       Storage
          SYS       Device used by system component. Status cannot change
          US        Subchannel for the device is unusable
    Device Types: DASD - Dasd   UREC - Unit Record  CTCA - Channel to Channel
                  TAPE - Tape   DISP - Display      COMM - Communications
```

Figure 3: The HELP screen

```
         DC    CL8'&SYSTIME'
         DC    CL8'&SYSDATE'
FOLLOW   EQU   *
         STORAGE OBTAIN,LENGTH=48,ADDR=(R1Ø)
         STORAGE OBTAIN,LENGTH=21ØØ,ADDR=(R11)
         USING UCBDSECT,R1Ø
         USING DISP327Ø,R3
         MVC   Ø(S327ØL,R11),S327Ø Move Screen Header
         LR    R3,R11
         MVC   FIELDIN2,FIELD2
         BAL   R6,SETTYPE          Set default unit type
         SR    R9,R9
******** Screen Loop **********************************************
LOOPØ    EQU   *
         LR    R3,R11
         LH    R8,=H'23'           Set line counter
         LA    R4,LINES            Clear screen area
         LH    R5,=H'184Ø'         23 lines length (23*8Ø)
         SR    R7,R7
         MVCL  R4,R6
         LTR   R9,R9               First time?
         BE    LOOP23              Yes, jump
         CLC   FIELDIN2,FIELD2     Different unit type entered?
         BE    LOOPØA              No, jump
         BAL   R6,SETTYPE          Set new unit type
LOOPØA   EQU   *
         CLC   FIELDIN1,FIELD1     Different start address?
         BE    LOOP23              No, jump
         MVC   WORK1(1ØØ),WORKØ    Clear UCBSCAN workareas
         MVC   Ø(48,R1Ø),WORKØ
         OC    FIELDIN1,=X'4Ø4Ø4Ø4Ø' Uppercase input field 1
         CLC   FIELDIN1,NFIRST     Input lower than first unit?
         BNL   SEARCHØ             No, jump
         MVC   FIELDIN1,NFIRST     Replace user input with first unit
SEARCHØ  EQU   *
         CLI   FLD11,C'Ø'          Lower EBCDIC numeric chars to allow
         BL    *+8                 comparisons with hexadecimal A-F.
         NI    FLD11,X'ØF'
         CLI   FLD12,C'Ø'
         BL    *+8
         NI    FLD12,X'ØF'
         CLI   FLD13,C'Ø'
         BL    *+8
         NI    FLD13,X'ØF'
         CLI   FLD14,C'Ø'
         BL    *+8
         NI    FLD14,X'ØF'
SEARCH1  EQU   *
         BAL   R9,UCBEXE           Get next UCB
         MVC   NAME1,UNITNAME
         CLI   NAME11,C'Ø'         Lower NAME1 numeric chars as above
         BL    *+8
         NI    NAME11,X'ØF'
         CLI   NAME12,C'Ø'
         BL    *+8
         NI    NAME12,X'ØF'
         CLI   NAME13,C'Ø'
         BL    *+8
         NI    NAME13,X'ØF'
         CLI   NAME14,C'Ø'
         BL    *+8
         NI    NAME14,X'ØF'
         CLC   NAME1,FIELDIN1      Now we can compare,
         BE    LOOP23A             until found or greater than
         BL    SEARCH1
******** Line Loop ***************************************************
LOOP23   EQU   *
         BAL   R9,UCBEXE           Get next UCB
******** UCB common segment codes (for all devices) *****************
LOOP23A  EQU   *
         MVC   NAME,UNITNAME       UCB Address
         TM    UCBJBNR,UCBMMSGP
         BZ    *+1Ø
         MVC   MMSGP,=C'MP'        Mount message pending
         TM    UCBFL5,UCBNALOC
         BZ    *+1Ø
         MVC   NALOC,=C'SYS'       Device used by system component
         TM    UCBSTAT,UCBONLI
         BZ    *+1Ø
         MVC   ONLI,=C'O'          On-line
         TM    UCBSTAT,UCBALOC
         BZ    *+1Ø
         MVC   ALOC,=C'A'          Allocated
         TM    UCBSTAT,UCBPRES
         BZ    *+1Ø
         MVC   PRSD,=C'RSD'        Permanently resident
         TM    UCBFLA,UCBNRY
         BZ    *+1Ø
         MVC   NRY,=C'NRD'         Not Ready
         TM    UCBFLA,UCBPERM
         BZ    *+1Ø
         MVC   PERM,=C'US'         Unusable subchannel
         TM    UCBFLA,UCBPSNS
         BZ    *+1Ø
         MVC   PSNS,=C'PS'         Pending sense operation
         TM    UCBFLA,UCBBOX
         BZ    *+1Ø
         MVC   BOX,=C'BOX'         Boxed
         TM    UCBFLB,UCBINCPT
         BZ    *+1Ø
         MVC   INCPT,=C'R'         ERP request
         TM    UCBFLB,UCBNOPTH
         BZ    *+1Ø
         MVC   NOPTH,=C'NP'        No operational paths
         TM    UCBFLB,UCBNOCON
         BZ    *+1Ø
         MVC   NOCON,=C'NS'        No subchannel connected
         TM    UCBFLB,UCBHDET
         BZ    *+1Ø
         MVC   HDET,=C'H'          Hot I/O, boxed or not recovered
         TM    UCBFLC,UCBATTP
         BZ    *+1Ø
         MVC   ATTP,=C'AP'         Attention pending
         TM    UCBFLC,UCBITFP
         BZ    *+1Ø
         MVC   ITFP,=C'IP'         Intercept condition pending
         TM    UCBFLC,UCBIVRS
         BZ    *+1Ø
         MVC   IVRS,=C'IR'         Intervention required message
******** Device dependent specific codes ****************************
         CLI   UTYPE,UCB3DACC      Find out what specific device
         BE    DASDTAPE            it is and jump to the
         CLI   UTYPE,UCB3TAPE      appropriate label
         BE    DASDTAPE
         CLI   UTYPE,UCB3CTC
         BE    CTC
         CLI   UTYPE,UCB3UREC
         BE    UREC
         CLI   UTYPE,UCB3COMM
         BE    COMM
         B     NEXTØ
DASDTAPE EQU   *                   DASD and Tape codes
         MVC   VOLI,UCBVOLI        Label / Volser
         TM    UCBSTAB,UCBBSTR
         BZ    *+1Ø
         MVC   BSTR,=C'STO'        Storage
         TM    UCBSTAB,UCBBPUB
         BZ    *+1Ø
         MVC   BPUB,=C'PUB'        Public
         TM    UCBSTAB,UCBBPRV
         BZ    *+1Ø
         MVC   BPRV,=C'PRV'        Private
         CLI   UTYPE,UCB3TAPE
         BNE   NEXTØ
TAPE     EQU   *                   Tape only codes
         TM    UCBTFL1,UCBNLTP
         BZ    *+14
         MVC   SPECIFIC,=C'NOLABEL' Tape has no label
         BAL   R9,NOTRASH          Avoid trash in label
         TM    UCBTFL1,UCBNSLTP    Label is not standard
         BZ    *+14
         MVC   SPECIFIC,=C'LABELNS'
         BAL   R9,NOTRASH
         TM    UCBTFL1,UCBBLP      Bypass Label Processing specified
         BZ    *+14
         MVC   SPECIFIC,=C'BLP    '
         BAL   R9,NOTRASH
         TM    UCBTBYT4,UCB34ØØ    Tape models
         BZ    *+1Ø
         MVC   SPECIFIC,=C'34ØØ   '
         TM    UCBTBYT4,UCB348Ø
         BZ    *+1Ø
         MVC   SPECIFIC,=C'348Ø   '
         TM    UCBTBYT4,UCB349Ø
         BZ    *+1Ø
         MVC   SPECIFIC,=C'349Ø   '
         B     NEXTØ
CTC      EQU   *                   CTC only codes
         CLI   UCBTBYT4,UCBPCTC
         BZ    *+1Ø
         MVC   SPECIFIC,=C'PARALEL'
         CLI   UCBTBYT4,UCBSCTC
         BZ    *+1Ø
         MVC   SPECIFIC,=C'SERIAL '
         CLI   UCBTBYT4,UCBBCTC
         BZ    *+1Ø
         MVC   SPECIFIC,=C'B ESCON'
         CLI   UCBTBYT4,UCBRS6K
         BZ    *+1Ø
         MVC   SPECIFIC,=C'RS6ØØØ '
         CLI   UCBTBYT4,UCB3172
         BZ    *+1Ø
         MVC   SPECIFIC,=C'3172   '
         CLI   UCBTBYT4,UCBOSA
         BZ    *+1Ø
         MVC   SPECIFIC,=C'OSA    '
         B     NEXTØ
UREC     EQU   *                    Urec only codes
         CLI   UCBTBYT4,UCB38ØØ     Unit record models
         BZ    *+1Ø
         MVC   SPECIFIC,=C'38ØØ   '
         CLI   UCBTBYT4,UCB3838
         BZ    *+1Ø
         MVC   SPECIFIC,=C'3838   '
         CLI   UCBTBYT4,UCB3895
         BZ    *+1Ø
         MVC   SPECIFIC,=C'3895   '
         CLI   UCBTBYT4,UCB4245
         BZ    *+1Ø
         MVC   SPECIFIC,=C'4245   '
         CLI   UCBTBYT4,UCB4248
         BZ    *+1Ø
         MVC   SPECIFIC,=C'4248   '
         B     NEXTØ
COMM     EQU   *                    Comm only codes
         CLI   UCBTBYT4,UCB3791L    Communications models
         BZ    *+1Ø
         MVC   SPECIFIC,=C'3791L  '
         CLI   UCBTBYT4,UCB42AD1
         BZ    *+1Ø
         MVC   SPECIFIC,=C'27Ø2   '
         B     NEXTØ
******** End of device dependent specific codes ********************
NEXTØ    EQU   *
         CH    R8,=H'23'            First line?
         BNE   NEXT1                No, jump
         MVC   FIELD1,NAME          Move first line unit to input
         CLI   NFIRST,X'4Ø'         First unit saved in nfirst?
         BNE   NEXT1                Yes, jump
         MVC   NFIRST,NAME          Keep first unit of current type
NEXT1    AH    R3,=H'8Ø'            Force R3 to point next line
         BCT   R8,LOOP23            Loop next line
******** Screen Display ********************************************
DISPLAY  EQU   *
         LA    R8,F327ØL            Screen length
         LA    R7,DISPIN            Input area
         STFSMODE ON,INITIAL=YES    Send 327Ø datastream
         STTMPMD  ON
         STLINENO LINE=1
         TPUT  (R11),(R8),FULLSCR,,HOLD
         L     R8,=F'2Ø'            Max input length allowed
         TGET  DISPIN,(R8),ASIS     Get 327Ø datastream
         STFSMODE OFF
         STTMPMD  OFF
         CLI   PFKEY,C'3'           PF3 or PF15, Exit
         BE    EXIT
         CLI   PFKEY,C'C'
         BE    EXIT
         CLI   PFKEY,C'1'           PF1 or PF13, send help screen
         BE    DISPHELP
         CLI   PFKEY,C'A'
         BE    DISPHELP
         B     LOOPØ                Loop for next screen
DISPHELP EQU   *                    Send help screen
         STFSMODE ON,INITIAL=YES
         STTMPMD  ON
         STLINENO LINE=1
         LA    R7,H327ØL
         TPUT  H327Ø,(R7),FULLSCR,,HOLD
         TGET  DUMMYG,1             Dummy get
         STFSMODE OFF
         STTMPMD  OFF
         B     DISPLAY              Redisplay current data screen
******** Free storage and exit **************************************
EXIT     EQU   *
         STORAGE RELEASE,LENGTH=48,ADDR=(R1Ø)
         STORAGE RELEASE,LENGTH=21ØØ,ADDR=(R11)
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         SR    R15,R15
         BR    R14
*********************************************************************
*        Subroutines
*********************************************************************
UCBEXE   EQU   *
         UCBSCAN WORKAREA=WORK1,     Scan UCB's                        *
               UCBAREA=(R1Ø),                                          *
               RANGE=ALL,                                              *
               DYNAMIC=YES,                                            *
               DEVNCHAR=UNITNAME,                                      *
               DEVCID=UTYPE
         CH    R15,=H'4'             End-of scan?
         BE    DISPLAY               Yes, send screen
         BR    R9                    Return
SETTYPE  EQU   *                     Set current type
         MVC   WORK1(1ØØ),WORKØ      Clear UCBSCAN work areas
         MVC   Ø(48,R1Ø),WORKØ
         MVC   NFIRST,=X'4Ø4Ø4Ø4Ø'  Clear first unit addr
         MVC   FIELD1,=C'AAAA'
         MVC   FIELDIN1,=C'AAAA'
         MVC   FIELD2,FIELDIN2
         OC    FIELD2,=X'4Ø4Ø4Ø4Ø'  Uppercase user input
         CLC   FIELD2,=C'DASD'      Compare to user input and set type
         BNE   SETTYPE1
SETTYPEØ MVI   UTYPE,UCB3DACC
         BR    R6                   Return
SETTYPE1 CLC   FIELD2,=C'CTCA'
         BNE   SETTYPE2
         MVI   UTYPE,UCB3CTC
         BR    R6
SETTYPE2 CLC   FIELD2,=C'TAPE'
         BNE   SETTYPE3
         MVI   UTYPE,UCB3TAPE
         BR    R6
SETTYPE3 CLC   FIELD2,=C'UREC'
         BNE   SETTYPE4
         MVI   UTYPE,UCB3UREC
         BR    R6
SETTYPE4 CLC   FIELD2,=C'COMM'
         BNE   SETTYPE5
         MVI   UTYPE,UCB3COMM
         BR    R6
SETTYPE5 CLC   FIELD2,=C'DISP'
         BNE   SETTYPEØ             If user input unknown, force DASD
         MVI   UTYPE,UCB3DISP
         BR    R6
NOTRASH  EQU   *                    Replace strange characters in
         CLI   VOLI,X'4Ø'           non-standard tape labels by dots
         BL    *+8                  (Anything lower than x'4Ø').
         MVI   VOLI,X'4B'           Otherwise, the 327Ø datastream
         CLI   VOLI+1,X'4Ø'         may be trashed.
         BL    *+8
         MVI   VOLI+1,X'4B'
         CLI   VOLI+2,X'4Ø'
         BL    *+8
         MVI   VOLI+2,X'4B'
         CLI   VOLI+3,X'4Ø'
         BL    *+8
         MVI   VOLI+3,X'4B'
         CLI   VOLI+4,X'4Ø'
         BL    *+8
         MVI   VOLI+4,X'4B'
         CLI   VOLI+5,X'4Ø'
         BL    *+8
         MVI   VOLI+5,X'4B'
         BR    R9
*********************************************************************
```

```
*        Static work areas
*********************************************************************
SAVEA    DS    18F                  Register area
DUMMYG   DS    C                    Dummy get for Help screen 'Enter'
UTYPE    DS    C                    Current unit type bit mask
NFIRST   DC    C'    '              First unit addr for each type
UNITNAME DS    CL4                  EBCDIC unit name
WORK1    DC    1ØØX'ØØ'             UCBSCAN work area
WORKØ    DC    1ØØX'ØØ'
NAME1    DS    ØCL4
NAME11   DS    C
NAME12   DS    C
NAME13   DS    C
NAME14   DS    C
DISPIN   DS    ØF                   327Ø screen data input area
PFKEY    DS    C                    PFkey code
         DS    CL5                  Filler for addresses
FIELDIN1 DS    ØCL4                 First field (unit addr)
FLD11    DS    C
FLD12    DS    C
FLD13    DS    C
FLD14    DS    C
         DS    CL3                  Filler for address
FIELDIN2 DS    CL4                  Second field (unit type)
         DS    CL4
H327Ø    EQU   *                    Help screen
         DC    X'F51DF8'
         DC    X'114ØC6'
         DC    C' A        '
         DC    X'1DFØ'
         DC    C'Allocated'
         DC    X'1141D51DF8'
         DC    C' AP       '
         DC    X'1DFØ'
         DC    C'Attention pending'
         DC    X'1142E51DF8'
         DC    C' BOX      '
         DC    X'1DFØ'
         DC    C'Boxed. Forced off-line due to error'
         DC    X'1143F51DF8'
         DC    C' BLP      '
         DC    X'1DFØ'
         DC    C'Bypass Label Processing (Tape)'
         DC    X'1145C51DF8'
         DC    C' H        '
         DC    X'1DFØ'
         DC    C'Hot I/O, device boxed or not recovered yet'
         DC    X'1146D51DF8'
         DC    C' IP       '
         DC    X'1DFØ'
         DC    C'Intercept condition pending'
         DC    X'1147E51DF8'
         DC    C' IR       '
         DC    X'1DFØ'
         DC    C'Intervention required message issued'
         DC    X'1148F51DF8'
         DC    C' MP       '
         DC    X'1DFØ'
         DC    C'Mount message pending, but message not issued'
         DC    X'114AC51DF8'
         DC    C' LABELNS  '
         DC    X'1DFØ'
         DC    C'Label not Standard (Tape)'
         DC    X'114BD51DF8'
         DC    C' NP       '
         DC    X'1DFØ'
         DC    C'No operational paths'
         DC    X'114CE51DF8'
         DC    C' NRD      '
         DC    X'1DFØ'
         DC    C'Not Ready'
         DC    X'114DF51DF8'
         DC    C' NS       '
         DC    X'1DFØ'
         DC    C'No subchannel connected'
         DC    X'114FC51DF8'
         DC    C' O        '
         DC    X'1DFØ'
         DC    C'Online'
         DC    X'115ØD51DF8'
         DC    C' PRV      '
         DC    X'1DFØ'
         DC    C'Private'
         DC    X'1151E51DF8'
         DC    C' PS       '
         DC    X'1DFØ'
         DC    C'Pending sense operation'
         DC    X'1152F51DF8'
         DC    C' PUB      '
         DC    X'1DFØ'
         DC    C'Public'
         DC    X'1154C51DF8'
         DC    C' R        '
         DC    X'1DFØ'
         DC    C'Intercept condition requires ERP processing'
         DC    X'1155D51DF8'
         DC    C' RSD      '
         DC    X'1DFØ'
         DC    C'Permanently resident'
         DC    X'1156E51DF8'
         DC    C' STO      '
```

