147 MVS update

December 1998

In this issue

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

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

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

Editor
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

An accelerated string search

INTRODUCTION

Many applications expend considerable resources and time searching for strings (ie searching for text, table entries, etc). The following subroutine, called FASTFIND, has been optimized to reduce the overhead of such searches.

A STANDARD ALGORITHM

A simple algorithm for searching for a phrase in a string could be performed as follows (in pseudo-code):

```
SET string index TO 1
DO loop FOREVER
IF phrase EQ string(index) THEN GOTO found
INCREMENT index BY 1
END loop
notfound:
```

Typically, this type of simple algorithm would require:

- *n* compares (with the length of the phrase)
- *n* increments to the index
- *n* passes through the loop.

Where *n* is either the character position in the string where the phrase was found, or the length of the string (phrase was not found).

AN ENHANCED ALGORITHM

The optimized search method reduces overhead by using the following algorithm (in pseudo-code):

```
SET string index TO 1
DO loop FOREVER
Search for initial character of the phrase in the string (index)
IF NOT FOUND THEN GOTO notfound
SET index to the next found initial character
IF phrase EQ string(index) THEN GOTO found
INCREMENT index BY 1
END loop
```

This effectively reduces the number of comparisons for ‘phrase’ only when the initial character of <phase> is found in ‘string’. Typically this algorithm would require:

- *m* searches for the initial character of phrase
- *m* compares (with the length of the phrase)
- *m* increments to the index
- *m* passes through the loop.

Where *m* is the number of times the first character of the ‘phrase’ occurs in the ‘string’. A typical estimate for *m* could be approximately *n*/62. This assumes that <phase> and ‘string’ consist of alphanumeric characters that occur with uniform frequency (62 = 26 uppercase characters + 26 lowercase characters + 10 numerics). This simple estimate shows that the optimized algorithm could generally be expected to yield a 62-fold (6200 percent) improvement over the standard search algorithm.

In practice the relative performance of the two algorithms depends on the data. The best-case scenario for FASTFIND would be where ‘phrase’ is not present in ‘string’. With this scenario only a single instruction is performed (search for the initial character), which yields the NOTFOUND condition. This corresponds to the worst case for the standard algorithm (the complete string must be indexed).

The worst case scenario for FASTFIND would be a situation where ‘string’ contains only the initial character of ‘phrase’. This effectively equalizes the two algorithms, but with FASTFIND having the additional overhead of searching for the initial character. This is the only case where FASTFIND performs marginally worse than the conventional search.

The performance of FINDFAST increases the more often the initial character of ‘phrase’ is present in ‘string’.

IMPLEMENTATION

The TRT (Translate and Test) instruction is normally used to search for one or more characters (in this case the table would only contain the initial character of ‘phrase’ as non-zero data). The use of TRT has two problems:

- The search is limited to 256 characters. TRT must be used repeatedly if ‘string’ is longer.
- TRT cannot be used if the initial character of ‘phrase’ is X'00'.

The new SRST (Search String) machine instruction avoids these problems. However, remember that for performance reasons, ‘phrase’ cannot be longer than 256 characters. This limitation permits the use of a simple CLC instruction. Obviously a CLCL could be used in its place, although there would be additional overhead to set up the registers before each compare. There is no limitation on the length of ‘string’.

USE IN PRACTICE

Although the overhead for FASTFIND is not large (it mainly involves setting up the conditions for SRST), and it almost always performs better than the conventional search, there are some circumstances when its use is not appropriate. These would include scenarios where:

- The searched string is short (say less than 20 characters).
- The search is performed infrequently.

However, there are circumstances where the use of FASTFIND would be highly beneficial. These would include situations when:

- The searched string is large.
- The initial character of the search phrase occurs infrequently in the searched string.

INVOCATION

The command is called by:

```
CALL FASTFIND,(aphrase,lphrase,astring,lstring,ix)
```

Where ‘aphrase’ – address of ‘phrase’.

- ‘lphrase’ – length of ‘phrase’ (binary word).
- ‘astring’ – address to ‘string’.

- ‘lstring’ – length of ‘string’ (binary word).
- ix – index to the phrase found in ‘string’ (binary word).
- 0 = not found.
- >0 = index in ‘string’. 1 = first position, etc.
- The return value for FASTFIND (register 15) has the same content as ‘ix’.

SAMPLE USE

```
WORKING-STORAGE SECTION.
Ø1 PHR     PIC X(3) VALUE 'lai'.
Ø1 PHR-LEN PIC 9(8) BINARY VALUE 3.
Ø1 STR     PIC X(1ØØ) VALUE 'The rain in Spain lies mainly on the plain'
Ø1 STR-LEN PIC 9(8) BINARY VALUE 1ØØ.
Ø1 STR-IX  PIC S9(8) BINARY VALUE -1.

     CALL 'FASTFIND' USING PHR PHR-LEN STR STR-LEN STR-IX
      END-CALL
```

FASTFIND

```
         TITLE 'Function for accelerated string search'
* FASTFIND is a time-optimized function that
* searches for the specified phrase in a string.
* Invocation: CALL FASTFIND,(aphrase,lphrase,astring,lstring,ix)
* <aphrase>: address of <phrase>
* <lphrase>: length of <phrase> (fullword)
* <astring>: address to <string>
* <lstring>: length of <string> (fullword)
* <ix>: index of the found phrase (fullword). (Ø = not found).
* Result: Ø (False) - phrase not found
*        >1 (True)  - phrase found (the index is returned)
FASTFIND CSECT
FASTFIND AMODE 31
FASTFIND RMODE ANY
         BAKR  R14,Ø         save invocation registers
         BASR  R12,Ø         base register
         USING *,R12         address base register
         LM    R5,R9,Ø(R1)   load paramter addresses
         LR    R4,R7         save initial address of string
* R5: address of parm 1 (<phrase>)
* R7: address of parm 2 (<string>)
         L     R6,Ø(R6)      INT (length of <phrase>)
         SH    R6,=H'1'      lengthcode
```

```
* R6: length of parm 1 (<phrase>)
         L     R8,Ø(R8)      INT (length of <string>)
* R8: length of parm 2 (string to be searched)
* search for first byte of <phrase>
         LA    R15,Ø         set False as default
         SR    RØ,RØ         zero high-order
         IC    RØ,Ø(R5)      initial search byte (SRST argument)
         LR    R11,R7        start address of <string>
LOOP     LA    R1Ø,Ø(R7,R8)  end address of <string>
         SRST  R1Ø,R11       search for <phrase>(1) in <string>
         BO    *-4           interrupted, continue
         BH    NOTFOUND      first byte not found (end of search)
* R1Ø: First byte of <phrase> found
         EX    R6,COMPARE    test for complete <phrase>
         LA    R11,1(R1Ø)    restart address
         BNE   LOOP          <phrase> not found
FOUND    LR    R15,R1Ø       address of <phrase> in <string>
         SR    R15,R4        - start address = index
NOTFOUND ST    R15,Ø(R9)     return index
         PR    ,             return
         SPACE
COMPARE  CLC   Ø(Ø,R5),Ø(R1Ø) test for <phrase> in <string>
* register equates
RØ       EQU   Ø
R1       EQU   1
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R14      EQU   14
R15      EQU   15
         END
```


If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from our Web site. Point your browser at www.xephon.com/contnote.html.

A cross-partition dataset copying dialog

THE PROBLEM

Our site, probably in common with many others, has implemented its Year 2000 testing strategy by using multiple LPARs. Each production LPAR has a Year 2000 version of itself, with duplicate systems and data, the only major difference being that the Year 2000 LPAR has its system date advanced beyond 2000. This has proved to be a useful method of Y2K testing because it allows not only individual programs and systems to be tested, but also the interaction of all systems and programs with each other in a pseudo-production environment.

However, the problem existed of how to allow end-users and developers to copy datasets from the production LPARs to their equivalent Y2K LPARs simply and quickly. Because we do not have anything as advanced as Sysplex, or shared catalogs/shared DASD/GRS rings, and because of the problems there would be with sharing data between two systems with different system dates, we needed an alternative.

THE SOLUTION

We decided to have a common shared DASD volume (let us call it Y2KSHR), that is varied ONLINE to ALL LPARS, but is not referenced by any catalogs, SMS storage groups, or esoteric DASD pools.

Once we had this volume, we could then implement a ‘push-pull’ method of data transfer by utilizing the DFDSS data copying utility. The idea was as follows:

- On the source LPAR, submit a DFDSS job to dump all required datasets to a back-up dataset on the Y2KSHR volume.
- On the target Y2K LPAR, run a DFDSS restore job to read the back-up dataset directly from the Y2KSHR volume, not from a catalog (ie using UNIT=, VOL=SER= in the JCL), and restore all the datasets to the Y2KLPAR.
- Because back-up and restore are separate, using an intermediary storage dataset, the data is never accessed simultaneously by both LPARs, avoiding the problems of shared DASD and catalogs.

Also, the benefit of using DFDSS as the copy program is that it can handle any format of dataset from PDS to VSAM KSDS.

Once the DFDSS batch JCL had been worked out, we needed to wrap some sort of front-end around this to simplify matters for the developers and end-users. The end result was the Y2KCOPY dialog presented below. Features of Y2KCOPY include:

- A simple 'locking' mechanism utilizing a temporary member in the user's ISPFPROF dataset to prevent concurrent runs of the utility. This stops the temporary back-up datasets used to hold the copied files from being accidentally deleted before the copy has completed.
- Notification of copy progress by IEBGENERing status messages into the lock member after each job step completes. The messages are displayed if the user tries to access the Y2KCOPY dialog.
- Ability of the user to enter multiple dataset names or generic dataset selection filters, allowing many datasets to be copied simultaneously.
- Use of the ISPF LMDLIST service to estimate the total size of all selected datasets and calculate the required size of the temporary back-up dataset. This helps to reduce the incidence of X37 abends during the back-up phase.

There are some points to note with Y2KCOPY. For example:

- You will notice the qualifier 'NSMS' used in the temporary back-up dataset created. We adopted this to allow SMS ACS rules to be defined to bypass SMS processing for these datasets, and allow them to be written directly to the Y2KSHR DASD volume. Check with your storage administrator whether you have a similar naming convention.
- You do need to have JES NJE links between the partitions to allow transfer of the batch jobs. If you do not have this set up yet (although many sites do), consult the *JES2 Initialization and Tuning Guide* for details on setting up an NJE network.
- The dialog relies on each production LPAR having a 'partner' Y2K LPAR. Obviously if your site has not adopted this strategy, the utility should still be easily modifiable.

- The inclusion of the SYSDA volumes in the dialog is to allow copying of non-SMS managed datasets. SMS-managed datasets will ignore the volume on the restore step. You will need to adjust the volsers assigned to the SYSDA variables to your site's values.

This dialog was written and developed under OS/390 1.3.0, DFSMS 1.3, TSO 2.5, and ISPF 4.4