```
         DC    X'1DFØ'
         DC    C'Storage'
         DC    X'1157F51DF8'
         DC    C' SYS      '
         DC    X'1DFØ'
         DC    C'Device used by system component. Status cannot change'
         DC    X'1159C51DF8'
         DC    C' US       '
         DC    X'1DFØ'
         DC    C'Subchannel for the device is unusable'
         DC    X'115ADØ1DF8'
         DC    C'Device Types: DASD - DASD   UREC - Unit Record'
         DC    C'   CTCA - Channel to Channel'
         DC    X'115BEF'
         DC    C'TAPE - Tape   DISP - Display       '
         DC    C'COMM - Communications'
H327ØL   EQU   *-H327Ø              Help screen length
S327Ø    EQU   *                    Data screen header
         DC    X'4Ø'
         DC    X'114ØØ813'          Cursor position
         DC    X'114Ø4Ø1DF8'
         DC    C'Addr.:'
         DC    X'1DC9'
FLD1     DC    CL4'ØØØØ'
         DC    X'1DF8'
         DC    C'  Type (Dasd,Tape,Ctca,Comm,Disp,Urec):'
         DC    X'1DC9'
FLD2     DC    CL4'DASD'            Default device type goes here
         DC    X'1DF8'
         DC    C' F1-Help F3-Exit'
         DC    X'1141DØ1DFØ'        Set line two address
FLD3     EQU   *
S327ØL   EQU   *-S327Ø              Header length
F327ØL   EQU   S327ØL+(8Ø*23)       Screen length (header+23 lines)
**********************************************************************
*        Acquired storage
**********************************************************************
DISP327Ø DSECT                      Display screen area
         DS    CL(FLD1-S327Ø)       Header datastream is copied here.
FIELD1   DS    CL4                  Ensure input fields are in
         DS    CL(FLD2-FLD1-4)      position
FIELD2   DS    CL4
         DS    CL(FLD3-FLD2-4)
LINES    DS    ØC                   Detail lines layout
NAME     DS    CL4
         DS    CL1
VOLI     DS    CL6
         DS    CL1
ONLI     DS    CL1
         DS    CL1
ALOC     DS    CL1
```

```
         DS    CL1
BPUB     DS    CL3
         DS    CL1
BSTR     DS    CL3
         DS    CL1
BPRV     DS    CL3
         DS    CL1
PRSD     DS    CL3
         DS    CL1
NRY      DS    CL3
         DS    CL1
BOX      DS    CL3
         DS    CL1
IVRS     DS    CL2
         DS    CL1
MMSGP    DS    CL2
         DS    CL1
NALOC    DS    CL3
         DS    CL1
ATTP     DS    CL2
         DS    CL1
ITFP     DS    CL2
         DS    CL1
PSNS     DS    CL2
         DS    CL1
INCPT    DS    CL1
         DS    CL1
HDET     DS    CL1
         DS    CL1
SPECIFIC DS    CL7
NOPTH    DS    CL2
         DS    CL1
NOCON    DS    CL2
         DS    CL1
PERM     DS    CL2
UCBDSECT DSECT
         IEFUCBOB                   Map UCB areas
         YREGS                      Register equates
         END
```

Luis Paulo Figueiredo Sousa Ribeiro
Systems Programmer
Edinfor (Portugal)

Access authority checking of implicit HSM recalls

BACKGROUND

Under ISPF/PDF, when an attempt is made to perform an action such as BROWSE or EDIT on a dataset that is migrated, PDF issues an implicit recall request to DFSMShsm, prior to dynamically allocating and opening that dataset. This recall request is not subject to any access authority checking, and so even a user who has ACCESS(NONE) to the dataset can cause it to be recalled. Of course, once the recall is complete, the OPEN will fail with the standard 913-38 ABEND, so in terms of actual access to the data this is not a security problem. In the meantime, however, a potentially large waste of resources (DFSMShsm CPU time, possibly one or more tape mounts, and occupation of Level 0 DASD) has occurred to no good purpose.

It is interesting to note that, if the user issues an explicit HRECALL request for a protected dataset, DFSMShsm does make an access authority check, and will reject the request with the messages:

```
ARC1001I dsname  RECALL FAILED, RC=0039, REAS=0008
ARC1139I ERROR PROCESSING RACF PROTECTED DATASET, RECALL TERMINATED
```

When tackled over this seeming inconsistency, IBM responded by stating that everything was ‘working as designed’; the justification being that when PDF makes the implicit recall request the user should not be aware of it and would not be expecting DFSMShsm error messages in the event of a recall failure, whereas he/she could reasonably be presented with such messages in response to an explicit HRECALL command.

THE PROBLEM

At the site where I work, there is a class of datasets, the majority of which are migrated, which we wish only to be recallable by their owner. As described above, IBM has by design made it impossible to enforce this requirement. Accordingly, I have implemented a pre-recall authorization checking mechanism, by front-ending the DFSMShsm SVC, IGX00024.

SVC interception is an approved process (see for example *MVS/ESA Programming: Authorised Assembler Services Guide*) and is normally achieved via the SVCUPDTE macro, which changes the address of the SVC routine in the SVC Table. IGX00024 is, however, an extended SVC, reached via SVC 109, routecode 24, and cannot be replaced via SVCUPDTE. Investigation showed that the SVC Table entry for SVC 109 is in fact the address of the ESR Table, which contains the addresses of the extended SVC routines. The ESR Table is in the read-only nucleus, and as such is page-protected, but under MVS/ESA this protection can easily be bypassed by a supervisor-state program using the STURA (Store Using Real Address) instruction.

THE SOLUTION

Our IGX00024 front end consists of two routines. The first, IGX24INT, runs as a transient started task to either install or remove the front end routine itself, IGX24CHK.

In install mode, IGX24INT opens the load library containing IGX24CHK, does a BLDL for it, and verifies that the load module has the attributes appropriate to a type 3 SVC. If OK, it acquires key 0 fixed ECSA storage (subpool 228) for IGX24CHK and LOADs it into that storage. It then locates the ESR Table entry for IGX00024 and verifies that it has not already been intercepted. If OK, it issues a WTOR requesting permission to update the ESR table. If given, it stores the address of IGX00024 into IGX24CHK, obtains and verifies the real address of the ESR Table entry for IGX00024, and updates the entry with the address of IGX24CHK using a STURA instruction. The SYSZSVC TABLE resource is used to serialize the update.

In delete mode, IGX24INT locates the ESR table entry for IGX00024 and issues a WTOR requesting permission to remove the intercept. If given, it verifies that the ESR table entry does point at IGX24CHK, and if, OK , extracts the address of IGX00024 from it. The real address of the ESR Table entry for IGX00024 is obtained and verified, and if OK is updated with the address of IGX00024, again using STURA. The SYSZSVC TABLE resource is used to serialize the update. The ECSA storage containing it is not deleted, in case the IGX24CHK code is in active use at that moment.

The processing in IGX24CHK is designed for minimum overhead. If the HSM request (MWE) is not for a recall, control passes directly to IGX00024. If the caller is a started task or a user with the RACF OPERATIONS attribute, control passes directly to IGX00024. If the caller is a non-privileged batch job or TSU, and the dataset prefix is the caller's RACF uid, control passes directly to IGX00024. In the remaining cases of a non-privileged batch job or TSU requesting a recall of someone else's dataset, a RACROUTE REQUEST=AUTH is issued. If access is allowed, control passes to IGX00024, otherwise a console message is issued, the MWE is filled in as if DFSMShsm had processed it and rejected it with MWERC=39, MWEREASN=RACF rc, and control returns to the issuer of the request. This detects the non-zero MWERC and generates the messages shown above.

IGX24INT should be link-edited into an APF-Authorized library with AC(1). For convenience, IGX24CHK can also be placed in the same library, but has no explicit authorization requirements. The JCL to run IGX24INT should be placed in a suitable system PROCLIB.

The code presented here runs under MVS/ESA 5.1.0 and DFSMS/MVS 1.2.0, but there is no reason why it should not work with earlier versions of MVS/ESA.

JCL FOR THE IGX24INT STARTED TASK

```
//IGX24INT PROC ACTION=
//*****************************************************************//
//*                                                               *//
//*    INSTALL OR DELETE HSM SVC FRONT END INTERCEPT ROUTINE      *//
//*                                                               *//
//*    TO INSTALL INTERCEPT : S IGX24INT,ACTION=INSTALL           *//
//*                                                               *//
//*    TO REMOVE  INTERCEPT : S IGX24INT,ACTION=DELETE            *//
//*                                                               *//
//*****************************************************************//
//IGX24INT EXEC PGM=IGX24INT,PARM='&ACTION'
//STEPLIB  DD   DSN=SYS1.APFLIB,DISP=SHR
//SVCLIB   DD   DSN=SYS1.APFLIB,DISP=SHR
//SYSUDUMP DD   SYSOUT=X
//*
```

SOURCE CODE FOR IGX24INT

```
TITLE 'IGX24INT - HSM SVC Front End Intercept Installer'
***********************************************************************
*  PROGRAM IGX24INT                                                   *
*  ----------------                                                   *
*  Installs or deletes the HSM communication SVC  (SVC 1Ø9 route code *
*  24 - IGXØØØ24) front end intercept.                                *
*  This program must be link-edited with AC(1) into an APF-Authorized *
*  library.                                                           *
*   ENVIRONMENT                                                       *
*     STATE    : PROBLEM & SUPERVISOR                                 *
*     KEY      : 8 & Ø                                                *
*     APF      : YES                                                  *
*     AMODE    : 31                                                   *
*     RMODE    : 24                                                   *
*     LOCATION : PRIVATE AREA                                         *
***********************************************************************
         EJECT
IGX24INT CSECT
IGX24INT AMODE 31
IGX24INT RMODE 24
RØ       EQU   Ø                       * REASON CODE ON RETURN
R1       EQU   1                       * WORK REGISTER
R2       EQU   2                       * WORK REGISTER
R3       EQU   3                       * @(SVCEP) FROM SVC TABLE
R4       EQU   4                       *
R5       EQU   5                       *
R6       EQU   6                       *
R7       EQU   7                       *
R8       EQU   8                       * REASON CODE
R9       EQU   9                       * RETURN CODE
R1Ø      EQU   1Ø                      * @(SVC ENTRY POINT)
R11      EQU   11                      * @(SVCLIB DCB)
R12      EQU   12                      * BASE REGISTER
R13      EQU   13                      * OUR SAVEAREA
R14      EQU   14                      * RETURN ADDRESS
R15      EQU   15                      * ENTRY ADDRESS/RETURN CODE
         USING *,R15                   * ADDRESSABILITY
         B     START                   * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)       * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    CL8'IGX24INT'
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                     * FINISHED WITH R15
         DS    ØF                      * ALIGN TO FULL WORD BOUNDARY
```

```
***********************************************************************
*  ADDRESSABILITY AND LINKAGE
***********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER
         USING IGX24INT,R12             * AND DEFINE ADDRESSIBILITY
         LR    R11,R13                  * R11 = ADDRESS OF HSA
         LA    R13,SAVEAREA             * R13 = ADDRESS OF LSA
         ST    R11,4(R13)               * STORE HSA ADDRESS
         ST    R13,8(R11)               * STORE LSA ADDRESS
* EXTRACT PARM AND DETERMINE ACTION
         L     R1,Ø(R1)                 * R1 = @(PARM FIELD)
         LH    R2,Ø(R1)                 * R2 = L'(PARM FIELD)
         LA    R8,4                     * PRESET REASON CODE = 4
         LTR   R2,R2                    * CAN'T CONTINUE ...
         BZ    BADPARM                  * ... IF L'PARM = Ø
         CLI   2(R1),C'I'               * PARM=I{NSTALL}
         BE    INSTALL                  * YES, INSTALL INTERCEPT
         CLI   2(R1),C'D'               * PARM=D{ELETE}
         BE    DELETE                   * YES, DELETE INTERCEPT
         B     BADPARM                  * ANYTHING ELSE IS BAD
         EJECT
***********************************************************************
* PARM=INSTALL : INSTALL THE IGXØØØ24 FRONT END INTERCEPT ROUTINE
***********************************************************************
* OPEN THE LOAD LIBRARY CONTAINING IGC24CHK
INSTALL  EQU   *
         LA    R11,SVCLIB               * R11 = DCB ADDRESS
         USING IHADCB,R11               * DEFINE DCB ADDRESSABILITY
         OPEN  ((R11),INPUT),MODE=31    * OPEN LOAD LIBRARY
         TM    DCBOFLGS,DCBBIT3         * BIT 3 SHOULD BE 1
         BZ    BADOPEN                  * ITS NOT SO AN ERROR OCCURRED
* BUILD THE PDS DIRECTRY ENTRY FOR IGX24CHK
         LA    R1,BIT24                 * SWITCH ...
         LA    R2,BIT31                 * ... TO ...
         BSM   R2,R1                    *     ... AMODE 24
BIT24    EQU   *
         BLDL  (R11),BLDLLIST           * GET THE DIRECTORY ENTRY
         BSM   RØ,R2                    * REVERT TO AMODE 31
BIT31    EQU   *
         LTR   R8,R15                   * BLDL OK ?
         BNZ   BADBLDL                  * QUIT IF NOT
* VERIFY THAT THE MODULE HAS THE CORRECT ATTRIBUTES
         LA    R8,4                     * PRESET ERROR REASON = 4
         TM    BLARMODE,BLAM31          * IGX24CHK MUST HAVE ...
         BNO   BADLMOD                  * ... AMODE = 31
         TM    BLARMODE,BLRMANY         * IGX24CHK MUST HAVE ...
         BNO   BADLMOD                  * ... RMODE = ANY
         LA    R8,8                     * PRESET ERROR REASON = 8
         TM    BLMATTR1,BLA1RENT        * IGX24CHK MUST BE ...
         BNO   BADLMOD                  * ... REENTRANT
         LA    R8,12                    * PRESET ERROR REASON = 12
         TM    BLMATTR1,BLA1REUS        * IGX24CHK MUST BE ...
         BNO   BADLMOD                  * ... REUSABLE
         LA    R8,16                    * PRESET ERROR REASON = 16
         TM    BLMATTR1,BLA1EXEC        * IGX24CHK MUST BE ...
         BNO   BADLMOD                  * ... EXCUTABLE
         LA    R8,2Ø                    * PRESET ERROR REASON = 2Ø
         TM    BLMATTR1+1,BLA1REFR      * IGX24CHK MUST BE ...
         BNO   BADLMOD                  * ... REFRESHABLE
* GETMAIN KEY ZERO STORAGE IN FIXED ECSA FOR THE MODULE AND LOAD THE
* IGX24CHK MODULE INTO THIS STORAGE
         SR    R2,R2                    * GET SIZE ...
         ICM   R2,B'Ø111',BLMODLN       * ... OF MODULE
         MODESET KEY=ZERO               * GET INTO KEY ZERO
         GETMAIN RU,LV=(R2),LOC=ANY,SP=228
         LR    R1Ø,R1                   * SAVE @(IGX24CHK)
         LOAD  DE=BLNAME,DCB=(R11),ADDR=(R1Ø)
         MODESET KEY=NZERO              * REVERT TO KEY 8
* CLOSE THE LOAD LIBRARY
         CLOSE ((R11)),MODE=31
         DROP  R11                      * FINISHED WITH DCB
         EJECT
*---------------------------------------------------------------------
* IGX24CHK MODULE LOADED, SO REQUEST PERMISSION TO PROCEED
*---------------------------------------------------------------------
* FIND OUT WHERE THE CURRENT IGXØØØ24 ENTRY POINT IS
         L     R11,FLCCVT-PSA(RØ)        * CVT -> ...
         L     R11,CVTABEND-CVT(R11)     * ... SCVT -> ...
         L     R11,SCVTSVCT-SCVTSECT(R11) *     ... -> SVC TABLE
         L     R1,SVC1Ø9                * OFFSET INTO SVC TABLE ...
         SLL   R1,3                     * ... = (SVC NUMBER) * 8
         AR    R11,R1                   * SVC TABLE ENTRY ...
         USING SVCENTRY,R11             * ... ADDRESSABILITY
         L     R11,SVCEP                * @(ESR TABLE)
         CLC   Ø(4,R11),ESR3            * VERIFY CORRECT ADDRESS
         BNE   BAD@ESRT                 * QUIT IF WRONG
         L     R1,RTCDE24               * OFFSET INTO ...
         SLL   R1,3                     * ... ESR TABLE ...
         LA    R1,8(,R1)                *     ... = (ROUTECDE * 8) + 8
         AR    R11,R1                   * ESR TABLE ENTRY ADDRESS
         L     R3,SVCEP                 * @(ENTRY POINT)
         CLC   5(8,R3),BLNAME           * INTERCEPT ALREADY INSTALLED?
         BNE   NOTINSTL                 * NO, OK TO PROCEED
         SR    RØ,RØ                    * CLEAR RØ (NO CONSOLE ID)
X24IØ1E  WTO   'X24IØ1E The IGXØØØ24 Front End Intercept is already ins+
               talled',ROUTCDE=2,DESC=3
         B     ALRDYDNE                 * QUIT
* ... AND ASK FOR PERMISSION TO INSTALL THE INTERCEPT
NOTINSTL EQU   *
         L     R3,SVCEP                 * GET EP ADDRESS
         MVC   IGXØØØ24,4(R3)           * SAVE MODULE NAME
         MVC   X24IØ1A+24(8),4(R3)      * COPY MODULE NAME
         SLDL  R2,4                     * SHIFT IN A DUMMY SIGN NIBBLE
         STM   R2,R3,TEMP8              * STORE IT AS PSEUDO-PACKED
         UNPK  X24IØ1A+39(8),TEMP8+3(5) * UNPACK TOKEN WORD
         NC    X24IØ1A+39(8),ZONEMASK   * CONVERT ZONES TO ZEROS
         TR    X24IØ1A+39(8),HEXTAB     * CONVERT TO EBCDIC
         SR    RØ,RØ                    * CLEAR RØ (NO CONSOLE ID)
X24IØ1A  WTOR  'X24IØ1A aaaaaaaa is at xxxxxxxx - OK to install front e+
               nd ?',REPLY,3,ECB,ROUTCDE=2
         WAIT  ECB=ECB                  * WAIT FOR RESPONSE
         CLC   REPLY,YES                * PERMISSION GRANTED ?
         BNE   DENIED                   * NO - QUIT
* INSERT THE ADDRESS OF THE 'REAL' IGXØØØ24 INTO THE IGX24CHK MODULE
         MODESET KEY=ZERO               * GET INTO KEY ZERO
         MVC   28(4,R1Ø),SVCEP          * STORE @IGXØØØ24 IN IGX24CHK
         MODESET KEY=NZERO              * REVERT TO KEY 8
         EJECT
*----------------------------------------------------------------------
* PERMISSION GRANTED, SO INSTALL THE IGXØØØ24 FRONT END INTERCEPT
*----------------------------------------------------------------------
* SORT OUT THE ENTRY POINT ADDRESS
DOINSTAL EQU   *
         SR    R2,R2                    * GET @(EP) RELATIVE ...
         ICM   R2,B'Ø111',BLEPADDR      * ... TO START OF MODULE
         AR    R1Ø,R2                   * COMPUTE ABSOLUTE ENTRY POINT
         O     R1Ø,AMODE31              * SET AMODE 31 BIT
* AND INSTALL THE INTERCEPT ADDRESS IN THE ESR TABLE
ESRUPDTE EQU   *
         MODESET KEY=ZERO,MODE=SUP      * KEY ZERO/SUPERVISOR STATE
         ENQ   (SYSZSVC,TABLE,E,L'TABLE,SYSTEM),RET=NONE
         LRA   R2,SVCEP                 * R2 = REAL ADDRESS OF ESRT
         BC    8,DOLURA1                * JUMP IF OK TO PROCEED
         LA    R8,4                     * ELSE ...
         B     INSTDONE                 * ... REJECT UPDATE
DOLURA1  EQU   *
         LURA  R3,R2                    * CHECK REAL ADDRESS ...
         CLC   4(8,R3),IGXØØØ24         * ... POINTS WHERE WE EXPECT
         BE    DOSTURA1                 * JUMP IF OK TO PROCEED
         LA    R8,8                     * ELSE ...
         B     INSTDONE                 * ... REJECT UPDATE
DOSTURA1 EQU   *
         STURA R1Ø,R2                   * UPDATE ESR TABLE ENTRY
         SLR   R8,R8                    * AND SIGNAL UPDATE OK
INSTDONE EQU   *
         DEQ   (SYSZSVC,TABLE,L'TABLE,SYSTEM),RET=NONE
         MODESET KEY=NZERO,MODE=PROB    * KEY 8/PROBLEM STATE
         LTR   R8,R8                    * ESR TABLE UPDATE OK ?
         BNZ   BADUPDTE                 * NOPE
         SR    R9,R9                    * ALL OK, SO SET RC = Ø
* SUCCESS - TELL US ALL ABOUT IT
```

```
         MVC   X24I01I+16(8),BLNAME      * MOVE IN MODULE NAME
         LR    R3,R10                    * COPY NEW ADDRESS
         SLDL  R2,4                      * SHIFT IN A DUMMY SIGN NIBBLE
         STM   R2,R3,TEMP8               * STORE IT AS PSEUDO-PACKED
         UNPK  X24I01I+51(8),TEMP8+3(5)  * UNPACK NEW ADDRESS
         NC    X24I01I+51(8),ZONEMASK    * CONVERT ZONES TO ZEROS
         TR    X24I01I+51(8),HEXTAB      * CONVERT TO EBCDIC
         SR    R0,R0                     * CLEAR R0 (NO CONSOLE ID)
X24I01I  WTO   'X24I01I aaaaaaaa successfully installed at xxxxxxxx',  +
               ROUTCDE=2,DESC=3
         B     RETURN
         DROP  R11                       * FINISHED WITH ESR TABLE
         EJECT
***********************************************************************
* PARM=DELETE : DELETE THE IGX00024 FRONT END INTERCEPT ROUTINE
***********************************************************************
* FIND OUT WHERE THE CURRENT IGX00024 ENTRY POINT IS
DELETE   EQU   *
         L     R11,FLCCVT-PSA(R0)         * CVT -> ...
         L     R11,CVTABEND-CVT(R11)      * ... SCVT -> ...
         L     R11,SCVTSVCT-SCVTSECT(R11) *     ... -> SVC TABLE
         L     R1,SVC109                 * OFFSET INTO SVC TABLE ...
         SLL   R1,3                      * ... = (SVC NUMBER) * 8
         AR    R11,R1                    * SVC TABLE ENTRY ...
         USING SVCENTRY,R11              * ... ADDRESSABILITY
         L     R11,SVCEP                 * @(ESR TABLE)
         CLC   0(4,R11),ESR3             * VERIFY CORRECT ADDRESS
         BNE   BAD@ESRT                  * QUIT IF WRONG
         L     R1,RTCDE24                * OFFSET INTO ...
         SLL   R1,3                      * ... ESR TABLE ...
         LA    R1,8(,R1)                 *     ... = (ROUTECDE * 8) + 8
         AR    R11,R1                    * ESR TABLE ENTRY ADDRESS
* ... AND ASK FOR PERMISSION TO DELETE THE INTERCEPT
         L     R3,SVCEP                  * GET EP ADDRESS
         MVC   X24I02A+24(8),5(R3)       * COPY MODULE NAME
         SLDL  R2,4                      * SHIFT IN A DUMMY SIGN NIBBLE
         STM   R2,R3,TEMP8               * STORE IT AS PSEUDO-PACKED
         UNPK  X24I02A+39(8),TEMP8+3(5)  * UNPACK TOKEN WORD
         NC    X24I02A+39(8),ZONEMASK    * CONVERT ZONES TO ZEROS
         TR    X24I02A+39(8),HEXTAB      * CONVERT TO EBCDIC
         SR    R0,R0                     * CLEAR R0 (NO CONSOLE ID)
X24I02A  WTOR  'X24I02A aaaaaaaa is at xxxxxxxx - OK to deinstall front+
                end ?',REPLY,3,ECB,ROUTCDE=2
         WAIT  ECB=ECB                   * WAIT FOR RESPONSE
         CLC   REPLY,YES                 * PERMISSION GRANTED ?
         BNE   DENIED                    * NO - QUIT
* EXTRACT THE ADDRESS OF THE REAL IGX00024 FROM THE IGX24CHK MODULE
         L     R10,SVCEP                 * R10 = @(IGX24CHK)
         CLC   5(8,R10),BLNAME           * VERIFY MODULE NAME
         BNE   WRONGINT                  * IF NOT IGX24CHK, QUIT
         L     R9,28(,R10)               * R9  = @(IGX00024)
         EJECT
*----------------------------------------------------------------------
* PERMISSION GRANTED, SO DELETE THE IGXØØØ24 FRONT END INTERCEPT
*----------------------------------------------------------------------
* RESTORE THE ADDRESS OF THE REAL IGXØØØ24 IN THE ESR TABLE
DODELETE EQU   *
         MODESET KEY=ZERO,MODE=SUP       * KEY ZERO/SUPERVISOR STATE
         ENQ   (SYSZSVC,TABLE,E,L'TABLE,SYSTEM),RET=NONE
         LRA   R2,SVCEP                  * R2 = REAL ADDRESS OF ESRT
         BC    8,DOLURA2                 * JUMP IF OK TO PROCEED
         LA    R8,4                      * ELSE ...
         B     DLTEDONE                  * ... REJECT UPDATE
DOLURA2  EQU   *
         LURA  R3,R2                     * CHECK REAL ADDRESS ...
         CLC   5(8,R3),BLNAME            * ... POINTS WHERE WE EXPECT
         BE    DOSTURA2                  * JUMP IF OK TO PROCEED
         LA    R8,8                      * ELSE ...
         B     DLTEDONE                  * ... REJECT UPDATE
DOSTURA2 EQU   *
         STURA R9,R2                     * UPDATE ESR TABLE ENTRY
         SLR   R8,R8                     * AND SIGNAL UPDATE OK
DLTEDONE EQU   *
         DEQ   (SYSZSVC,TABLE,L'TABLE,SYSTEM),RET=NONE
         MODESET KEY=NZERO,MODE=PROB     * KEY 8/PROBLEM STATE
         LTR   R8,R8                     * ESR TABLE UPDATE OK ?
         BNZ   BADUPDTE                  * NOPE
         SR    R9,R9                     * ALL OK, SO SET RC = Ø
* SUCCESS - TELL US ALL ABOUT IT
         LR    R3,R1Ø                    * COPY OLD ADDRESS
         MVC   X24IØ2I+16(8),5(R3)       * MOVE IN MODULE NAME
         SLDL  R2,4                      * SHIFT IN A DUMMY SIGN NIBBLE
         STM   R2,R3,TEMP8               * STORE IT AS PSEUDO-PACKED
         UNPK  X24IØ2I+28(8),TEMP8+3(5)  * UNPACK OLD ADDRESS
         NC    X24IØ2I+28(8),ZONEMASK    * CONVERT ZONES TO ZEROS
         TR    X24IØ2I+28(8),HEXTAB      * CONVERT TO EBCDIC
         SR    RØ,RØ                     * CLEAR RØ (NO CONSOLE ID)
X24IØ2I  WTO   'X24IØ2I aaaaaaaa at xxxxxxxx successfully deinstalled',+
               ROUTCDE=2,DESC=3
         DROP  R11                       * FINISHED WITH ESR TABLE
         EJECT
***********************************************************************
* ALL DONE, SO RETURN
***********************************************************************
RETURN   EQU   *
         L     R13,4(R13)                * RESTORE HSA ADDRESS
         LR    RØ,R8                     * SET REASON CODE
         LR    R15,R9                    * SET RETURN CODE
         L     R14,12(R13)               * RESTORE R14
         LM    R1,R12,24(R13)            * RESTORE R1 - R12
         BR    R14                       * AND RETURN
         EJECT
```

```
***********************************************************************
* ERROR CONDITIONS
***********************************************************************
BADPARM  EQU   *
         LA    R9,4                       * RETURN CODE = 4
         B     ABANDON                    * ABANDON SHIP
BADOPEN  EQU   *
         SR    R8,R8                      * REASON CODE = Ø
         LA    R9,8                       * RETURN CODE = 8
         B     ABANDON                    * ABANDON SHIP
BADBLDL  EQU   *
         LA    R9,12                      * RETURN CODE = 12
         B     ABANDON                    * ABANDON SHIP
BADLMOD  EQU   *
         LA    R9,16                      * RETURN CODE = 16
         B     ABANDON                    * ABANDON SHIP
BAD@ESRT EQU   *
         LA    R9,2Ø                      * RETURN CODE = 2Ø
         B     ABANDON                    * ABANDON SHIP
ALRDYDNE EQU   *
         SR    R8,R8                      * REASON CODE = Ø
         LA    R9,24                      * RETURN CODE = 24
         B     ABANDON                    * ABANDON SHIP
DENIED   EQU   *
         SR    R8,R8                      * REASON CODE = Ø
         LA    R9,28                      * RETURN CODE = 28
         B     ABANDON                    * ABANDON SHIP
WRONGINT EQU   *
         SR    R8,R8                      * REASON CODE = Ø
         LA    R9,32                      * RETURN CODE = 32
         B     ABANDON                    * ABANDON SHIP
BADUPDTE EQU   *
         LA    R9,36                      * RETURN CODE = 36
ABANDON  EQU   *
         LR    R1,R9                      * DIVIDE ...
         SRL   R1,1                       * ... RETURN CODE BY 2
         LA    R1,RETCODES(R1)            * ADDRESS OF EBCDIC EQUIVALENT
         MVC   X24IØ2E+4Ø(2),Ø(R1)        * MOVE RETCODE INTO MSG
         LR    R1,R8                      * DIVIDE ...
         SRL   R1,1                       * ... REASON CODE BY 2
         LA    R1,RETCODES(R1)            * ADDRESS OF EBCDIC EQUIVALENT
         MVC   X24IØ2E+51(2),Ø(R1)        * MOVE REASON CODE INTO MSG
         SR    RØ,RØ                      * CLEAR RØ (NO CONSOLE ID)
X24IØ2E  WTO   'X24IØ2E IGX24INT failed with rc=xx, reason=xx',        +
               ROUTCDE=2,DESC=3
         B     RETURN                     * ABANDON SHIP
         EJECT
***********************************************************************
* CONSTANTS, VARIABLES AND DATA AREAS
* NB FOR DETAILS OF LOAD MODULE DIRECTORY ENTRY SEE LINKAGE EDITOR
*    LOGIC MANUAL.
```

```
***********************************************************************
         DS    ØD
SAVEAREA DS    18F
TEMP8    DS    D
SVC1Ø9   DC    F'1Ø9'
RTCDE24  DC    F'24'
AMODE31  DC    X'8ØØØØØØØ'
ECB      DC    F'Ø'
ESR3     DC    CL4'ESR3'
IGXØØØ24 DC    CL8'********'
REPLY    DC    CL3'   '
YES      DC    CL3'YES'
         DS    ØF
SVCLIB   DCB   DDNAME=SVCLIB,DSORG=PO,MACRF=R,RECFM=U
         DS    ØF
BLDLLIST DC    H'1'
         DC    H'76'
BLNAME   DC    CL8'IGX24CHK'
BLTTR    DS    XL3
BLK      DS    XL1
BLZ      DS    XL1
BLC      DS    XL1
BLTTRTX1 DS    XL3                        * TTR OF FIRST TEXT RECORD
BLZ2     DS    XL1                        * ZEROS
BLTTRNL  DS    XL3                        * TTR OF NOTE LIST (IF ANY)
BLNNOTEL DS    XL1                        * NUMBER OF NOTE LIST ENTRIES
BLMATTR1 DS    XL2                        * MODULE ATTRIBUTES (1)
BLA1RENT EQU   X'8Ø'                      *  REENTRANT
BLA1REUS EQU   X'4Ø'                      *  REUSABLE
BLA1EXEC EQU   X'Ø2'                      *  EXECUTABLE
BLA1REFR EQU   X'Ø1'                      *  REFRESHABLE
BLMODLN  DS    XL3                        * VIRTUAL STORAGE REQUIRED
BLLTX1   DS    XL2                        * LENGTH OF FIRST TEXT RECORD
BLEPADDR DS    XL3                        * MODULE ENTRY POINT
BLMATTR2 DS    XL1                        * MODULE ATTRIBUTES (2)
BLARMODE DS    XL1                        * AMODE/RMODE FLAGS
BLRMANY  EQU   X'1Ø'                      *  RMODE = ANY
BLAM31   EQU   X'Ø2'                      *  AMODE = 31
BLRLDCNT DS    XL1                        * RLD COUNT
BLLSCLST DS    XL2                        * LENGTH OF SCATTER LIST
BLLTRTAB DS    XL2                        * LENGTH OF TRANSLATION TABLE
BLLESDTX DS    XL2                        * ESDID FOR FIRST TEXT RECORD
BLLESDEP DS    XL2                        * ESDID FOR ENTRY POINTS
BLMEMEPA DS    XL3                        * EPA OF 'REAL' MEMBER
BLMEMNAM DS    XL8                        * NAME OF 'REAL' MEMBER
         DS    XL22
         DS    ØF
SYSZSVC  DC    CL8'SYSZSVC '
TABLE    DC    CL5'TABLE'
         DS    ØF
ZONEMASK DC    XL8'ØFØFØFØFØFØFØFØF'
```

```
HEXTAB   DC    CL16'Ø123456789ABCDEF'
RETCODES DC    C'ØØØ4Ø8ØC1Ø14181C2Ø24282C3Ø'
         EJECT
*-----------------------------------------------------------------------
* SYSTEM CONTROL BLOCK DSECTS
*-----------------------------------------------------------------------
         PRINT NOGEN
         IHAPSA LIST=NO                 * PSA MAPPING MACRO
         CVT    DSECT=YES               * CVT MAPPING MACRO
         IHASCVT                        * SCVT MAPPING MACRO
         IHASVC                         * SVC TABLE MAPPING MACRO
         DCBD  DSORG=PS,DEVD=DA         * DCB MAPPING MACRO
         END
```

SOURCE CODE FOR THE IGX24CHK INTERCEPT ROUTINE