Y2KCOPY

```
/*REXX
   Member    : Y2KCOPY
   Function  : Generates DFDSS jobs to copy datasets to Y2K test system
   Called by : TSO Y2KCOPY
   Panels    : Y2KCOPY Y2KCOPYH
   Skeletons : Y2KCPYS1 Y2KCPYS2 Y2KCPYS3
*/
arg y2kparm
address ispexec   /* default host commands are ISPF             */
call init         /* set initial variables and check if running */
/*
  Main Display Loop. Display main panel using TBDISPLAY to allow
scrollable list of previously entered rules.
*/
start:
“TBDISPL Y2KCOPY PANEL(Y2KCOPY) MSG(“msg”) CURSOR(“cur”) MSGLOC(“cur”)”
msg = '';ur = ''
if rc ¬= Ø then signal fin    /* PF3/15 pressed                */
first = 'NO'           /* reset panel verification control      */
if rule ¬= '' then     /* filter rule entered?                  */
do
  rules = rule         /* set table variable                    */
  call spacecalc       /* subroutine to estimate backup size    */
  select
    when result = 'MIG' then
    do
      zedlmsg = 'Dataset is migrated - cannot be copied'
      msg = 'ISRZØØ1'
      cur = 'RULE'
      signal start
    end
    when result = 'LMD' then
    do
      zedlmsg = 'Invalid dataset name or filter'
      msg = 'ISRZØØ1'
      cur = 'RULE’
      signal start
    end
    otherwise NOP
```

```
  end
  “TBADD Y2KCOPY”      /* add to table for display */
  signal start         /* loop round for next one  */
end
/*
  Here if blank rule entered.

  First allocate temporary back-up dataset on shared volume
*/
/*
  totspace = total space required for all rules in tracks

  convert to cylinders and round up
*/
totcyl = (totspace % 15) + 1
/*
  if greater then 50 cylinders, set variable to include 'COMPRESS'
  option in DFDSS, else set off as trying to compress anything smaller
  wastes CPU cycles.
*/
if totcyl > 50 then compress = 'YES’
else compress = 'NO’
/*
  use half total value (rounded up) for primary and 1/10th
  (rounded up) for secondary space allocation.
   No particular reason for these values - I just liked them !
*/
prim = (totcyl % 2) + 1
sec  = (totcyl % 10) + 1
address tso "ALLOC FI(Y2KTEMP) DA("tdsn") NEW CATALOG",
"UNIT(3390) VOL("tvol") CYLINDERS SPACE("prim sec")",
"DSORG(PS) RECFM(F) REUSE"
if rc ¬= 0 then
do
  zedsmsg = 'ALLOCATION FAILURE'
  zedlmsg = "UNABLE TO ALLOCATE '"tdsn"'. POSSIBLY LACK OF SPACE."
  "SETMSG MSG(ISRZ001)"
  signal fin
end
address tso “FREE FI(Y2KTEMP)"
/*
  Now allocate 'lock' member in users ISPF Profile dataset
  and write status message line in
*/
address tso "ALLOC FI(Y2KLOCK) DA("lockdsn") SHR REUSE"
lock1 = 'JOB1 SUBMITTED - AWAITING EXECUTION'
address mvs "EXECIO 1 DISKW Y2KLOCK (STEM LOCK FINIS"
address tso "FREE FI(Y2KLOCK)"
/*
  Now generate batch JCL from skeleton and submit
*/
"FTOPEN TEMP"
```

```
"FTINCL Y2KCPYS1"
"FTCLOSE"
"VGET (ZTEMPF) ASIS"
/*
  use the following statement for debuging purposes as it allows
  you to pre-edit the generated JCL deck prior to submission

 “EDIT DATASET("ztempf")"
*/
call outtrap 'LINE.' /* trap submit messages to get jobname/number*/
address tso "SUBMIT '"ztempf"'"
call outtrap 'OFF'
do cnt = 1 to line.Ø
  if word(line.cnt,2) = 'JOB' then
  parse var line.cnt . . jobname .
end
/*
  Set informational message
*/
zedlmsg = 'Copy job 'jobname' submitted'
"SETMSG MSG(ISRZØØ1)"
fin:
/*
  Ditch temporary table and quit
*/
"TBEND Y2KCOPY"
exit
/*
  Initialization.

  set up various variables
  and check if Y2KCOPY already running.
*/
init:
msg = ''                /* message id for TBDISPL             */
cur = ''                /* cursor position for TBDISPL        */
first = 'YES'           /* indicator for panel verification   */
totspace = Ø            /* space allocation for temp dataset  */
/*
  Name strings for each destination LPAR - used in JCL comments
*/
name.MVSA = 'Company A Production'
name.MVSB = 'Company B Production'
name.MVSC = 'Company A Year2ØØØ test'
name.MVSD = 'Company B Year2ØØØ test'
/*
  SYSDA voumes for each LPAR - used in OUTDYNAM in case of non-SMS d/set
*/
sysda.MVSA = 'SYSAØ1'
sysda.MVSB = 'SYSBØ1'
sysda.MVSC = 'SYSCØ1'
sysda.MVSD = 'SYSDØ1'
```

```
/*
  Dsname prefix (HLQ) for temporary dataset for each partition
  We had to do this as the different systems have different dataset
  naming conventions.
*/
tdsnpref.MVSA = userid()'.D.NSMS'
tdsnpref.MVSB = 'SYS7.'userid()
tdsnpref.MVSC = userid()'.D.NSMS'
tdsnpref.MVSD = 'SYS7.'userid()
/*
  JES NJE node names for each LPAR - used in XEQ cards in JCL
*/
node.MVSA = 'JESANJE'
node.MVSB = 'JESBNJE'
node.MVSC = 'JESAY2K'
node.MVSD = 'JESBY2K'
/*
  'partner' nodes for each LPAR - used to set destination node
*/
partner.MVSA = 'MVSC'
partner.MVSB = 'MVSD'
partner.MVSC = 'MVSA'
partner.MVSD = 'MVSB'
/*
  extract current LPAR id from ECVT
*/
cvt = c2x(storage(1Ø,4))
ecvt = c2x(storage(d2x(x2d(cvt)+14Ø),4))
sysid = storage(d2x(x2d(ecvt)+344),4)
/*
  now set variables
*/
lparid = partner.sysid   /* destination LPAR                */
lparname = name.lparid   /* destination name                */
homenode = node.sysid    /* current JES2 NJE node           */
destnode = node.lparid   /* destination JES2 NJE node       */
ody = sysda.lparid       /* output volume                   */