```
TITLE 'IGX24CHK - HSM Communication SVC Front End Intercept'
***********************************************************************
*   IGX24CHK - HSM Communication SVC Front End Intercept
*   ---------------------------------------------------             *
*   This routine,  installed  as  a  front-end intercept to the HSM   *
*   Communication SVC,  IGXØØØ24 (SVC 1Ø9 routcode 24), is intended   *
*   to prevent the recall of migrated datasets by users who have no   *
*   access and cannot do anything with them anyway.                   *
*   The intercept is installed and deleted by the IGX24INT program,   *
*   which updates the ESR table directly.                             *
*   ENVIRONMENT                                                       *
*     STATE    : SUPERVISOR                                           *
*     KEY      : Ø                                                    *
*     APF      : NO                                                   *
*     AMODE    : 31                                                   *
*     RMODE    : ANY                                                  *
*     LOCATION : ECSA SUBPOOL 228 (LOADED BY IGX24INT)                *
*                                                                     *
*   REGISTERS ON ENTRY                                                *
*     RØ     - HSM REQUEST TYPE, 'ØØØØØØxx'                           *
*     R1     - @(MWE)                                                 *
*     R2     - UNPREDICATABLE                                         *
*     R3     - @(CVT)                                                 *
*     R4     - @(TCB)                                                 *
*     R5     - @(SVRB)                                                *
*     R6     - ENTRY ADDRESS                                          *
*     R7     - @(ASCB)                                                *
*     R8-R12 - UNPREDICTABLE                                          *
*     R13    - AS CALLER LEFT IT                                      *
*     R14    - RETURN ADDRESS                                         *
*     R15    - ESR ROUTE CODE, X'ØØØØØØ18'                            *
*   REGISTERS ON RETURN IF USER NOT AUTHORISED                        *
*     RØ-R14 - RESTORED TO CONTENTS AT ENTRY                          *
*     R15    - RETURN CODE :                                          *
*              Ø : ROUTER RETURN CODE                                 *
***********************************************************************
         EJECT
IGX24CHK CSECT
IGX24CHK AMODE 31
IGX24CHK RMODE ANY
RØ       EQU    Ø                        * HSM FUNCTION CODE
R1       EQU    1                        * @(MWE)
R2       EQU    2                        * MODESET STORAGE KEY
R3       EQU    3                        * @(CVT)
R4       EQU    4                        * @(TCB)
R5       EQU    5                        * @(SVRB)
R6       EQU    6                        * @(ENTRY POINT)/BASE REGISTER
R7       EQU    7                        * @(ASCB)
R8       EQU    8                        * @(DSN)
R9       EQU    9                        * L'DSN
R1Ø      EQU   1Ø                        * RACROUTE RETURN CODE
R11      EQU   11                        * @(ACEE)
R12      EQU   12                        * @(MWE) LOCALLY
R13      EQU   13                        * @(WORKAREA)
R14      EQU   14                        * RETURN ADDRESS
R15      EQU   15                        * RETURN CODE
*
         USING IGX24CHK,R6               * CSECT ADDRESSABILITY
         B     START                     * SKIP THE HEADER TEXT
         DC    AL1(START-*-1)
         DC    CL9'IGX24CHK '
         DC    CL8'&SYSDATE'
         DC    CL6' &SYSTIME'
@IGX24EP DC    A(Ø)                      * @(REAL IGXØØØ24)
*
***********************************************************************
* ADDRESSABILITY AND LINKAGE
***********************************************************************
         DS    ØF                        * ALIGN TO FULL WORD BOUNDARY
START    EQU   *
         USING TCB,R4                    * TCB  ADDRESSABILITY
         USING RBBASIC,R5                * RB   ADDRESSABILITY
         USING ASCB,R7                   * ASCB ADDRESSABILITY
         STM   RØ,R2,RBEXSAVE            * SAVE RØ - R2
         STM   R8,R15,RBEXSAVE+12        * SAVE R8 - R15
         LR    R12,R1                    * MWE ...
         USING ARCMWE,R12                * ... ADDRESSABILITY
         CLI   MWEFUNC,MWERECAL          * FAST PASS THROUGH IF ...
         BNE   PASSTHRU                  * ... NOT A RECALL REQUEST
         EJECT
***********************************************************************
* THE MWE IS A RECALL MWE ...
***********************************************************************
* WHITTLE OUT THE PRIVILEGED CALLERS
```

```
         ICM   R1Ø,B'1111',ASCBJBNI      * CONTINUE CHECKING ...
         BNZ   CHECKUSR                  * ... IF CALLER A BATCH JOB
         ICM   R1Ø,B'1111',ASCBTSB       * BYPASS CHECKS ...
         BZ    PASSTHRU                  * ... IF CALLER A STARTED TASK
CHECKUSR EQU   *
         L     R11,ASCBASXB              * ASCB -> ASXB ...
         ICM   R11,15,ASXBSENV-ASXB(R11) * ... ASXB -> ACEE
         BZ    PASSTHRU                  * FAST PASS THROUGH IF NO RACF
         USING ACEE,R11                  * ACEE ADDRESSABILITY
         TM    ACEEFLG1,ACEEOPER         * BYPASS CHECKS ...
         BO    PASSTHRU                  * ... IF CALLER HAS OPERATIONS
         SLR   R1,R1                     * GET ...
         ICM   R1,B'ØØØ1',ACEEUSRL       * ... L'UID
         BZ    PASSTHRU                  * PASS THROUGH IF ZERO
         BCTR  R1,Ø                      * COMPARE UID ...
         EX    R1,COMPPFX                * ... WITH DATASET PREFIX
         BE    PASSTHRU                  * IF SAME NO NEED TO CHECK
         B     CHEKAUTH                  * IF NOT, CHECK IS NEEDED
COMPPFX  CLC   MWEDSN(Ø),ACEEUSRI        * COMPARE PREFIX AND UID
         EJECT
*----------------------------------------------------------------------
* NON-PRIVILEGED TSU OR JOB TRYING TO RECALL A DATASET. IS IT ALLOWED ?
*----------------------------------------------------------------------
* GET STORAGE FOR RACROUTE AND BUILD DATASET "ENTITYX" BUFFER
CHEKAUTH EQU   *
         MODESET EXTKEY=TCB,WORKREG=2    * ASSUME TCB KEY
*DEBUG   WTO   'IGX24CHK Checking non-privileged caller',ROUTCDE=2
*DEBUG
         STORAGE OBTAIN,LENGTH=WKALEN,SP=23Ø,CALLRKY=YES,LOC=ANY
         LR    R13,R1                    * WORKAREA ...
         USING WORKAREA,R13              * ... ADDRESSABILITY
         LA    R1,L'DSNAME               * GET A 44
         SLA   R1,16                     * L'BUFFER = 44 ...
         ST    R1,DSNBUF                 * ... AND L'(ENTITY NAME) = Ø
         MVC   DSNAME,MWEDSN             * MOVE IN DATASET NAME
* CHECK RACF AUTHORIzATION TO THE DATASET
CHKAUTH1 EQU   *
         MVC   RACCHKW(LRACCHKL),RACCHKL * MOVE RACROUTE INTO WORKAREA
         RACROUTE REQUEST=AUTH,                                        +
               WORKA=RACFWORK,                                         +
               MSGSUPP=NO,                                             +
               LOG=ASIS,                                               +
               RELEASE=2.1,                                            +
               ATTR=READ,                                              +
               CLASS=DATASET,                                          +
               ENTITYX=DSNBUF,                                         +
               VOLSER=X24VOL,                                          +
               MF=(E,RACCHKW)
         LTR   R1Ø,R15                   * TEST RACROUTE RETURN CODE
         BZ    CHKAUTH7                  * IF ZERO OK TO PROCEED
* CALLER IS NOT AUTHORISED, SO BUILD AND ISSUE A NICE MESSAGE
```

```
         LA    R1,WTOBUF                 * @(WTO BUFFER)
         MVC   Ø(LMSG1,R1),MSG1          * MOVE MF=L WTO INTO WORKAREA
         LA    R8,MWEDSN                 * @(DSNAME)
         LA    R9,43                     * L'DSNAME IS 44 CHARS MAXIMUM
CHKAUTH2 EQU   *
         LA    R2,Ø(R9,R8)               * @(LAST CHARACTER)
         CLI   Ø(R2),C' '                * IF NOT BLANK ...
         BNE   CHKAUTH3                  * ... LENGTH NOW CORRECT
         BCTR  R9,Ø                      * OTHERWISE ...
         B     CHKAUTH2                  * ... MOVE BACK ONE
CHKAUTH3 EQU   *
         EX    R9,MOVEDSN2               * INSERT DATASET NAME
         B     CHKAUTH4                  * JUMP OVER EXECUTED MVC
MOVEDSN2 MVC   23(Ø,R1),Ø(R8)            * MOVE DSN INTO WTO
CHKAUTH4 EQU   *
         MVC   68(25,R1),67(R1)          * INSERT NEXT ...
         LA    R9,24(R9,R1)              * ... BIT OF MESSAGE ...
         MVC   Ø(4,R9),MSG1+67           *     ... AT THE PROPER PLACE
         ICM   R2,B'1111',ASCBJBNI       * @(BATCH JOB NAME)
         BNZ   *+8                       * JUMP IF SET
         L     R2,ASCBJBNS               * ELSE GET @(STC/TSU JOB NAME)
         MVC   4(8,R9),Ø(R2)             * INSERT CALLER'S JOBNAME
         LA    R9,11(,R9)                * @(LAST CHAR OF JOBNAME)
CHKAUTH5 EQU   *
         CLI   Ø(R9),C' '                * MOVE ...
         BNE   CHKAUTH6                  * ... BACK ...
         BCTR  R9,Ø                      *     ... UNTIL ...
         B     CHKAUTH5                  *         ... HIT NON-BLANK
CHKAUTH6 EQU   *
         MVC   1(14,R9),MSG1+79          * INSERT LAST BIT OF MESSAGE
         SR    RØ,RØ                     * CLEAR RØ FOR WTO
         WTO   MF=(E,(1))                * LOG REJECTED REQUEST
* RELEASE RACROUTE WORKAREA AND CHECK RACF RETURN CODE AGAIN
CHKAUTH7 EQU   *
         LR    R1,R13                    * R1 = @(WORKAREA)
         IVSK  R2,R1                     * R2 = STORAGE KEY
         STORAGE RELEASE,LENGTH=WKALEN,ADDR=(1),SP=23Ø,KEY=(2)
         DROP  R13                       * FINISHED WITH WORKAREA
         LTR   R1Ø,R1Ø                   * CHECK RACF RC AGAIN
         BNZ   CHKAUTH8                  * DENY ACCESS IF NOT ZERO
         MODESET EXTKEY=ZERO,WORKREG=2   * ELSE REVERT TO KEY Ø ...
         B     PASSTHRU                  * ... AND PASS ON THROUGH
* USER NOT AUTHORIzED, SO COMPLETE MWE AS IF HSM HAD PROCESSED IT
CHKAUTH8 EQU   *
         IVSK  R2,R12                    * ASSUME KEY ...
         MODESET KEYREG=2                * ... OF MWE STORAGE
         LA    R1,LMWE                   * INSERT ...
         STCM  R1,B'Ø111',MWELEN         * ... L'MWE=288
         TIME  DEC                       * INSERT ...
         STM   RØ,R1,MWESTIM             * ... TIME AND DATE
         ICM   R1,B'1111',ASCBTSB        * SET ...
         BZ    CHKAUTH9                 * ... TSOR FLAG ...
         OI    MWEFLG11,MWEFTSOR        *     ... IF CALLER A TSU
CHKAUTH9 EQU   *
         OI    MWEFLG13,MWEFDONE        * MWE HAS BEEN PROCESSED
         MVC   MWEUID,ACEEUSRI          * MOVE RACF UID INTO MWE
         ST    R7,MWEASCB               * MOVE @(ASCB) INTO MWE
         MVC   MWERC,AUTHFAIL           * SET MWERC   = 39
         ST    R1Ø,MWEREAS              * SET MWEREAS = RACF REASON
         MVC   MWEID,MWESTAR            * INSERT MWEID = 'MWE*'
         MVC   MWEGROUP,ACEEGRPN        * MOVE RACF GROUP INTO MWE
         L     R9,TCBTCT                * TCB -> TCT ...
         L     R9,TCTJMR-SMFTCT(,R9)    * ... TCT -) JMR
         USING JMR,R9                   * JMR ADDRESSABILITY
         MVC   MWEJBN(16),JMRJOB        * MOVE JMR INFO INTO MWE
         DROP  R9                       * FINISHED WITH JMR
         LA    R1,1                     * INSERT ...
         STH   R1,MWEMCNT               * ... #(MWES) = 1
         MVC   MWEVSN,BLANKS            * INSERT MWEVSN = '      '
         MODESET EXTKEY=ZERO,WORKREG=2  * REVERT TO KEY Ø
         B     RETURN                   * AND REJECT RECALL REQUEST
         DROP  R11,R12                  * FINISHED WITH ACEE,MWE
         EJECT
***********************************************************************
* PASS THROUGH TO THE REAL IGXØØØ24 OR RETURN TO CALLER WITH RC=Ø
***********************************************************************
PASSTHRU EQU   *
*DEBUG   WTO   'IGX24CHK Passing request to IGXØØØ24',ROUTCDE=2
*DEBUG
         LM    RØ,R2,RBEXSAVE           * RESTORE RØ - R2
         LM    R8,R15,RBEXSAVE+12       * RESTORE R8 - R15
         L     R6,@IGX24EP              * PASS THROUGH ...
         BR    R6                       * ... TO THE REAL IGXØØØ24
RETURN   EQU   *
         SLR   RØ,RØ                    * SET RØ = Ø
         LM    R1,R2,RBEXSAVE+4         * RESTORE R1 - R2
         LM    R8,R14,RBEXSAVE+12       * RESTORE R8 - R14
         SLR   R15,R15                  * RETURN TO CALLER ...
         BR    R14                      * ... WITH RC = Ø
         DROP  R4,R5,R7                 * FINISHED WITH TCB,RB,ASCB
         EJECT
***********************************************************************
* CONSTANTS AND DATA AREAS
***********************************************************************
         DS    ØF
LMWE     EQU   288                      * REJECTED MWE LENGTH
MWESTAR  DC    CL4'MWE*'                * MWE ID
AUTHFAIL DC    F'39'                    * 'AUTH FAILED' MWE RETCODE
         DS    ØF
DATASET  DC    XL1'Ø7'                  * LENGTH OF DATASET CLASS NAME
         DC    CL7'DATASET'             * DATASET CLASS NAME
X24VOL   DC    CL6'X24VOL'
```

```
BLANKS   DC    CL6'      '
         DS    ØF
RACCHKL  RACROUTE REQUEST=AUTH,RELEASE=2.1,MF=L
LRACCHKL EQU     *-RACCHKL
         DS    ØF
MSG1     WTO   'IGX24CHK Recall of                                      +
                        by          not permitted',ROUTCDE=9,MF=L
LMSG1    EQU   *-MSG1
         EJECT
*----------------------------------------------------------------------
* REENTRANT RACROUTE WORKAREA DSECT
*----------------------------------------------------------------------
WORKAREA DSECT
SAVEAREA DS    18F                     * SAVEAREA FOR RACROUTE
         DS    ØF
RACCHKW  DS    CL(LRACCHKL)            * MF=L RACROUTE
         DS    ØF
DSNBUF   DS    H                       * BUFFER LENGTH (44)
         DS    H                       * DSNAME LENGTH (Ø)
DSNAME   DS    CL44                    * DSNAME
         DS    ØF
RACFWORK DS    CL512                   * RACROUTE WORKAREA
WTOBUF   EQU   RACFWORK,144            * WTO BUFFER
WKALEN   EQU   *-WORKAREA
         EJECT
*----------------------------------------------------------------------
* SYSTEM CONTROL BLOCK DSECTS
*----------------------------------------------------------------------
         PRINT  NOGEN
         IHAASCB                       * ASCB MAPPING MACRO
         IHAASXB                       * ASXB MAPPING MACRO
         IHAACEE                       * ACEE MAPPING MACRO
         IKJTCB                        * TCB MAPPING MACRO
         IHARB                         * RB MAPPING MACRO
         IEFTCT                        * TCT MAPPING MACRO
         IEFJMR                        * JMR MAPPING MACRO
ARCMWE   DSECT                         * MWE MAPPING MACRO
         DS    XL9
MWELEN   DS    AL3                     * MWE LENGTH
         DS    A
MWESTIM  DS    F                       * TIME WHEN MWE WAS QUEUED
MWESDAT  DS    F                       * DATE WHEN MWE WAS QUEUED
MWEFUNC  DS    XL1                     * MWE FUNCTION CODE
MWERECAL EQU   3                       * CODE 3 = RECALL A DATASET
MWEFLG11 DS    BL1                     * FLAG BYTE 1
MWEFTSOR EQU   X'1Ø'                   *  INTERACTIVE TSO REQUEST
MWEFLG12 DS    BL1                     * FLAG BYTE 2
MWEFLG13 DS    BL1                     * FLAG BYTE 3
MWEFDONE EQU   X'1Ø'                   *  MWE HAS BEEN PROCESSED
MWEUID   DS    CL8                     * TSO UID IF REQUEST FROM TSO
         DS    F
```

```
MWEASCB  DS    F                           * ADDRESS OF REQUESTORS ASCB
         DS    F
MWERC    DS    F                           * HSM RETURN CODE
MWEREAS  DS    F                           * HSM REASON CODE
MWEID    DS    CL4                         * MWE INDENTIFIER, 'MWE*'
         DS    F
MWEGROUP DS    CL8                         * REQUESTORS RACF GROUP
         DS    XL4
MWEJBN   DS    CL8                         * REQUESTING JOB NAME
MWERST   DS    F                           * READER JOB START TIME
MWERSD   DS    F                           * READER JOB START DATE
         DS    XL48
MWEBODY  EQU   *
MWEMCNT  DS    H                           * NUMBER OF MWES IN REQUEST
         DS    XL6
MWEDSN   DS    CL44                        * DATASET NAME
         DS    XL2Ø
MWEVSN   DS    CL6                         * VOLUME SERIAL NUMBER
         DS    XL7Ø
         END
```

P R S Wright
Associate Consultant
Tessella Support Services plc (UK)

Interpreting GTF CCW trace entries

We recently encountered a problem with elongated DASD response times. A preliminary investigation revealed that the responsible component of the I/O operation was disconnect time. I ran a GTF CCW trace on the relevant packs to get a handle on exactly what the I/Os were doing to cause this effect, then imported the trace data into IPCS for analysis.

After spending a great deal of time thumbing through reference cards and various manuals to interpret the output from IPCS, I decided that a far better way to do this would be to write a simple REXX EXEC to format the trace in a more readable way. The result is a program called CCWDISPL which reads a GTF trace file and presents a formatted report of the data therein in a concise and clear format.

CCWDISPL accepts two input parameters, a job name and a four character device number. The job name can be 'ALL' to process all jobs that accessed the device in question, but only one device can be specified. Note that this does not require that only one device be traced by GTF thoughout.

Running the EXEC is simplicity itself, indeed I find it most useful and time saving to append a run of CCWDISPL directly to a GTF CCW trace procedure.

SAMPLE JCL

```
//REXXJCL  EXEC PGM=IRXJCL,
//  PARM='CCWDISPL ALL Ø2Ø1'
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*
//GTFDAT   DD DSN=SYS1.TRACE,DISP=SHR
//SYSEXEC  DD DSN=SYS1.REXX,DISP=SHR
```

The GTF trace parameters are as follows:

```
TRACE=IOP,SSCHP,CCWP
CCW=(S,DATA=2Ø,CCWN=32767)
IO=SSCH=(xxx,yyy,zzz)
```

where xxx,yyy,zzz are the devices to be traced.

The output from CCWDISPL displays the components of each I/O in some detail. The first line describes the start subchannel (SSCH) operation (or RSCH if applicable) including the timestamp and cylinder and head addresses. For I/O events subsequent to the first complete one traced, the seek distance is calculated.

The second line is the beginning of the CCW chain and has only a timestamp. The subsequent lines list the detail of the CCWs, including a command code description, byte count of the data involved in the operation, interpretation of the flag bytes, the CCW itself, and any data that appears in the GTF record in hex and display formats.

The last line of each I/O is the I/O interrupt trace record, signifying the completion of the I/O operation. It includes a timestamp and from this is calculated the elapsed time of the I/O operation. This is the service time for the I/O. Additionally the I/O record includes the connect time monitored, so this is also displayed.

CCWDISPL is useful both as a tool for analysing DASD I/O problems and for gaining an understanding of how various different types of access method handle their I/O. There follows a number of examples of various I/O operations. Please note that, for obvious reasons, I have removed the data portion of the display from the output.

Below is an example of the output of CCWDISPL for an update to a VSAM data component, a fairly straightforward I/O operation.

```
Jobname  Type  <--- Time ---> CCCC  Seek  HH
<-- CCW type --> Bytes <-------- CC Flags ---------> <-- CCW/IDAW -->
                                 Serv    Conn

VSAMAPPL SSCH 11:43:Ø8.94Ø549 Ø39D  473 ØØØB
VSAMAPPL CCWs 11:43:Ø8.94Ø691
Define extent       16 Eckd, Cache, Dfw             634ØØØ1ØØ3CØ87D8
Locate record       16 Write data Ø39D,ØB,Ø2,46     474ØØØ1ØØ3CØ87E8
Transfer in chan     Ø                              Ø8ØØØØØØ3CØ8868
Write upd data   16384                              854440ØØ3CØ8878
IDAW                                                Ø4152ØØØØ8ØØØØØØ
IDAW                                                Ø41528ØØØ8ØØØØØØ
IDAW                                                1943ØØØØØ8ØØØØØØ
IDAW                                                19430800Ø8ØØØØØØ
IDAW                                                17595ØØØØ8ØØØØØØ
IDAW                                                175958ØØØ8ØØØØØØ
IDAW                                                Ø84A7ØØØØ8ØØØØØØ
IDAW                                                Ø84A78ØØØ8ØØØØØØ
No operation         1                              Ø32ØØØØ1ØØØØØØØØ
VSAMAPPL IO   11:43:Ø9.ØØ6674    66.1     4.9
```

An example of the output of CCWDISPL used in the analysis of the problem described at the outset of this article can be seen below. It is a DB2 database component of a SAP R/2 system. After seeing how DB2 was chaining 4K CCWs together, it became apparent why the disconnect time was high.

```
Jobname  Type  <--- Time ---> CCCC  Seek  HH
<-- CCW type --> Bytes <-------- CC Flags ---------> <-- CCW/IDAW -->
                                 Serv    Conn

DB2SDBM1 SSCH 11:43:35.156287 Ø113    Ø ØØØ7
DB2SDBM1 CCWs 11:43:35.156538
Define extent       16 Eckd, Cache, Dfw             634ØØØ1ØØ77Ø98D8
Locate record       16 Read data Ø113,Ø7,Ø5,4D      474ØØØ1ØØ77Ø98E8
MT Read data      4Ø96                              86441ØØØØ77Ø98F8
MT Read data      4Ø96                              86441ØØØØ77Ø99ØØ
MT Read data      4096                               8644100007709908
MT Read data      4096                               8644100007709910
MT Read data      4096                               8644100007709918
MT Read data      4096                               8644100007709920
MT Read data      4096                               8644100007709928
MT Read data      4096                               8644100007709930
MT Read data      4096                               8644100007709938
MT Read data      4096                               8644100007709940
Locate record       16 Read data 0113,08,08,83       4740001007709948
MT Read data      4096                               8644100007709958
MT Read data      4096                               8644100007709960
MT Read data      4096                               8644100007709968
MT Read data      4096                               8644100007709970
MT Read data      4096                               8644100007709978
MT Read data      4096                               8644100007709980
MT Read data      4096                               8644100007709988
MT Read data      4096                               8644100007709990
MT Read data      4096                               8644100007709998
MT Read data      4096                               86441000077099A0
MT Read data      4096                               86441000077099A8
MT Read data      4096                               86441000077099B0
MT Read data      4096                               86441000077099B8
MT Read data      4096                               86441000077099C0
MT Read data      4096                               86441000077099C8
MT Read data      4096                               86441000077099D0
MT Read data      4096                               86041000077099D8
DB2SDBM1 IO   11:43:35.213784    57.5    15.4
```

An update to the JES2 checkpoint dataset is shown below, it is another complicated I/O operation involving a great number of components.

```
Jobname  Type  <--- Time ---> CCCC  Seek  HH
<-- CCW type --> Bytes <-------- CC Flags ---------> <-- CCW/IDAW -->
                                 Serv    Conn

JES2     SSCH 11:43:11.623977 0001  924 0000
JES2     CCWs 11:43:11.624335
Define extent       16 Eckd, Cache, Dfw              63FF973840000010
Transfer in chan     0                               088A808800000000
Locate record       16 Write data 0001,00,03,0E      476A56C040000010
Transfer in chan 26304                               088A80B0000F66C0
Write upd data    6641                               856A53C0440019F1
IDAW                                                 1507900008000000
IDAW                                                 1507980008000000
IDAW                                                 0677F00008000000
IDAW                                                 0677F80001F10000
Transfer in chan   264                               088A810800100108
Locate record       16 Write data 0001,01,01,06      476A570040000010
Transfer in chan 26368                               088A8130000F6700
Write upd data    4096                               858A810044001000
IDAW                                                 17D2000008000000
IDAW                                                  17D2Ø8ØØØ8ØØØØØØ
Transfer in chan   392                                Ø88A8188ØØ1ØØ188
Locate record       16 Write data ØØØ1,Ø1,Ø3,29       476A574Ø4ØØØØØ1Ø
Transfer in chan 26432                                Ø88A81BØØØØF674Ø
Write upd data    4Ø96                                858A818Ø44ØØ1ØØØ
IDAW                                                  ØB566ØØØØ8ØØØØØØ
IDAW                                                  ØB5668ØØØ8ØØØØØØ
Transfer in chan   584                                Ø88A8248ØØ1ØØ248
Locate record       16 Write data ØØØ1,Ø1,Ø6,5F       476A57AØ4ØØØØØ1Ø
Transfer in chan 26528                                Ø88A827ØØØØF67AØ
Write upd data    4Ø96                                858A824Ø44ØØ1ØØØ
IDAW                                                  ØC936ØØØØ8ØØØØØØ
IDAW                                                  ØC9368ØØØ8ØØØØØØ
Transfer in chan  3Ø8Ø                                Ø88A8CØ8ØØ1ØØCØ8
Locate record       16 Write data ØØØ1,Ø4,Ø9,95       476A5C8Ø4ØØØØØ1Ø
Transfer in chan 27776                                Ø88A8C3ØØØØF6C8Ø
Write upd data    4Ø96                                858A8CØØ44ØØ1ØØØ
IDAW                                                  ØA8CFØØØØ8ØØØØØØ
IDAW                                                  ØA8CF8ØØØ8ØØØØØØ
Transfer in chan     8                                Ø88A8ØØ8ØØ1ØØØØ8
Locate record       16 Write data ØØØ1,ØØ,Ø1,Ø6       476A568Ø4ØØØØØ1Ø
Transfer in chan 2624Ø                                Ø88A8Ø3ØØØØF668Ø
Write upd data     256                                856A51784ØØØØ1ØØ
Transfer in chan   112                                Ø88A8Ø7ØØØ1ØØØ7Ø
Write upd data     256                                856A52786ØØØØ1ØØ
Transfer in chan 25472                                Ø86A538ØØØØF638Ø
Read count           8                                126A53784ØØØØØØ8
No operation         1                                Ø3ØØØØØØ2ØØØØØØ1
JES2     IO   11:43:11.717983    94.Ø    1Ø.8
```

Finally below is a local page dataset access that demonstrates the way in which paging operations need not terminate like other I/O operations, but can make use of resume subchannel (RSCH).

```
Jobname  Type  <--- Time ---> CCCC  Seek  HH
<-- CCW type --> Bytes <-------- CC Flags ---------> <-- CCW/IDAW -->
                                  Serv    Conn

*MASTER* SSCH 13:44:11.446599 ØØ2C   44 ØØØ5
*MASTER* CCWs 13:44:11.447422
Define extent       16 Eckd, Cache, Dfw               634ØØØ1ØØC29772Ø
Locate record       16 Read data ØØ2C,Ø5,Ø8,83        474ØØØ1ØØC29773Ø
Read data         4Ø96                                Ø64Ø1ØØØØØDDØØØØ
No operation         1                                Ø322ØØØ1ØØØØØØØØ
*MASTER* RSCH 13:44:11.4857Ø2
Define extent       16 Eckd, Cache, Dfw               634ØØØ1ØØC2AE82Ø
Locate record       16 Read data ØØ2C,Ø5,Ø9,95        474ØØØ1ØØC2AE83Ø
Read data         4Ø96                                864Ø1ØØØØØA14ØØØ
Transfer in chan     Ø                                Ø8ØØØØØØØC29Ø7DØ
Read data         4Ø96                                864Ø1ØØØØØ684ØØØ
```

```
 No operation         1                               Ø322ØØØ1ØØØØØØØØ
 *MASTER* RSCH 13:44:11.537448
 Define extent       16 Eckd, Cache, Dfw              634ØØØ1ØØC29492Ø
 Locate record       16 Read data ØØ59,Ø6,Ø6,5F       474ØØØ1ØØC29493Ø
 Read data         4Ø96                               864Ø1ØØØØ5CCØØØ
 Transfer in chan     Ø                               Ø8ØØØØØØØC2633DØ
 Read data         4Ø96                               864Ø1ØØØØØA9ØØØ
 Transfer in chan     Ø                               Ø8ØØØØØØØC2AE85Ø
 Read data         4Ø96                               864Ø1ØØØØ59AØØØ
 Transfer in chan     Ø                               Ø8ØØØØØØØC29775Ø
 Read data         4Ø96                               864Ø1ØØØØØDBCØØØ
 No operation         1                               Ø322ØØØ1ØØØØØØØØ
```

It is interesting to note that CCWDISPL can also be run on the output from a GTF trace specifying just I/O and SSCH events. The program will run and produce reports as before, but without the CCW details. This is useful for getting a quick look at a device in terms of I/O service time and seek distance without having the in-depth CCW information.

CCWDISPL SOURCE

```
/*----------------------------- REXX ------------------------------*/
/* Function   : GTF CCW trace analysis                              */
/*-----------------------------------------------------------------*/
numeric digits 21
arg job dev
trecs = Ø; srecs = Ø; crecs = Ø; irecs = Ø; xrecs = Ø
inccws = 'n'; first = 'y'
call init_ccw
sact = Ø; iact = Ø
say ' '
if job = '' then
  say 'No jobname specified, processing all jobs'
  else
  if job = 'ALL' then
    say 'Processing all jobs'
    else
    say 'Processing Job' job
if dev = '' then
  do
  say 'No device specified, exiting'
  exit
  end
  else
  say 'Processing Device' dev
say ' '
say 'Jobname  Type  <--- Time ---> CCCC  Seek  HH'
say '<-- CCW type --> Bytes <-------- CC Flags --------->',
    '<-- CCW/IDAW -->',
    '<-- Data -------------------------->'
say '                                Serv    Conn'
say ' '
done = 'n'
do while done = 'n'
  "execio 1 diskr gtfdat"
  if rc = 0 then
    do
    parse pull gtfrec
    trecs = trecs + 1
    call proc_rec
    end
  else
    done = 'y'
end
say ' '
say 'SSCH  records accepted  =' format(srecs,9,0)
say 'CCW   records accepted  =' format(crecs,9,0)
say 'IO    records accepted  =' format(irecs,9,0)
say 'xSCH  records accepted  =' format(xrecs,9,0)
say ' '
arecs = srecs + irecs + crecs + xrecs
say 'Total records accepted  =' format(arecs,9,0)
rrecs = trecs - (srecs + irecs + crecs + xrecs)
say 'Total records rejected  =' format(rrecs,9,0)
say 'Total records processed =' format(trecs,9,0)
say ' '
exit 0
/*-------------------------------------------------------------------*/
/* Initialize CCW variables                                          */
/*-------------------------------------------------------------------*/
init_ccw:
ccwexp.   = 'ZZ'
ccwexp.00 = 'Test I/O        '
ccwexp.02 = 'Read IPL        '
ccwexp.03 = 'No operation    '
ccwexp.04 = 'Sense           '
ccwexp.05 = 'Write data      '
ccwexp.06 = 'Read data       '
ccwexp.07 = 'Seek            '
ccwexp.08 = 'Transfer in chan'
ccwexp.09 = 'Wrt spec home ad'
ccwexp.0A = 'Read spc home ad'
ccwexp.0B = 'Seek cylinder   '
ccwexp.0D = 'Write key & data'
ccwexp.0E = 'Read key & data '
ccwexp.0F = 'Space count     '
ccwexp.11 = 'Erase           '
ccwexp.12 = 'Read count      '
ccwexp.13 = 'Recalibrate     '
ccwexp.14 = 'Uncond reserve  '
ccwexp.15 = 'Write rec zero  '
ccwexp.16 = 'Read record zero'
ccwexp.17 = 'Restore         '
ccwexp.19 = 'Write home addr '
ccwexp.1A = 'Read home addr  '
ccwexp.1B = 'Seek head       '
ccwexp.1D = 'Write ck&d      '
ccwexp.1E = 'Read ck&d       '
ccwexp.1F = 'Set file mask   '
ccwexp.22 = 'Read sector     '
ccwexp.23 = 'Set sector      '
ccwexp.27 = 'Prfm subsys func'
ccwexp.29 = 'Search key equal'
ccwexp.31 = 'Search id equal '
ccwexp.34 = 'Sense path gr id'
ccwexp.39 = 'Search ha equal '
ccwexp.3E = 'Read subsys data'
ccwexp.44 = 'Reset allegiance'
ccwexp.47 = 'Locate record   '
ccwexp.49 = 'Search key high '
ccwexp.4E = 'Read message id '
ccwexp.51 = 'Search id high  '
ccwexp.54 = 'Sens subsys stat'
ccwexp.5B = 'Suspend mpath rc'
ccwexp.5E = 'Read multi ck&d '
ccwexp.63 = 'Define extent   '
ccwexp.64 = 'Read dev chars  '
ccwexp.69 = 'Search key =/hi '
ccwexp.71 = 'Search id =/hi  '
ccwexp.73 = 'Diagnostic write'
ccwexp.85 = 'Write upd data  '
ccwexp.86 = 'MT Read data    '
ccwexp.87 = 'Set subsys mode '
ccwexp.8D = 'Write upd k&data'
ccwexp.8E = 'Read key & data '
ccwexp.92 = 'MT Read count   '
ccwexp.94 = 'Device release  '
ccwexp.96 = 'MT Read rec zero'
ccwexp.9A = 'MT Read home adr'
ccwexp.9D = 'Write ck&d nxttr'
ccwexp.9E = 'MT Read ck&d    '
ccwexp.A4 = 'Read & reset bl '
ccwexp.A9 = 'MT Search key eq'
ccwexp.AF = 'Set path grp id '
ccwexp.B1 = 'MT Search id eq '
ccwexp.B4 = 'Device reserve  '
ccwexp.B9 = 'MT Search ha eq '
ccwexp.BE = 'MT read k & d   '
ccwexp.C4 = 'Diagnostic s/r  '
ccwexp.C9 = 'MT Search key hi'
ccwexp.D1 = 'MT Search id hi '
ccwexp.DE = 'Read track      '
ccwexp.E4 = 'Sense id        '
ccwexp.E9 = 'MT Search k =/hi'
ccwexp.F1 = 'MT Srch id =/hi '
ccwexp.F3 = 'Diagnostic cntl '
ccwexp.FA = 'Read config data'
return
/*--------------------------------------------------------------------*/
/* Process TOD                                                        */
/*--------------------------------------------------------------------*/
proc_tod:
sec = c2d(tod) / (4Ø96 * 1ØØØ * 1ØØØ)
sec = sec - 3Ø294432ØØ
act = sec
day = sec % (24 * 6Ø * 6Ø)
sec = sec - (24 * 6Ø * 6Ø * day)
day = day + 1
hr  = sec % (6Ø * 6Ø)
sec = sec - (6Ø * 6Ø * hr)
min = sec % 6Ø
sec = sec - (6Ø * min)
hr  = format(hr,2,Ø)
min = format(min,2,Ø)
sec = format(sec,2,6)
ttod = hr || ':' || min || ':' || sec
ttod = translate(ttod,'Ø',' ')
return
/*--------------------------------------------------------------------*/
/* Process a record                                                   */
/*--------------------------------------------------------------------*/
proc_rec:
jnm = substr(gtfrec,19,8)
if job = '' then
  nop
  else
  if job = 'ALL' then
    nop
    else
    if jnm ¬= job then
      return
dvn = c2x(substr(gtfrec,27,2))
if dev ¬= dvn then
  return
fid = substr(gtfrec,2,1)
if fid = 'ØØ'x then
  call proc_ios
  else
  if fid = 'Ø7'x then
    call proc_ccw
    else
    return
```