lockdsn = userid()'.ISPF.ISPPROF(Y2KCPYLK)' /* 'lock' member */
tdsn=tdsnpref.sysid'.Y2K'sysid  /* back-up dataset          */
tvol = 'Y2KSHR'                 /* shared DASD volume       */
/*
  invoke subroutine function to test for lock.
  function returns 'true' (1) or 'false' (Ø)
*/
if check_active() = true then exit
/*
  set up temporary table for displaying existing rules on panel
*/
"TBCREATE Y2KCOPY NOWRITE REPLACE NAMES(RULES)"
return
/*
  check if Y2KCOPY already running by:-
```

```
  -  Check if Y2KCPYLK member exists in user's ISPF profile dataset.
  -  If it does, read message line from it and display
  -  If user passed 'UNLOCK' as argument, give option to
     manually unlock. This is used after unrecoverable job failures
     such as job cancelled by operator.
*/
check_active:
/*
  set true/false variables for readability
*/
true=1
false=Ø
if sysdsn(lockdsn) = 'OK' then   /* lock exists? */
/*
  Here if lock file exists.
  Read status message from lock file.
  If Status message is job failure (indicated by '*' as first character)
  then display status panel with instructions, and provide unlock
  option.
  Otherwise, set status message as ISPF message and return true (1)
*/
  do
    address tso "ALLOC FI(Y2KLOCK) DA("lockdsn") SHR REUSE"
    address mvs "EXECIO 1 DISKR Y2KLOCK (STEM LINE FINIS"
    address tso "FREE FI(Y2KLOCK)"
    If left(line1,1) = '*' then    /* error message?                  */
      return unlock()  /* yes - let unlock function set return option */
    Else
    Do
      If y2kparm = 'UNLOCK' then   /* manual unlock requested?        */
      Do
        line1 = '*Manual override' /* set dummy message               */
        return unlock() /* and let unlock function set return option  */
      End
      zedlmsg = line1
      "SETMSG MSG(ISRZØØ1)"        /* set as ISPF message and quit    */
      return true
    End
  end
/*
  obviously not locked, so return false (Ø)
  to continue
*/
return false
/*
  unlock routine.
  Here if lock member contains error message.
  Error message identified by '*' in column 1.
  Display panel containing error message.

  If user specifies 'YES' to unlock then return false (Ø) as
  result (will be propagated as check_active result to allow
  routine to continue)
```

```
  lock member will also be deleted.

  Else return true (1) to exit Y2KCOPY
*/
unlock:
/*
Set error message (less '*') as panel variable
*/
  msg1 = strip(line1,'L','*')
/*
  display panel, check if unlock specified and call PDSMAN to
  delete lock file.
*/
"DISPLAY PANEL(Y2KCOPY1)"
If rc \= Ø then signal goback  /* pf3/15 pressed - stay locked    */
if unlock = 'YES' then
do
  call outtrap 'LINE'
  address tso "DELETE '"userid()".ISPF.ISPPROF'"
  if sysdsn(tdsn) = 'OK' then
    address tso "DELETE '"tdsn"'"
  call outtrap 'OFF'
  zedlmsg = 'Y2KCOPY Unlocked'
  "SETMSG MSG(ISRZØØ1)"        /* set as ISPF message and continue */
  return false
end
goback:
zedlmsg = 'Y2KCOPY Locked'
"SETMSG MSG(ISRZØØ1)"          /* set as ISPF message and continue */
return true
spacecalc:
/*
  Space calculation subroutine.
  Uses ISPF LMDLIST service to list out all datasets matching the
  rule. The space used in tracks is then added up and added to the
  total space used.
  This value will be used when allocating the temporary back-up dataset
*/
/*
  first, check to see if rule is single dataset
*/
call msg 'OFF'
onedsn = false
chek = listdsi(rule 'NORECALL')
if chek = Ø then onedsn = true  /* rule is single dataset        */
if chek = 16 & sysreason = 1 then onedsn=false /* is filter      */
if chek = 16 & sysreason = 9 then return 'MIG'  /* dataset migrated
                         and so  will not be backed up by DFDSS  */
if chek = 16 & sysreason = 5 then return 'LMD'
call msg 'on'
"CONTROL ERRORS RETURN"
"LMDINIT LISTID(LID) LEVEL("rule")" /* initialize LMD processing */
```

```
/*
  Use LMDLIST to generate dataset list containing allocation info
  into a dataset called userid.TEMP.DATASETS
*/
"LMDLIST LISTID("lid") OPTION(SAVE) STATS(YES) GROUP(TEMP)"
lmdrc = rc
"LMDLIST LISTID("lid") OPTION(FREE)"
"LMDFREE LISTID("lid")"
zedlmsg = ''            /* set empty message                          */
“SETMSG MSG(ISRZØØ1)"   /* and display to hide LMDLIST save message */
if lmdrc ¬= Ø then return 'LMD'        /* no datasets matched rule  */
address tso "ALLOC FI(LMDLIST) DA("userid()".DFTS.DATASETS) OLD REUSE"
address tso "EXECIO * DISKR LMDLIST (STEM LMD. FINIS"
address tso "FREE FI(LMDLIST) DELETE"
if onedsn then oneflag = Ø
do cnt = 1 to lmd.Ø
/*
 if rule is single database, check first word on line. If it matches
 then this is the line for that dataset, else skip.
 This is because LMDLIST can match more than one dataset, even for
 a single dataset rule, if there are other datasets with
 matching qualifers. eg USER.TEST.LIB would also match
 USER.TEST.LIB.NEW, etc.
*/
 if onedsn & word(lmd.cnt,1) ¬= rule then iterate
 if onedsn then oneflag = 1 /* single dataset matched] */
 parse var lmd.cnt . 75 trk 82 .
 trk = strip(trk)
 if trk ¬= '' then
    totspace = totspace + trk
end
if onedsn and oneflag = Ø then return 'LMD'
return
```