```
return
/*-----------------------------------------------------------------*/
/* Process IOS                                                     */
/*-----------------------------------------------------------------*/
proc_ios:
eid = substr(gtfrec,11,2)
select
  when eid = '51Ø2'x then
    call proc_ios_csch
  when eid = '51Ø3'x then
    call proc_ios_hsch
  when eid = '51Ø4'x then
    call proc_ios_msch
  when eid = '51Ø5'x then
    call proc_ios_ssch
  when eid = '51Ø6'x then
    call proc_ios_rsch
  when eid = '52ØØ'x then
    call proc_ios_io
  otherwise
    return
end
return
/*-----------------------------------------------------------------*/
/* Process CSCH                                                    */
/*-----------------------------------------------------------------*/
proc_ios_csch:
tod = substr(gtfrec,3,8)
call proc_tod
xrecs = xrecs + 1
say jnm 'CSCH' ttod
return
/*-----------------------------------------------------------------*/
/* Process HSCH                                                    */
/*-----------------------------------------------------------------*/
proc_ios_hsch:
tod = substr(gtfrec,3,8)
call proc_tod
xrecs = xrecs + 1
say jnm 'HSCH' ttod
return
/*-----------------------------------------------------------------*/
/* Process MSCH                                                    */
/*-----------------------------------------------------------------*/
proc_ios_msch:
tod = substr(gtfrec,3,8)
call proc_tod
xrecs = xrecs + 1
say jnm 'MSCH' ttod
return
/*-----------------------------------------------------------------*/
```

```
/* Process SSCH                                                    */
/*-----------------------------------------------------------------*/
proc_ios_ssch:
if first = 'y' then
  okc = c2d(substr(gtfrec,57,2))
  else
  okc = c2d(skc)
first = 'n'
tod = substr(gtfrec,3,8)
call proc_tod
stod = ttod
sact = act
skc = substr(gtfrec,57,2)
skd = format(abs(c2d(skc) - okc),4,0)
skh = substr(gtfrec,59,2)
srecs = srecs + 1
say jnm 'SSCH' stod c2x(skc) skd c2x(skh)
inccws = 'n'
return
/*-----------------------------------------------------------------*/
/* Process RSCH                                                    */
/*-----------------------------------------------------------------*/
proc_ios_rsch:
tod = substr(gtfrec,3,8)
call proc_tod
xrecs = xrecs + 1
say jnm 'RSCH' ttod
return
/*-----------------------------------------------------------------*/
/* Process IO interrupt                                            */
/*-----------------------------------------------------------------*/
proc_ios_io:
tod = substr(gtfrec,3,8)
call proc_tod
itod = ttod
iact = act
if sact = 0 then
  serv = '*******'
  else
  serv = format((iact - sact) * 1000,5,1)
dct = c2d(substr(gtfrec,51,2))
dct = dct * 128 / 1000
irecs = irecs + 1
say jnm 'IO  ' itod serv format(dct,5,1)
say ' '
inccws = 'n'
return
/*-----------------------------------------------------------------*/
/* Process CCW entry                                               */
/*-----------------------------------------------------------------*/
proc_ccw_ent:
```

```
ccwt = substr(gtfrec,cent+9,1)
i = c2x(ccwt)
ccwtyp = ccwexp.i
if ccwtyp = 'ZZ' then
ccwtyp = c2x(ccwt) || ' undefined ' || c2x(substr(gtfrec,cent+9,8))
cf1  = substr(gtfrec,cent,1)
cent = cent + 1
cf2  = substr(gtfrec,cent,1)
cent = cent + 1
ltr  = substr(gtfrec,cent,2)
cent = cent + 2
lti  = substr(gtfrec,cent,1)
cent = cent + 1
idaw = substr(gtfrec,cent,4)
cent = cent + 4
eccw = substr(gtfrec,cent,8)
if fmt1 = 'ff'x then
  bytc = format(c2d(substr(eccw,3,2)),5,Ø)
  else
  bytc = format(c2d(substr(eccw,7,2)),5,Ø)
cent = cent + 8
cctext = ''
if lti ¬= 'ØØ'x then
  do
  data = substr(gtfrec,cent,c2d(lti))
  cent = cent + c2d(lti)
  if i = '63' then
    do
    ccattr = substr(data,2,1)
    if bitor(ccattr,'3f'x) = 'ff'x then
      cctext = 'Eckd'
    if bitor(ccattr,'e3'x) = 'e4'x then
      cctext = cctext || ', Bpc'
      else
      if bitor(ccattr,'e3'x) = 'e5'x then
        cctext = cctext || ', Icl'
        else
        if bitor(ccattr,'e3'x) = 'e6'x then
          cctext = cctext || ', Seq'
          else
          cctext = cctext || ', Cache'
    if bitor(ccattr,'f2'x) = 'ff'x then
      cctext = cctext || ', Cfw'
    if bitor(ccattr,'f1'x) = 'ff'x then
      nop
      else
      cctext = cctext || ', Dfw'
    cctovr = '                             '
    cctext = overlay(cctext,cctovr,1,29)
    end
  if i = '47' then
```

```
    do
    ccattr = bitand(substr(data,1,1),'3f'x)
    select
      when ccattr = 'ØØ'x then
        cctext = 'Orient'
      when ccattr = 'Ø1'x then
        cctext = 'Write data'
      when ccattr = 'Ø3'x then
        cctext = 'Format write'
      when ccattr = 'Ø6'x then
        cctext = 'Read data'
      when ccattr = 'Øb'x then
        cctext = 'Write track'
      when ccattr = 'Øc'x then
        cctext = 'Read tracks'
      when ccattr = '16'x then
        cctext = 'Read'
      otherwise
        nop
    end
    ccscc  = c2x(substr(data,9,2))
    ccshh  = c2x(substr(data,12,1))
    ccsr   = c2x(substr(data,13,1))
    ccsctr = c2x(substr(data,14,1))
    cchhrs = ccscc || ',' || ccshh || ',' || ccsr || ',' || ccsctr
    cctext = cctext cchhrs
    cctovr = '                             '
    cctext = overlay(cctext,cctovr,1,29)
    end
  end
  else
  data = ''
if cctext = '' then
  cctext = '                             '
if idaw = 'IDAW' then
  say 'IDAW                                                ',
    c2x(eccw) c2x(data) data
  else
  say ccwtyp bytc cctext c2x(eccw) c2x(data) data
return
/*-----------------------------------------------------------------*/
/* Process CCW                                                     */
/*-----------------------------------------------------------------*/
proc_ccw:
lrec = length(gtfrec)
tod = substr(gtfrec,3,8)
call proc_tod
ctod = ttod
fmt1 = bitor(substr(gtfrec,31,1),'DF'x)
if inccws = 'n' then
  say jnm 'CCWs' ctod
```

```
inccws = 'y'
cent = 35
call proc_ccw_ent
do while cent <= lrec
  call proc_ccw_ent
end
crecs = crecs + 1
return
```

Patrick Mullen
Systems programmer

Validating a path

In recent years our site has for various reasons been steadily increasing its range of LPARS. As a result, we have also been making extended use of EMIF for attaching our devices. The consequent increase in complexity of the HCD definition and the physical cabling has increased the chances for error to creep in. I therefore wanted some way of easily checking if a device was satisfactorily connected. Initially I made do by using the D M=DEV() command, but while reading the macros manual I came across a new MVS Version 5 macro called IOSPATHV. This macro effectively allows a program to be written that can carry out a check of a path to see if it is attached and if the device and path are I/O capable. Also, should there be a problem detected by the macro, it returns diagnostic information to enable the user to resolve what is wrong. As a result, I decided to create the following ISPF dialog and REXX function to allow me to exploit the macro. Furthermore, because the REXX routine is re-entrant, it could also be exploited by any REXX-supporting MVS system (eg console automation products). In order to use the code, though, it is necessary to have two SVCs available: one to permit dynamic APF authorization and one to de-authorize. I would assume most sites have access to such SVC's, but in case they do not I have included the code for these SVCs as well.

Before supplying the code, the following is a description of how the dialog works. To begin with, issue TSO PATHVAL (the name of the REXX). This will cause a pop up panel to be displayed as follows:

```
.-------------------- Device path validation ----------------------.
| Specify Device Number ===>
|
| Specify test channel  ===>
'__________________________________________________________________'
```

Enter the device address and the channel that you want to test. If the channel is OK, then the following panel will appear (in this case a device address of 200 and a channel of 11 was entered):

```
.-------------------- Device path validation ----------------------.
| Results of path validation for device 200   channel 11           |
|                                                                  |
| PATH PHYSICALLY AVAILABLE                                        |
'__________________________________________________________________'
```

If there is a problem with the channel, a variety of diagnostic details may appear of which the following is an example:

```
.-------------------- Device path validation ----------------------.
| Results of path validation for device 200   channel 11           |
|                                                                  |
| PATH NOT PHYSICALLY AVAILABLE                                    |
| IOS552I PATH NOT PHYSICALLY AVAILABLE                            |
'__________________________________________________________________'
```

ASSEMBLER CODE

```
        MACRO
        REXREGS
        LCLA &CNT
&CNT    SETA 0
.LOOP   ANOP
R&CNT   EQU &CNT
&CNT    SETA &CNT+1
        AIF (&CNT LT 16).LOOP
        MEND
        MACRO
        SHOW &LABEL,&ASNAME
***********************************************************************
* MACRO FORMAT:
*       SHOW &LABEL,&ASNAME
* WHERE:
*       &LABEL IS THE NAME OF THE LABEL WHICH ADDRESS THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
**********************************************************************
        AIF (D'SHOW_START).NONEED
        B  BY_SHOW_START
SHOW_START DS 0H
        ST R10,COMRET
        LA 6,COMSHVB
        USING SHVBLOCK,R6
        XC COMSHVB(SHVBLEN),COMSHVB
        XC SHVNEXT,SHVNEXT
        MVI SHVCODE,C'S'
        BR 14
BY_SHOW_START DS 0H
.NONEED ANOP
        BAL 14,SHOW_START
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME'
        AIF   (D'&LABCHECK).BYPASS
@_&ASNAME DC C'&ASNAME'
.BYPASS ANOP
&SYSECT LOCTR
        LA 1,@_&ASNAME
        ST 1,SHVNAMA
        LA 1,L'@_&ASNAME
        ST 1,SHVNAML
        LA 1,&LABEL
        ST 1,SHVVALA
        LA 1,L'&LABEL
        ST 1,SHVVALL
        LR 0,10
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ abend001
        MEND
        EJECT
UCBPING TITLE 'REXX FUNCTION FOR A QUICK CHECK OF DEVICE/PATH STATUS'
UCBPING  AMODE 31
***********************************************************************
* This routine accepts two parameters. The device address and the
* path number to check. Using the MVS V5 path validation service
* a 'single ping' I/O is executed to the device to check out device
* and path status.
* Four variables are created by this routine as follows:
* RC .............. Return code. Values for this can be:
*                   0 ===> Function successfully performed.
*                   4 ===> supplied device number incorrectly
*                          specified. Must be no more than 4
*                          characters in length.
*                    8 ===> As return code 4, but its the path number
*                           which was incorrectly specified.
*                    99 ==> Not all parameters supplied.
* RETURN ........... The return code information from the path
*                    validation service. Used in conjunction with the
*                    next variable (REASON) it gives a description of
*                    the status.
* REASON ........... Linked to the above it gives a status description
*                    as follows:
*                    RETURN=0 ===> path physically available
*                    RETURN=4, REASON=4 ===> path NOT physically avail
*                    RETURN=4, REASON=8 ===> Device took too long to
*                                            respond
*                    RETURN=8, REASON=4 ===> Device not known.
*                    RETURN=8, REASON=8 ===> path not known.
*                    RETURN=8, REASON=12 ==> Should never happen as
*                                            this indicates a coding
*                                            fault in the validation
*                                            service.
*                    RETURN=8, REASON=36 ==> IOS address space not
*                                            available.
*                    RETURN=8, REASON=40 ==> Insufficient storage for
*                                            request.
*                    RETURN=12 ===> In the event of this being seen
*                                   there is some form of system
*                                   fault. According to the manual
*                                   this is a contact IBM situation.
* MESSAGE .......... This variable may or may not contain data
*                    depending upon the current detected problems.
*                    If present it will contain a system message
*                    to further explain the device/path situation.
***********************************************************************
UCBPING  CSECT
         REXREGS
         BAKR  R14,R0
         LR    R12,R15
         USING UCBPING,r12
         LR    R10,R0                 * R10 --> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R10
*
         LR    R11,R1                 * R11 --> A(PARAM LIST (EFPL))
         USING EFPL,R11
*
         L     R9,ENVBLOCK_IRXEXTE    * R9 --> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
*
         L     R6,EFPLARG             * R6 --> A(ARGUMENT TABLE)
         USING ARGTABLE_ENTRY,R6
         L     R7,EFPLEVAL
         L     R7,0(R7)             *R7 --> A(EVALUATION BLOCK
         USING EVALBLOCK,R7
*
         STORAGE OBTAIN,LENGTH=OBLEN,ADDR=(8)
         USING COMSDS,R8
*
* PREPARE THE GOTTEN AREA FOR USE
*
         XC    COMS(OBLEN),COMS      * SET TO LOW VALUES
         LA    R15,COMID
         ST    R15,COMS
         LA    R15,COMDUMMY
         ST    R15,COMS+4
         ST    R15,COMS+8
         LA    R15,COMSHVB
         ST    R15,COMS+12
         LA    R15,COMRET
         ST    R15,COMS+16
         OI    COMS+16,X'80'           * INDICATE END OF PARMS
         MVC   COMID,=C'IRXEXCOM'
*
         CLC   ARGTABLE_ARGSTRING_PTR(8),=2F'-1' *END OF ARGS?
         BE    RCNOK              * YES SO 99 RCODE
*
         L     R2,ARGTABLE_ARGSTRING_PTR    * R2 --> A(ARGUMENT)
         L     R1,ARGTABLE_ARGSTRING_LENGTH *R1 --> L(ARGUMENT)
*
         C     R1,=F'4'    * DEVICE MUST BE NOT MORE THAN 4 CHARS
         BH    RCN4        * INDICATE ERROR
*
         SLL   R1,2        * MULTIPLY BY 4
         MVC   DEVICE,=C'0000' * INITIALISE DEVICE FIELD
         B     MOVE_B(R1)  * GO TO DEPENDING ON SET
*
MOVE_B   DS    0H
         B     RCN4   * CANNOT BE A ZERO ADDRESS
         B     LAST_BYTE
         B     THIRD_BYTE
         B     SECOND_BYTE
         B     FIRST_BYTE
*
LAST_BYTE DS 0H
         MVC  DEVICE+3(1),0(R2)
         B    GET_CHP
*
THIRD_BYTE DS 0H
         MVC  DEVICE+2(2),0(R2)
         B    GET_CHP
*
SECOND_BYTE DS 0H
         MVC  DEVICE+1(3),0(R2)
         B    GET_CHP
*
FIRST_BYTE DS 0H
         MVC  DEVICE,0(R2)
*
GET_CHP   DS 0H
*
         LA    R6,ARGTABLE_NEXT
         CLC   ARGTABLE_ARGSTRING_PTR(8),=2F'-1' *END OF ARGS?
         BE    RCNOK              * YES SO 99 RCODE
*
         L     R2,ARGTABLE_ARGSTRING_PTR    * R2 --> A(ARGUMENT)
         L     R1,ARGTABLE_ARGSTRING_LENGTH *R1 --> L(ARGUMENT)
*
         C     R1,=F'4'
         BH    RCN8
         SLL   R1,2
         MVC   CHPID,=C'0000' * INITIALISE CHANNEL FIELD
         B     MOVE_C(R1)  * GO TO DEPENDING ON SET
*
MOVE_C   DS    0H
         B     RCN8   * CANNOT BE A ZERO ADDRESS
         B     LAST_BYTE_C
         B     THIRD_BYTE_C
         B     SECOND_BYTE_C
         B     FIRST_BYTE_C
*
LAST_BYTE_C DS 0H
          MVC  CHPID+3(1),0(R2)
          B    DO_PACKS
*
THIRD_BYTE_C DS 0H
          MVC  CHPID+2(2),0(R2)
          B    DO_PACKS
*
SECOND_BYTE_C DS 0H
          MVC  CHPID+1(3),0(R2)
          B    DO_PACKS
*
FIRST_BYTE_C DS 0H
          MVC  CHPID,0(R2)
*
DO_PACKS  DS 0H
*
*** CONVERT THE DEVICE NUMBER TO BINARY FORMAT. NOTE NOT A CONVENTIONAL
*** CONVERSION SINCE AN INPUT F0F2F4C1 CONVERTS TO X'024A'.
*** THEREFORE A TRANSLATE IS REQUIRED.
*
                    TR   DEVICE,TRANTAB
*
```