PANEL Y2KCOPY

```
)ATTR DEFAULT(%+_)
       %   TYPE(TEXT) INTENS(HIGH)
       +   TYPE(TEXT) INTENS(LOW) SKIP(ON)
       _   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
       $   TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY EXPAND(##)
%-#-#- Year2ØØØ Cross-Partition Dataset Copy Utility -#-#-
%Command ===>_zcmd   # #  %Scroll ==>_amt +
+
       Enter following details:

      +Include Filter%:_rule                                          +

% Press+ENTER%to add filter rule. Null (blank) rule submits job.
%-#-#- Active Rules: -#-#-
```

```
)MODEL
+    $RULES
)INIT
.CURSOR = RULE
IF (.MSG = &Z)
   &RULE = &Z
.HELP=Y2KCOPYH
)PROC
IF (&FIRST = 'YES')
   VER (&RULE,NB)
)END
```

PANEL Y2KCOPY1

```
)ATTR DEFAULT(%+_)
       %   TYPE(TEXT) INTENS(HIGH)
       +   TYPE(TEXT) INTENS(LOW) SKIP(ON)
       _   TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
       $   TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT)
)BODY EXPAND(##)
%-#-#- Year2000 Cross-partition dataset copy utility -#-#-
%command ===>_zcmd   # #  %scroll ==>_amt +
+
+ # #  Y2KCOPY has failed:                                    # #
+
+ # # $msg1                                                   # #
+
+ # #  Check the output from the failing job to try and       # #
+ # #  determine the cause of the failure. You may need to    # #
+ # #  contact Technical Support for assistance.              # #
+
+ # #  Once the error is corrected, you may reply%YES+to the  # #
+ # #  prompt below to proceed.                               # #
+
+ # #  If the error is%MANUAL OVERRIDE+then it is assumed     # #
+ # #  that you have used the manual unlock option under      # #
+ # #  instructions from IT Technical Support. If so, you may # #
+ # #  reply%YES+to proceed.                                  # #
+
% # #  Unlock (yes/no)+==>_z  + # #
+
+
)INIT
.ZVARS = '(UNLOCK)'
.CURSOR = UNLOCK
&UNLOCK = 'NO'
.HELP=Y2KCOPYH
)PROC
VER (&UNLOCK,NB,LIST,YES,NO)
)END
```

PANEL Y2KCOPYH

```
)ATTR DEFAULT(@+_)
 ' TYPE(PT)                          /* panel title line            */
 ? TYPE(PIN)                         /* panel instruction line      */
 # TYPE(NT)                          /* normal text attribute       */
 } TYPE(ET)                          /* emphasized text attribute   */
 ] TYPE(DT)                          /* description text            */
 ~ AREA(SCRL)                        /* scrollable area attribute   */
)BODY EXPAND([[)
'-[-[- Year2000 Cross-partition dataset copy utility -[-[-
#
}This panel provides the user with the ability to transfer data to the
}&lparname partition (&lparid).
}
}==============================================================================
~pnarea                                                                       ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
~                                                                             ~
}==============================================================================
#
?ENTER = Scroll forward. LEFT/RIGHT = Scroll Up/Down
)AREA pnarea
#
}HOW IT WORKS
#
 This dialog generates a series of batch jobs to back-up (copy)
#selected datasets from &sysid and then restore them onto the
#&lparid partition.
 The batch jobs utilize an IBM program called DFDSS (the program
 name in the jobs is actually ADRDSSU), which is a general-purpose
 data copying and moving facility.
#
 The three jobnames are:-
          }&JOBNAM1# ..... Back-up selected datasets on &SYSID
          }&JOBNAM2# ..... Restore selected datasets on &LPARID
          }&JOBNAM3# ..... 'unlock' Y2KCOPY for next run.
#                                  (runs on &SYSID)
#
}HOW TO ENTER YOUR DATA
#
```

```
 The}INCLUDE FILTER#field is where you enter the required dataset
 name selection rules.
 DFDSS supports full generic filtering when specifying datasets. This
 allows you to request a group of datasets which share common qualifiers
 with a single statement. The format is identical to that used by
 ISPF dataset list dialogs such as EZYEDIT or 1.3.4.
#
 Y2KCOPY allows multiple rules to be entered. A scrollable list of the
 rules is displayed at the bottom of the screen.
#
 When all rules have been entered, a null reply (ie pressing enter
 with a blank rule in the}INCLUDE FILTER#field) will start the process.
#
 Examples of valid rules are:

       }DSH.DKSS.TEST.VSAM.FILE#  - example of a single dataset filter
        }&ZUSER..** #             - all datasets for  &ZUSER
       }&ZUSER..**.NEW#           - all &ZUSER dataset data end in NEW
       }&ZUSER..DPSS.%%%VSAM.FILE#- all &ZUSER..DPSS.**.FILE data
setsMVSB                              with a 7-character third-level
qualifer
                                   ending in 'VSAM'.
}'LOCKOUT' FEATURE AND JOB FAILURES
#
 Whilst a Y2KCOPY run is executing, you will be 'locked-out' of
 the dialog until the run completes (ie job3 has finished).

 If you try and access Y2KCOPY during the run, you will be given
 a status message detailing the current status (eg JOB1
 executing, JOB2 submitted, etc).
#
 Should any job fail, then Y2KCOPY will attempt to clean up
 any temporary files, but will leave the dialog locked-out.
 When you access Y2KCOPY you will be presented with a
 panel detailing which job failed.
 Check the job output from the failing run to try and determine the
 cause of the failure. You may need to contact technical support
 for assistance.
 Once you have determined the reason, you can reply 'YES' to the
 error panel prompt to re-enable the Y2KCOPY dialog and try
 the run again.
#
 If you receive any other types of errors, or dialog failures, then
 contact technical support for assistance.
)INIT
&JOBNAM1 = '&ZUSER.Y2K1'
&JOBNAM2 = '&ZUSER.Y2K2'
&JOBNAM3 = '&ZUSER.Y2K3'
IF (&SYSID = 'GLP2' ! &SYSID = 'MVSD')
  &JOBNAM1 = '&ZUSER.1'
  &JOBNAM2 = '&ZUSER.2'
```

```
  &JOBNAM3 = '&ZUSER.3'
)PROC
&ZUP=Y2KCOPYH
&ZTOP=Y2KCOPYH
&ZCONT=Y2KCOPYH
)END
```