```
*** NOW F0F2F4C1 IS 0002040A. THEREFORE NEED THE SECOND NIBBLE OF EACH
*** BYTE
*
         PACK PACKER,DEVICE(5)
         MVC  BINDEV,PACKER+5    * MOVE RELEVANT BIT
*
*** CONVERT THE CHANNEL PATH NUMBER TO BINARY USING SIMILAR TECHNIQUE
*
         TR   CHPID,TRANTAB
         PACK PACKER,CHPID(5)
         MVC  BINCHP,PACKER+5
*** HAVING OBTAINED THE INPUT VARIABLES NOW ANALYSE THEM
*** WILL HAVE TO BE SUPERVISOR STATE TO DO THIS
*** THEREFORE USE AUTHORISING SVC TO DO THIS
         SVC 235 * <=== SET TO YOUR APF ON SVC NUMBER
         MODESET MODE=SUP
         IOSPTHV DEVN=BINDEV,CHPID=BINCHP+1,RETCODE=RCODE,RSNCODE=RSN, X
             MSGBUF=MESSBUFF,MF=(E,LISTFORM)
*
*** NOW GET BACK TO NORMAL
*
         MODESET MODE=PROB
         SVC 236  * <=== SET TO YOUR APF OFF SVC NUMBER
         SHOW RCODE,RETURN
         SHOW RSN,REASON
         SHOW MESSBUFF,MESSAGE
         SHOW RC0,RC
ENDREXX  DS 0H
         STORAGE RELEASE,LENGTH=OBLEN,ADDR=(8)
         PR
RCNOK    DS 0H
         SHOW RC99,RC
         B  ENDREXX
RCN4     DS 0H
         SHOW RC4,RC
         B  ENDREXX
RCN8     DS 0H
         SHOW RC8,RC
         B  ENDREXX
*
*** Routine used should there be a REXX problem
*
ABEND001 DS 0H
         ABEND 1
         LTORG
LOAD_POINT DS F
RC99     DC C'99'
RC0      DC C'0'
RC4      DC C'4'
RC8      DC C'8'
```

```
TRANTAB  DC 256X'00'
         ORG TRANTAB+X'C1'
         DC X'0A0B0C0D0E0F'
         ORG TRANTAB+X'F0'
         DC X'00010203040506070809'
         ORG
**********************************************************************
***      IRXEXCOM PARAMETER AREA                                   ***
**********************************************************************
COMSDS   DSECT
COMS     DS    5AL4
COMID    DS    CL8             * IRXEXCOM ID - C'IRXEXCOM'
COMDUMMY DS    AL4             * NOT USED
COMSHVB  DS    (SHVBLEN)X      * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET   DS    AL4             * IRXECOM RC
COMSLEN  EQU *-COMS
DEVICE   DS    CL4
         DS    X
CHPID    DS    CL4
         DS    X
RCODE    DS    F
RSN      DS    F
BINDEV   DS    CL3
BINCHP   DS    CL3
PACKER   DS    CL8
MESSBUFF DS CL48
LISTFORM IOSPTHV MF=(L,PATHAREA)
OBLEN    EQU *-COMS
         DS 0D
         IRXEFPL
         IRXARGTB
         IRXEVALB
         IRXENVB
         IRXEXTE
         IRXSHVB
         END
```

THE REXX CODE (PATHVAL)

```
/* REXX */
/* */
ADDRESS ISPEXEC
zwinttl='Device path validation'
looper:
call pop_request
if reply='END' then exit
call ucbping devc,chpd
return=c2d(return)
reason=c2d(reason)
```

```
if return=0 then mymess='path physically available'
else if return=4 & reason=4 then mymess='path not physically available'
else if return=4 & reason=8 then mymess='I/O took too long'
else if return=8 & reason=4 then mymess='device number not known'
else if return=8 & reason=8 then mymess='path number not known'
else if return=8 & reason=12 then mymess='iospthv coding error'
else if return=8 & reason=36 then mymess='ios address space unavailable'
else if return=8 & reason=40 then mymess='insufficient storage ,
          for request'
else if return=12 then mymess='Real funny. Contact IBM'
else mymess='Should not appear. Error in manual'
call pop_request1
if reply='END' then exit
signal looper
pop_request:
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(DASDP7)'
'REMPOP'
RETURN
pop_request1:
'ADDPOP ROW(1) COLUMN(9)'
'DISPLAY PANEL(DASDP71)'
'REMPOP'
RETURN
```

THE FIRST PANEL DASDP7

```
)Attr Default(%+_)
)Body Window(70,2)
%Specify Device Number ===>_devc+
%Specify test channel  ===>_chpd+
)init
.help=dasdhp7
)proc
&reply=.resp
         VER (&devc,NB)
         VER (&chpd,NB)
 )End
```

THE SECOND PANEL DASDP71

```
)Attr Default(%+_)
% type(output) intens(high)
)Body Window(70,4)
+Results of path validation service for device%devc+channel%chpd
+
%mymess
```

```
%message
)init
.help=dasdhp7
)proc
&reply=.resp
)End
```

THE HELP PANEL DASDPHP7

```
)BODY
`------------- HELP PANEL FOR SYSTEMS FUNCTION ------------------------
+
+This function issues a single ping IO to the device to get the latest
+configuration status for the device. The information returned consists
+of one or two lines of information. The first line which is always
+produced gives a one line description of the status.
+The second line is sometimes produced by the path validation service
+and gives additional diagnostic information regarding device and path
+status.
)PROC
.help=isp00004
)END
```

THE TWO SVCS

Each of the following SVCs (assuming you haven't already got similar code) will need installing into the LPA. Remember to change the associated code in the REXX function to match whatever SVC numbers you choose.

THE SWITCH APF ON SVC

```
AUTOSVC  CSECT
AUTOSVC  AMODE 31
         USING *,6
         USING TCB,4
          L    3,TCBJSCB     * ADDRESS THE JSCB
          USING IEZJSCB,3
          OI JSCBOPTS,JSCBAUTH
          BR 14
          LTORG
          PRINT NOGEN
          IKJTCB
          IEZJSCB
```

```
AUTOSVC1 CSECT                 ADDR
AUTOSVC1 AMODE 31
        USING *,6
        USING TCB,4
        L    3,TCBJSCB     * ADDRESS THE JSCB
        USING IEZJSCB,3
        NI JSCBOPTS,X'FF'-JSCBAUTH
        BR 14
        LTORG
        PRINT NOGEN
        IKJTCB
        IEZJSCB
```

C A Jacques
Systems programmer (UK)

Year 2000 aid: change JCL dates – part 1

INTRODUCTION

This program, YEAR2KC, reads a PDS, identifies EXEC statements, and determines if these statements contain ‘DATE=’ fields within a ‘PARM=’ operand. When such fields are found they are modified to a specified date. Dates may be changed from eight character format (eg MM/DD/YY) to ten character formats (eg MM/DD/CCYY), and conversely. Hence, this program may be used to convert to ten character date formats or may be used to simply change the dates for normal production runs. See the TODAY option, below, for this later usage. The program options are controlled by PARM= fields, as follows:

- DATE=string, specifies the date to replace those found in the JCL members.

If ‘string’ is a character string, TODAY results in the replacement date being that of the execution date. See the FMT= parameter to use an alternative format of the date.

Otherwise, the string is examined for date format validity as defined by the default or explicit FMT= format specification.

- FMT=string, specifies the date format of the replacement string. Valid values are:
  - MM/DD/CCYY, (default)
  - CCYY/MM/DD
  - MM/DD/YY
  - YY/MM/DD.
- PRNT=string or PRNT=(string1,...,stringn) specifies the different print option(s), the valid values for stringi are:
  - BEFORE – to list the image prior to changes.
  - AFTER – to list the image after changes are made.
  - LIST – to list all records, regardless of changes.
  - DIAG – to provide diagnostic traces of statement parsing. This is intended for testing only.
- FROM=member, specifies that processing of the PDS is to begin with the member name member. The default is the first member of the PDS.
- THRU=member, specifies that processing of the PDS is to end after member is processed.

ERROR CONDITIONS AND REPORT

If the long format is to replace the short format, the statement is analysed to see if space is available between its fields or at the end of the statement. If insufficient space is found, the replacement is made by removing characters from the end of the statement. In this case the before and after images are written to report ERRORS and a message is appended to the normal output report (PRINTER).

SAMPLE JCL

```
//SYSTØØ2I JOB ...
//*------------------------------------------------------------*//
//STEP1    EXEC PGM=YEAR2KC,PARM='FMT=MM/DD/CCYY,DATE=11/11/1996'
//SYSABEND DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//PRINTER  DD  SYSOUT=*
//ERRORS   DD  SYSOUT=*
//PDS      DD  DSN=SYSTØØ2.ESAEDIT.LIBRARY,DISP=SHR
//CARDS    DD    *
L8Ø*
//
```

PROGRAM SOURCE

```
         LCLC  &MYNAME
&MYNAME  SETC  'YEAR2KC'                CSECT NAME
RBASE    EQU   12                       BASE REGISTER FOR CSECT
RBAL     EQU   1Ø                       BAL REGISTER
         TITLE '&MYNAME'                LISTING TITLE
***********************************************************************
***   THIS PROGRAM SEARCHES JCL PDS MEMBERS FOR 'DATE=' PARMS ON    ***
***   EXEC STATEMENT AND REPLACES THE SPECIFIED VALUE WITH THE      ***
***   VALUE SPECIFIED IN THE CURRENT 'DATE=' PARM.                  ***
***                                                                 ***
***   IF THE PARM SPECIFIES AN 8 DIGIT DATE, ALL 6 DIGIT DATES      ***
***   FOUND WILL BE ADJUSTED TO 8 DIGIT VALUES.  IN THIS CASE,      ***
***   IF THERE DOESN'T APPEAR TO BE SUFFICIENT SPACE TO MOVE THE    ***
***   CONTENTS TO THE RIGHT (IE COLUMNS 71 & 72 ARE NOT BLANK)   ***
***   THE REPLACEMENT IS MADE AND A WARNING IS MADE THAT DATA MAY   ***
***   BE LOST.                                                      ***
***********************************************************************
         EJECT
***********************************************************************
***      LINKAGE CONVENTIONS ENTERING PROGRAM                       ***
***********************************************************************
         MACRO
&LABEL   SMUMØØ2  &DSECT=YES,&C=Ø
         PUSH  PRINT
         PRINT GEN
.*********************************************************************
.*  MACRO TO DESCRIBE PDS BLDL ENTRY WITH ISPF STATISTICS,          ***
.*  TO BE USED BY 'BLDL' MACRO.                                     ***
.*                                                                  ***
.*  DSECT=YES   WILL CAUSE A DSECT TO BE CREATED.                   ***
.*  DSECT=NO    DATA WILL BEGIN ON A DOUBLEWORD BOUNDRY.            ***
.*  C=_         LABELS WILL BE GU_2XX (_ MAY BE ANY ALPHAMERIC      ***
.*              CHARACTER(S),  INTENDED FOR GENERATING MULTIPLE     ***
.*              COPIES OF THE GENERATED LAYOUT).                    ***
.*                                                                  ***
.*** THIS MACRO IS A MODIFICATION TO 'GTEUMØ2' FROM THE             ***
.*** CONNECTICUT BANK TAPE.  THE IMPLEMENTATION OF THIS SOURCE      ***
.*** MANAGEMENT SYSTEM WAS MUCH EASIER BY UTILIZING THIS EXISTING   ***
.*** CODE.  MUCH GRADITUDE AND APPRECIATION IS GIVEN TO:            ***
.*                                                                  ***
.*  CHUCK HOFFMAN, SYSTEMS PROGRAMMING, GTEL COMPUTING CENTER       ***
.*                                                                  ***
.*  MODIFICATION OF HIS MACRO ON THE CONNECTICUT BANK TAPE EASED    ***
.*  THE IMPLEMENTATION OF THIS SYSTEM.                              ***
.*********************************************************************
         AIF   ('&DSECT' EQ 'YES').GUMØ2A
&LABEL   DS    ØD                      , ISPF STATS PDS BLDL ENTRY
         AGO   .GUMØ2B
.GUMØ2A  ANOP
&LABEL   DSECT                         , ISPF STATS PDS BLDL ENTRY
.GUMØ2B  ANOP
.*
GU&C.2FF DS    XL2                       BLDL COUNT OF ENTRIES
GU&C.2LL DS    XL2                       BLDL LENGTH OF ENTRIES
GU&C.2NAM DS   CL8                       MEMBER NAME
GU&C.2TTR DS   XL3                       PDS VALUE 'TTR'
GU&C.2K  DS    X                         BLDL VALUE 'K'
GU&C.2Z  DS    X                         BLDL VALUE 'Z'
GU&C.2C  DS    X                         PDS VALUE 'C'
GU&C.2VER DS   X                         ISPF VERSION NUMBER (BIN)
GU&C.2MOD DS   X                         ISPF MOD NUMBER (BIN)
         DS    XL2                       (UNUSED, X'ØØØØ')
GU&C.2DATC DS  PL4                       ISPF DATE CREATED (PACK)
GU&C.2DATM DS  PL4                       ISPF DATE MODIFIED (PACK)
GU&C.2TIMM DS  XL2                       ISPF TIME MODIFIED (PK NOSIGN)
GU&C.2SIZE DS  XL2                       ISPF SIZE (BIN)
GU&C.2INIT DS  XL2                       ISPF INITIAL SIZE (BIN)
GU&C.2MODL DS  XL2                       ISPF COUNT OF MOD LINES (BIN)
GU&C.2ID DS    CL7                       ISPF USERID
         DS    CL3                       (UNUSED X'4Ø4Ø4Ø')
         POP   PRINT
         MEND
&MYNAME  CSECT ,
         STM   R14,R12,12(R13)           SAVE REGS TO CALLER S.A.
         B     (BEGIN-&MYNAME)(R15)      BRANCH AROUND EYECATCHER
         DC    A(L'NAME)                 LENGTH OF CSECT NAME
NAME     DC    C'&MYNAME'                CSECT NAME
         DC    C' &SYSDATE &SYSTIME '    ASSEMBLY DATE/TIME STAMP
BEGIN    LR    RBASE,R15                 LOAD BASE REGISTER
         USING &MYNAME,RBASE               ADDRESSABILITY
         PRINT NOGEN
         GETMAIN R,LV=WORKDLEN          GET SAVE/WORK AREA
         ST    R1,8(Ø,R13)              MY S.A. ADDR INTO CALLER S.A.
         ST    R13,4(Ø,R1)              CALLER S.A. ADDR INTO MY S.A.
         LR    R13,R1                   R13 POINTS TO MY S.A.
         USING WORKD,R13                ADDRESSABILITY OF SAVE AREA
         L     R1,4(Ø,R13)              R1 POINTS TO CALLER S.A.
         LM    R15,R1,16(R1)            R15 RØ AND R1 ARE RESTORED
         EJECT
***********************************************************************
***      MAINLINE ROUTINE                                           ***
***********************************************************************
MAIN     EQU   *                        BEGIN MAINLINE ROUTINE
         ST    R1,R1SAVE                SAVE INITIAL R1
         XC    COMPCODE,COMPCODE        CLEAR COMPLETION CODE
*
         L     R1,=A(INITIAL)      POINT TO INITIALIZATION ROUTINE
         BALR  RBAL,R1             GO PERFORM INITIZLIZATION
*
MAINDIRL BAL   RBAL,GETDIR         GET MEMBER NAME
         LTR   R15,R15             END OF DIRECTORY REACHED?
         BNZ   MAINEND              YES
         MVI   SWITCHES,Ø          CLEAR ALL CONDITION FLAGS
         ZAP   CARDS,=P'Ø'         INITIALIZE RECORD COUNT
         L     R3,EXCLUDE1         POINT TO CURRENT EXCLUSION
         LR    R4,R3               POINT TO BEGINNING OF MEMBER NAME
         LA    RØ,7                MAXIMUM LENGTH-1
MAINWC   CLI   1(R4),C'*'          WILD CARD PATTERN?
         BE    MAINWCX              YES
         LA    R4,1(R4)            POINT TO NEXT CHARACTER
         BCT   RØ,MAINWC           CONTINUE
MAINWCX  SR    R4,R3               GET LENGTH-1
MAINXL   EX    R4,MAINXCLC         IS MEMBER TO BE EXCLUDED?
         BL    MAINNX               NO
         BH    MAINXMB              MAYBE
         AP    EXCLUDED,=P'1'      COUNT EXCLUSION
         MVC   LINE+9(8),MEMBER    MOVE MEMBER NAME TO OUTPUT LINE
         MVC   LINE+18(8),=C'EXCLUDED' SET EXCLUSION MESSAGE
         MVC   LINE+26(6),EDITPAT   SET EDIT PATTERN
         ED    LINE+26(6),EXCLUDED  FORMAT EXCLUSION COUNT
         MVI   LINE,C'Ø'           SET TO DOUBLE SPACE
         BAL   RBAL,DOUBLESP       ALLOW FOR DOUBLE SPACE
         BAL   RBAL,PRINT          GO PRINT LINE
         B     MAINDIRL            GO GET NEXT MEMBER
MAINXCLC CLC   MEMBER(*-*),Ø(R3)   IS MEMBER TO BE EXCLUDED?
MAINXMB  LA    R3,L'EXCLUDES(R3)   POINT TO NEXT ENTRY
         ST    R3,EXCLUDE1         SAVE POSITION
         B     MAINXL              GO CHECK
MAINNX   ST    R15,INRECLOC        INITIALIZE FOR GETREC
MAINNXTR BAL   RBAL,GETREC         READ RECORD FROM CURRENT MEMBER
         LTR   R15,R15             END OF MEMBER REACHED?
         BNZ   MAINDIRL             YES
         BAL   RBAL,SCANREC        SCAN RECORD FOR DATE= PARMS
         TM    SWITCHES,UPDATBIT   RECORD MODIFIED?
         BZ    MAINNXTR             NO
         NI    SWITCHES,X'FF'-UPDATBIT TURN OFF UPDATE BIT
         BAL   RBAL,WRITEREC       UPDATE RECORD
         B     MAINNXTR            GO GET NEXT RECORD
MAINEND  DS    ØH
         BAL   RBAL,HEADPAGE       PUT TOTALS ON NEW PAGE
         MVC   LINE+5(6),EDITPAT   SET EDIT PATTERN
         ED    LINE+5(6),MEMBERS   FORMAT MEMBER NUMBER
         MVC   LINE+12(13),=C'MEMBERS FOUND'
         BAL   RBAL,PRINT          PRINT TOTAL
         MVC   LINE+5(6),EDITPAT   SET EDIT PATTERN
         ED    LINE+5(6),EXCLUDED  FORMAT MEMBER NUMBER
         MVC   LINE+12(16),=C'MEMBERS EXCLUDED'
         BAL   RBAL,PRINT          PRINT TOTAL
         MVC   LINE+5(6),EDITPAT   SET EDIT PATTERN
         SP    MEMBERS,EXCLUDED    COMPUTE REMAINDER
         ED    LINE+5(6),MEMBERS   FORMAT MEMBER NUMBER
         MVC   LINE+12(16),=C'MEMBERS ANALYZED'
         BAL   RBAL,PRINT          PRINT TOTAL
         MVC   LINE+5(6),EDITPAT   SET EDIT PATTERN
         ED    LINE+5(6),MODIFIED  FORMAT MEMBERS MODIFIED
         MVC   LINE+12(16),=C'MEMBERS SELECTED'
         BAL   RBAL,PRINT          PRINT TOTAL
         MVI   LINE,C'Ø'           SET TO DOUBLE SPACE
         BAL   RBAL,DOUBLESP       ALLOW FOR DOUBLE SAPCE
         MVC   LINE+1(1Ø),OCCUR1   SET EDIT PATTERN
         ED    LINE+1(1Ø),TRECS    FORMAT TOTAL RECORD COUNT
         MVC   LINE+12(16),=C'RECORDS ANALYZED'
         BAL   RBAL,PRINT          PRINT TOTAL
         MVC   LINE+1(1Ø),OCCUR1   SET EDIT PATTERN
         ED    LINE+1(1Ø),TFINDS   FORMAT TOTAL RECORDS SELECTED
         MVC   LINE+12(16),=C'RECORDS SELECTED'
         BAL   RBAL,PRINT          PRINT TOTAL
         CP    ERRORTOT,=P'Ø'      ANY ERRORS?
         BNH   MAINNONE             NO
         MVC   LINE(2),=C'Ø*'      SET DOUBLE SPACE/SEED
         MVC   LINE+3(L'LINE-3),LINE+1 SET '* * *'...
         BAL   RBAL,PRINT          PRINT FLAG
         MVC   LINE(4Ø),=C'Ø*** WARNING ***:  SEE ''ERRORS'' FILE FOR'
         MVC   LINE+4Ø(L'EDITPAT),EDITPAT SET EDIT PATTERN
         ED    LINE+4Ø(L'EDITPAT),ERRORTOT FORMAT COUNT
         MVC   LINE+41+L'EDITPAT(16),=C'POSSIBLE ERRORS.'
         BAL   RBAL,PRINT          PRINT FLAG
MAINNONE DS    ØH
* BEGIN DCB CLOSE
         CLOSE (PRINTER),MF=(E,PRCLOSL) CLOSE IT
         CLOSE (PDSDIR),MF=(E,DRCLOSL)  CLOSE PDSDIR
         CLOSE (PDS),MF=(E,PDCLOSL)     CLOSE PDS
         CLOSE (ERRORS),MF=(E,ERCLOSL)  CLOSE ERRORS
```