SKELETON Y2KCPYS1

```
)DEFAULT  )@>£<!>
)SEL @SYSID = MVSA ! @SYSID = MVSC
//@ZUSER.Y2K1 JOB   Y2K,
//         'Y2K COPY JOB1',
//         CLASS=S,
//         MSGLEVEL=(1,1),
//         MSGCLASS=T,
//         REGION=8M
)ENDSEL
)SEL @SYSID = MVSB ! @SYSID = MVSD
//@ZUSER.1 JOB   Y2K,
//         'Y2K COPY JOB',
//         CLASS=G,
//         MSGLEVEL=(1,1),
//         MSGCLASS=A,
//         REGION=8M
)ENDSEL
//*  Y2KCOPY1 - DFDSS COPY DATASET(S) TO @LPARNAME
//*
//*  GENERATED BY   : @ZUSER
//*  GENERATED DATE : @ZDATE
//*  GENERATED TIME : @ZTIME
//*
//GENER    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *
JOB1 EXECUTING
/*
//SYSUT2   DD DSN=@ZUSER..ISPF.ISPPROF(Y2KCPYLK),DISP=SHR
//SYSIN    DD DUMMY
//LIBCPY   EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//TAPE1    DD   DUMMY
//TAPE2    DD DSN=@TDSN,DISP=(OLD,KEEP,KEEP)
//SYSIN    DD   *
 DUMP -
 CANCELERROR -
 DATASET(INCLUDE( -
)DOT Y2KCOPY
         @RULES, -
```

```
)ENDDOT
        )) -
)SEL @COMPRESS EQ YES
         COMPRESS -
)ENDSEL
         NOVALID -
         SPHERE -
        OPTIMIZE(4) -
         OUTDDNAME(TAPE2) WAIT(Ø,Ø) TOL(ENQF)
/*
// IF (RC GT 4) THEN
//DSSERROR EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//DELTEMP  DD DSN=@TDSN,DISP=(OLD,DELETE,DELETE)
//SYSUT1   DD *
*JOB1 FAILED WITH DFDSS FAILURE. CHECK OUTPUT.
/*
//NOTIFY   EXEC PGM=IKJEFTØ1
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
SEND 'Y2KCOPY JOB1 HAS FAILED!' USER(@ZUSER) LOGON
/*
//SYSUT2   DD DSN=@ZUSER..ISPF.ISPPROF(Y2KCPYLK),DISP=SHR
//SYSIN    DD DUMMY
// ELSE
//GENER1   EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=*
//SYSUT1   DD    DATA,DLM=$$
)IM Y2KCPYS2
$$
//SYSUT2   DD    SYSOUT=(,INTRDR)
//SYSIN    DD DUMMY
//GENER2   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD *
JOB1 COMPLETED - JOB2 SUBMITTED TO @LPARID
/*
//SYSUT2   DD DSN=@ZUSER..ISPF.ISPPROF(Y2KCPYLK),DISP=SHR
//SYSIN    DD DUMMY
// ENDIF
//
```

SKELETON Y2KCPYS2

```
)DEFAULT  )@?]<!>
)SEL @SYSID = MVSA ! @SYSID = MVSC
//@ZUSER.Y2K2 JOB   Y2K,
//         'Y2K COPY JOB2',
//         CLASS=S,
//         MSGLEVEL=(1,1),
//         MSGCLASS=T,
```

```
//         REGION=8M
)ENDSEL
)SEL @SYSID = MVSB ! @SYSID = MVSD
//@ZUSER.2 JOB   Y2K,
//         'Y2K COPY JOB2',
//         CLASS=G,
//         MSGLEVEL=(1,1),
//         MSGCLASS=A,
//         REGION=8M
)ENDSEL
/*XEQ @DESTNODE
//*  Y2KCOPY2 - DFDSS RESTORE DATASET(S) ON @LPARNAME
//*
//*  GENERATED BY   : @ZUSER
//*  GENERATED DATE : @ZDATE
//*  GENERATED TIME : @ZTIME
//*
//LIBCPY   EXEC PGM=ADRDSSU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//TAPE1    DD DSN=@TDSN,
//         DISP=OLD,
//         UNIT=339Ø,VOL=SER=@TVOL
//SYSIN    DD   *
 RESTORE -
 DATASET(INCLUDE(**.**)) -
 INDDNAME(TAPE1) -
 TGTGDS(SOURCE) -
 OUTDY(@ODY) -
 REPLACE -
 SPHERE CATALOG
/*
// IF (RC GT 4) THEN
//GENER    EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=*
//SYSUT1   DD    DATA,DLM='££'
)SET FAIL = YES
)IM Y2KCPYS3
££
//SYSUT2   DD    SYSOUT=(,INTRDR)
//SYSIN    DD DUMMY
// ELSE
//GENER    EXEC  PGM=IEBGENER
//SYSPRINT DD    SYSOUT=*
//SYSUT1   DD    DATA,DLM='££'
)SET FAIL = NO
)IM Y2KCPYS3
££
//SYSUT2   DD    SYSOUT=(,INTRDR)
//SYSIN    DD DUMMY
//  ENDIF
```

SKELETON Y2KCPYS3

```
)DEFAULT  )@?£<!>
)SEL @SYSID = MVSA ! @SYSID = MVSC
//@ZUSER.Y2K3 JOB   Y2K,
//         'Y2K COPY JOB',
//         CLASS=S,
//         MSGLEVEL=(1,1),
//         MSGCLASS=T,
//         REGION=8M
)ENDSEL
)SEL @SYSID = MVSB ! @SYSID = MVSD
//@ZUSER.3 JOB   Y2K,
//         'Y2K COPY JOB3',
//         CLASS=G,
//         MSGLEVEL=(1,1),
//         MSGCLASS=A,
//         REGION=8M
)ENDSEL
/*XEQ @HOMENODE
//*  Y2KCOPY3 - UNLOCK Y2KCOPY FOR NEXT RUN OR UPDATE STATUS IF FAILED
//*
//*  GENERATED BY   : @ZUSER
//*  GENERATED DATE : @ZDATE
//*  GENERATED TIME : @ZTIME
//*
)SEL @FAIL EQ YES
//DSSERROR EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//DELTEMP  DD DSN=@TDSN,DISP=(OLD,DELETE,DELETE)
//SYSUT1   DD *
*JOB2 FAILED WITH DFDSS FAILURE. CHECK OUTPUT.
/*
//SYSUT2   DD DSN=@ZUSER..ISPF.ISPPROF(Y2KCPYLK),DISP=SHR
//SYSIN    DD DUMMY
//NOTIFY   EXEC PGM=IKJEFTØ1
//SYSTSPRT DD SYSOUT=A
//SYSTSIN  DD *
SEND 'Y2KCOPY JOB2 HAS FAILED]' USER(@ZUSER) LOGON
/*
)ENDSEL
)SEL @FAIL EQ NO
//UNLO
```

The VLF API

INTRODUCTION

The Virtual Lookaside Facility (VLF) can create and manage data spaces for applications and therefore is sometimes referred to as a data space manager. It has been provided by IBM so that an application, IBM or user defined, can store and retrieve named objects from virtual storage rather than perform repetitive I/O operations from DASD. VLF uses data spaces to store the frequently used objects. Internally, VLF manages classes, each class having one or more major names associated with it, and each major name having one or more minor names associated with it. There are two data spaces created for each class name. Each name comprises a prefix, followed by the class name. One of the following values is the prefix:

- C contains the control blocks for the class
- D contains the user objects for the class.