```
* END DCB CLOSE
ENDØØ    LA    R15,Ø                    SET COMPLETION CODE ØØ
         ST    R15,COMPCODE               INTO STORAGE
         B     ENDING                   GO TO ENDING
         EJECT
**********************************************************************
***      LINKAGE CONVENTIONS EXITING PROGRAM                       ***
**********************************************************************
ENDING   L     R14,COMPCODE             R14 SAVES COMP CODE
         LR    R1,R13                   R1 SAVES ADDR OF MY S.A.
         L     R13,4(Ø,R1)              R13 RESTORED, PTR CALLER S.A.
         FREEMAIN R,LV=WORKDLEN,A=(R1)  FREE MY SAVE/WORK AREA
         LR    R15,R14                  R15 SET TO COMP CODE
         LM    RØ,R12,2Ø(R13)           RØ-R12 RESTORED
         L     R14,12(Ø,R13)            R14 RESTORED
         MVI   12(R13),X'FF'            SET COMPLETION SIGNAL
         BR    R14                      RETURN TO CALLER
* BEGIN STUB DEFINE
         EJECT
**********************************************************************
***   GET DIRECTORY RECORD                                         ***
**********************************************************************
GETDIR   ST    RBAL,SAVGDBAL       SAVE LINKAGE REGISTER
         CLI   DFLAG,Ø             FIRST TIME?
*        BNE   GDNOT1ST             NO
         MVI   DFLAG,X'FF'         SET FLAG
GDRD     BAL   RBAL,READDIR        READ DIRECTORY RECORD
         LTR   R15,R15             NORMAL RETURN?
*        BNZ   GDRETURN             NO
         BNZ   GDEND                NO
GDNOT1ST L     R2,DIRENTRY         LOAD ADDRESS OF MEMBER DATA
         AP    TRECS,RECORDS       ACCUMULATE TOTAL RECORDS PROCESSED
         ZAP   RECORDS,=P'Ø'       CLEAR MEMBER RECORD COUNT
         AP    MEMBERS,=P'1'       COUNT NUMBER OF MEMBERS
         CLI   Ø(R2),X'FF'             END OF DIRECTORY BLOCK?
         BE    GDRD                     YES
         MVC   MEMBER,Ø(R2)            MOVE MEMBER NAME TO OUTPUT AREA
         XR    R15,R15             SET NORMAL RETURN
GDRETURN L     RBAL,SAVGDBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
GDEND    LA    R15,4               SET END-OF-DIRECTORY EXIT
         B     GDRETURN            GO EXIT
         EJECT
**********************************************************************
***   READ DIRECTORY RECORD                                        ***
**********************************************************************
READDIR  ST    RBAL,SAVRDBAL       SAVE LINKAGE REGISTER
         L     R6,DIRENTRY         LOAD ADDRESS OF CURRENT LOCATION
         LTR   R6,R6               FIRST DIRECTORY BLOCK?
         BZ    RDNXTDIR             YES
         MVI   LINE,C'Ø'           SET TO DOUBLE SPACE
         BAL   RBAL,DOUBLESP       ALLOW FOR DOUBLE SPACE
         MVC   LINE+1(6),EDITPAT   SET EDIT PATTERN
         ED    LINE+1(6),MEMBERS   FORMAT MEMBER NUMBER
         MVC   LINE+9(8),MEMBER    MOVE MEMBER NAME TO OUTPUT LINE
         MVC   LINE+18(LOCCURS),OCCURS
         ED    LINE+18+OCCUR1-OCCURS(L'OCCUR1),RECORDS FORMAT RECORDS
         ED    LINE+18+OCCUR2-OCCURS(L'OCCUR2),FINDS " FIND OCCURRENCES
         BAL   RBAL,PRINT          PRINT MEMBER HEADING LINE
         CP    FINDS,=P'Ø'         ANY FINDS?
         BZ    RDNXTMEM             NO
         BAL   RBAL,GETSTATS       GET MEMBER STATISTICS
         LTR   R15,R15             STATS OKAY?
         BNZ   RDNOSTAT             NO
         BAL   RBAL,PUTSTATS       PRINT MEMBER/STATS
RDNOSTAT AP    TFINDS,FINDS        ACCUMULATE GRAND TOTAL
         ZAP   FINDS,=P'Ø'         RESET COUNTER
         AP    MODIFIED,=P'1'      COUNT MEMBERS MODIFIED
         B     RDNXTMEM            GO GET NEXT ENTRY
RDNXTDIR GET   PDSDIR,DIRBLOCK     READ DIRECTORY RECORD
         LA    R6,DIRBLOCK+2       POINT TO ENTRY
         ST    R6,DIRENTRY         SAVE ADDRESS (NOT REALLY NEEDED)
         LH    R5,DIRBLOCK         LOAD NUMBER NUMBER OF BYTES USED
         STH   R5,DIRSPACE         SAVE
         SH    R5,=H'2'            REDUCE BY LENGTH OF FIELD
         BNP   RDNXTDIR            IF EMPTY DIRECTORY BLOCK, GO TO NEXT
         B     RD1STMEM            GO PROCESS FIRST ENTRY IN BLOCK
RDNXTMEM L     R6,DIRENTRY         LOAD ADDRESS OF CURRENT LOCATION
         LH    R5,DIRSPACE         LOAD REMAINING SPACE IN BLOCK
         IC    R1,11(R6)           LOAC 'C' FIELD
         N     R1,=F'31'           GET USER AREA HALFWORDS (5 LOW BITS)
         LA    R1,12(R1,R1)        BYTES + MEMBER NAME, 'TTR', AND 'C'
         SR    R5,R1               DEDUCT CURRENT ENTRY LENGTH
         AR    R6,R1               POINT TO NEXT ENTRY
RD1STMEM CLI   Ø(R6),X'FF'         LAST DIRECTRY ENTRY?
         BE    RDDIREND             YES
         CH    R5,=H'14'           ROOM FOR ADDITIONAL ENTRIES?
         BL    RDNXTDIR             NO
         ST    R6,DIRENTRY         SAVE CURRENT POINTER
         STH   R5,DIRSPACE         SAVE REMAINING SPACE
         MVC   TTRN,8(R6)          SAVE RELATIVE DASD ADDRESS
*        MVI   TTRN+3,Ø            CLEAR 'N'
         CLI   TTRN+2,Ø            VALID ADDRESS?
         BNE   RDOKAY               YES
         MVC   LINE+2(8),Ø(R6)     SET MEMBER NAME
         MVC   LINE+11(9),=C'NOT FOUND' SET ERROR MESSAGE
         MVI   LINE,C'Ø'           SET TO DOUBLE SPACE BEFORE PRINT
         BAL   RBAL,DOUBLESP       ALLOW FOR DOUBLE SPACE
         BAL   RBAL,PRINT          PRINT ERROR LINE
         B     RDNXTDIR            GO PROCESS REMAINDER OF LIST
*DOKAY   POINT PDS,TTRN            POINT TO NOTE LIST RECORD
RDOKAY   FIND  PDS,(R6),D          POINT TO NOTE LIST RECORD
```

```
         XR    R15,R15             CLEAR RETURN CODE
RDRETURN L     RBAL,SAVRDBAL       RESTORE LINKAGE REGISTER
         BR    RBAL                RETURN
RDDIREND LA    R15,4               INDICATE END OF DIRECTORY
         B     RDRETURN            GO RETURN
         EJECT
***********************************************************************
***   READ RECORD FROM MEMBER                                       ***
***********************************************************************
GETREC   ST    RBAL,SAVGRBAL       SAVE LINKAGE REGISTER
         L     R1,INRECLOC         POINT TO RECORD LOCATION
         LTR   R1,R1               FIRST RECORD OF MEMBER?
         BNZ   GRNXTREC             NO
GRNXTBLK LA    R2,DECBA            POINT TO DECB
         L     R3,BLOCKLOC         POINT TO AREA ADDRESS
         ST    R3,INRECLOC         SAVE RECORD POINTER
         READ  (R2),SF,PDS,(R3),MF=E  READ BLOCK FROM MEMBER
         CHECK (R2)                AWAIT ECB POSTING
         LH    R5,INLRECL          LOAD RECORD LENGTH
         LH    R3,INBLKSIZ         LOAD MAXIMUM BLOCK SIZE
         L     R1,DECBA+16         LOAD RECORD POINTER WORD (IOB)
         SH    R3,14(R1)           SUBTRACT REMAINING COUNT
         L     R1,BLOCKLOC         GET ADDRESS OF BLOCK
         AR    R3,R1               POINT TO END OF BLOCK
         BCTR  R3,Ø                POINT TO LAST BYTE OF BLOCK
         ST    R3,BLOCKEND         SAVE ENDING ADDRESS
         L     R1,INRECLOC         POINT TO BEGINNING OF BLOCK
         B     GR1STREC            GO PROCESS FIRST RECORD OF BLOCK
GRNXTREC L     R1,INRECLOC         GET PREVIOUS RECORD LOCATION
         AH    R1,INLRECL          POINT TO NEXT RECORD
         C     R1,BLOCKEND         PAST END OF BLOCK?
         BNL   GRNXTBLK             YES
GR1STREC ST    R1,INRECLOC         SAVE ADDRESS OF RECORD
         XR    R15,R15             SET 'RECORD FOUND' CODE
         AP    RECORDS,=P'1'       COUNT RECORD
```

Editor's note: this article will be continued next month when the rest of the code will be published

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)

MVS news

Princeton Softech has unveiled its Ager 2000 tool for date ageing MVS files and databases. The software supports both linear and semantic date ageing, prompts for formats and business rules, allows users to modify business rules for the ageing process, and calculates holidays and significant business dates automatically, supporting table entry for special dates that cannot be calculated.
Also, it allows the holiday rules and calculations to be modified so that different units in different countries can customize the thing for their own needs. It can also read copybooks and capture the metadata it needs, support multiple record types per file, and handle records with redefine clauses, ODO definitions, and recursive ODOs, which might result in a variable number of dates being included in a given record. Ager 2000 will be available at the beginning of 1998, and is priced by MIPS, starting at $40,000.

For further information contact:
Princeton Softech, 1060 State Road, Princeton, NJ 08540-1423, USA
Tel: 609 497 0205
Fax: 609 497 0302

* * *

IBM has announced a maintenance tape for its SnapShot duplication solution. It contains PTFs for SnapShot for MVS/ESA Version 1 Release 2 and will be updated twice a year.
It will contain maintenance PTFs since the previous maintenance tape, and new accumulation of all maintenance PTFs created since the base product was packaged, all on the single tape. For instance, all PTFs for SnapShot MVS Version 1.1 and for SnapShot MVS Version 1.2 would be on one SnapShot MVS maintenance tape.
And, to help figure out which level of tape users have, the tape VOLSERs have a new numbering scheme, which is PPYYMM, where PP is for product identifier (SS for SnapShot), YY for year, and MM for the month the tape was created. The first maintenance tape is due out today.

Contact your local IBM marketing representative for further information.

* * *

Universal Software has added TCP/IP support to its Universal-Link file transfer and mailboxing system for MVS. Universal-Link also supports BSC/3780, Async, SNA/LU1, and LU6.2, enabling MVS and VSE sites to send and receive data between the host computer and various systems. Remote users cannot access the mainframe files or applications. The inclusion of TCP/IP protocol support allows MVS sites to exploit the economy of the Internet without compromising security and reliability.

For further information contact:
Universal Software Inc, 304 Federal Road, Brookfield Office Park, Brookfield, CT 06804, USA
Tel: 203 792 5100
Fax: 203 775 2897

* * *