The MVS components LLA, TSO/E, RACF, and CAS utilize the services available within VLF as an alternate method to access data.

The following MVS services have been updated to call VLF. These calls occur when a PDS member has been modified by a:

- MVS allocation
- CLOSE request
- RENAME request
- STOW request
- SCRATCH request
- DFDSS functions

This article considers the VLF API services that have been provided by IBM to enable applications to interface with VLF.

STARTING VLF

VLF runs as a non-swappable address space that is started at IPL time. Parameters in member COFVLFxx in SYS1.PARMLIB define classes that can be managed by VLF. To start VLF the following command must be issued:

```
S VLF,SUB=MSTR,N=xx
```

VLF must be started under the Master Subsystem. The two alphanumeric characters *xx* are appended to COFVLF to form the name of the COFVLFxx PARMLIB member.

VLF MACROS

VLF provides seven macro calls. Their functions are as follows:

- COFDEFIN – define a class of VLF objects
- COFIDENT – identify an end user to VLF
- COFRETRI – retrieve a VLF object
- COFCREAT – create a VLF object
- COFNOTIF – notify VLF of a change
- COFREMOV – remove a VLF end user
- COFPURGE – purge a VLF user.

VLF CLASS

A group of related data objects must be defined to VLF as a class. To activate a class, a class statement must be defined in the current COFVLFxx PARMLIB member, and the application must issue the COFDEFIN macro. The COFDEFIN macro defines a class of Virtual Lookaside Facility objects. The classname may be one to seven alphanumeric characters long. When you define a VLF class, the system creates a VLF data space that will maintain related control blocks. The name of the data space is formed by prefixing a character C before the class name. With LLA the following data space is created:

```
CCSVLLA
```

The system obtains the attributes of the class from the input parameters

of the macro and the description of the class in the active COFVLFxx PARMLIB member. An optional parameter of the COFDEFIN macro, TRIM, specifies how you want VLF to manage virtual storage for the objects in the class:

- TRIM=ON – VLF automatically removes the least recently used objects when it needs space.
- TRIM=OFF – objects are removed only when VLF is specifically notified.

The parameter AUTHRET=YES enables only authorized code to retrieve objects.

An important consideration when defining applications that use the services of VLF is that the application must be designed to continue when VLF is not active. All VLF API services issue a return code X'28' when VLF is not active. VLF provides two different implementations of its services, and this is dependent on what class the object belongs to. A class can consist of:

- OBJECTS belonging to a PDS class
- OBJECTS belonging to a non-PDS class.

PDS classes consist of objects that correspond to members of partitioned datasets, and non-PDS do not correspond to partitioned data sets, they are a named collection of data. The type of class VLF is to deal with is specified in the current COFVLFxx PARMLIB member. PDS classes are specified using the EDSN and VOL(optional) parameters, and non-PDS classes are specified by using the EMAJ parameter. The following definition defines a PDS class:

```
CLASS NAME(VLFMAJ)
      EDSN(SYS1.VLF.DATASET)
```

The following definition defines a non-PDS class:

```
CLASS NAME(VLFMAJ)
      EMAJ(VLFMAJORNAME)
```

MAJOR AND MINOR NAMES

In both the PDS and non-PDS classes, VLF uses two levels of name to identify an object, as follows:

- Major name
- Minor name

The major name for a PDS class consists of a 6-character volume serial number concatenated to a 44-character dataset name. The major name for a non-PDS class is user defined and can be between 1 and 64 characters long. In addition to the major name, VLF also needs a minor name to identify a unique data object within the major name. The minor names are specified by using the COFCREAT macro. The minor name of a VLF object belonging to a PDS class is the PDS member name. For a NON-PDS class, this is left to the application developer. For a PDS class the length is always 8. For a non-PDS class the length can be between 1 and 64 characters (the length is specified in the COFDEFIN macro). Within a major name, each minor name must be unique; however, the same minor name can exist across more than one major name.

IDENTIFYING A VLF USER

Individual users who require access to a particular class of VLF objects must be defined using the COFIDENT macro. A user token (UTOKEN) is returned that uniquely identifies a user with access to a particular class of objects. The user token must be provided on requests that create or retrieve VLF objects on behalf of the user. An important part of this macro is the major name search order. For a PDS class the DDNAME parameter is used, for a non-PDS class the MAJNLIST parameter is used. For a non-PDS class, each entry in the list must match a major name defined for this class through the EMAJ parameter in the active COFVLFxx PARMLIB member. For a PDS class the DDNAME parameter specifies the DDNAME of a concatenated dataset list. The DDNAME that VLF is to use to locate the objects is normally supplied by the user.

An important specification on the COFIDENT macro is the SCOPE parameter. This parameter determines which tasks can retrieve objects with the user token (UTOKEN) as follows:

- SCOPE=HOME – only tasks under the same home address space (ASID) as the task that issued the COFIDENT macro can retrieve objects.

- SCOPE=SYSTEM – tasks running in any address space can retrieve objects. This specification allows subsystems running under a particular home address space to control a set of VLF objects, and allows all tasks in the system to access these objects.

The value that is specified for the SCOPE affects only the retrieving of objects with the COFRETRI macro. All other macros that supply the user token must have the same home ASID as the user of COFIDENT.

CREATING A VLF OBJECT

The COFCREAT macro is used to create an object to a class of VLF objects. Issuing the COFCREAT requires the following parameters:

- The MAJOR name for the object.
- For a PDS class the major name is indirectly specified through the concatenation index on the CINDEX parameter. The CINDEX value is obtained by issuing a BLDL macro (BLDL 'K' value).

For a non-PDS class, the major name is specified in the MAJOR parameter.

- The minor name of the object
- The user token (UTOKEN)
- Source and size of the object.

IBM recommends the following code logic:

- The COFRETRI macro is issued before the COFCREAT macro in an attempt to retrieve the object. This is to ensure that VLF does not create an object if the permanent source data changes between the time the object is obtained and the time the object is created.
- To ensure the integrity of the data that is used to create the VLF object, the storage key of the data must not be key 8.

For a non-PDS class the parameter REPLACE can be specified. If REPLACE is specified, VLF does not require that COFRETRI precede COFCREAT. If REPLACE is specified and the object already exists in VLF storage, VLF replaces the existing object. If COFCREAT is specified without REPLACE for an object that already exists in

VLF storage, VLF returns a successful completion code but does not replace the object (VLF assumes that the objects are identical).

The objects are stored in a data space that is named by VLF by prefixing a character D before the class name. In the case of LLA the following data space is created:

```
DCSVLLA
```

RETRIEVING A VLF OBJECT

The COFRETRI macro obtains a copy of a VLF object. The COFIDENT macro must be issued before the COFRETRI macro to identify the user. A return code of 8 from COFRETRI specifies that the object was not found. COFCREAT can then be issued to create the object in VLF storage.

NOTIFYING VLF OF A CHANGE

The COFNOTIF macro notifies VLF that certain VLF objects are no longer valid. The system automatically informs VLF of all PDS updates when VLF is running on a system that is part of a Sysplex and the changed data belongs to a PDS class (VLF receives notification through Sysplex services). In non-Sysplex systems or non-PDS data, notification to VLF is not automatic (IBM provides the VLFNOTE TSO command as a user interface for this purpose). The type of notify change is specified on the FUNC parameter. The options include:

- DELMAJOR – specifies one or more major names to be deleted.
- DELMINOR – specifies one or more minor names to be deleted.
- ADDMINOR – adds one or more minor names to a major names.
- UPDMINOR – specifies that one or more objects corresponding to existing minor names have been changed.
- PURGEVOL – specifies that a physical storage device has been logically disconnected from the system.

An important point to note for a PDS class is that VLF views a minor name with one or more alias names as separate objects. Thus to change a minor name, the COFNOTIF macro must be issued for the minor name and for each alias name.

REMOVING A VLF END USER

The COFREMOV macro terminates an end user's access to the class of VLF objects associated with the specified user token (UTOKEN). After the user is removed, VLF rejects an unknown UTOKEN. The COFREMOV macro must be issued from a task which issued the COFIDENT to identify the user.

PURGE A VLF USER

The COFPURGE macro requests that VLF deletes a class of VLF objects. The class is deleted by VLF automatically, so that any user requesting this class will fail. The class can be reinstated by issuing the COFDEFIN macro. Once the class has been reinstated, users of the class can be identified by issuing the COFIDENT macro.

TSO/E AND VLF

TSO/E from Version 2.1 has used VLF to save and retrieve CLISTs and REXX EXECs invoked by TSO users. TSO/E communicates with VLF using a PDS-type class, IKJEXEC. Libraries containing CLISTs and REXX EXECs that VLF is to manage must be defined using the EDSN parameter in COFVLFXX.

When a TSO user executes a CLIST or REXX EXEC, TSO routines determine whether a UTOKEN already exists for the user in question. If a UTOKEN does not exist, the COFIDENT macro is issued on behalf of the user. As part of the COFIDENT processing, VLF is passed the names of the datasets in the TSO user's SYSPROC concatenation. This enables VLF to record the SYSPROC search order in its control blocks.

When a TSO user executes a CLIST, the CLIST processor issues a COFCREAT macro to try to obtain the CLIST from VLF. If the CLIST is found in the DIKJEXEC data space, the CLIST is moved to the user's address space and the CLIST processor accesses it directly. If the CLIST is not found within the data space, the CLIST processor must read it from DASD. It then processes the CLIST through Phase 1 processing, which results in a tokenized CLIST. The COFCREAT macro is then issued to add the tokenized CLIST to the data space. This will enable future requests for the CLIST to bypass DASD fetch, and Phase 1 processing.

VLF APPLICATION EXAMPLE

The VLF application contained in this article uses the services of VLF to create and retrieve non-PDS classes. The following VLF class definition must be added to the active COFVLFxx member:

```
CLASS NAME(VLFMESS)
      EMAJ(VMESSAGE)
```

VLF will create the following data spaces:

```
CVLFMESS
DVLFMESS
```

The logic of the program is as follows:

- The COFDEFIN macro is issued to define the CLASS, VLFMESS. The minor name length is set to 8 bytes. TRIM is specified as ON.
- The COFIDENT macro is issued to identify a VLF user for class VLFMESS. A 16-byte user token is returned to identify the user. The scope of the request is set to SYSTEM.
- The COFRETRI macro is issued to obtain a copy of the VLF object VMESS001(minor name). If the object does not exist the COFCREAT macro is issued to create the object and the macro COFRETRI is reissued. The first time, the object VMESS001 will not exist.
- The COFNOTIF macro is issued to notify VLF that the object VMESS001 has been changed.
- Update the VLF object VMESS001.
- The COFCREAT macro is issued to create the updated object, and the COFRETRI macro is issued to return the updated object. This process could also have been achieved by issuing the COFCREAT macro with the replace option. In this case the COFNOTIF macro does not need to be issued.
- The COFREMOV macro is issued to terminate the user's access to the class of VLF objects associated with the user token.
- The COFPURGE macro is issued to purge the VLF class VLFMESS.

The sample program must be link-edited as RENT and AC=1, and stored into an authorized library.

VLFAPPL

```
TITLE 'SAMPLE VLF PROGRAM'
*-------------------------------------------------------------------*
* THIS SAMPLE PROGRAM USES THE SERVICES OF VLF TO CREATE AND        *
* RETRIEVE A NON-PDS CLASS OBJECT. VLF PROVIDES SEVEN BASIC         *
* FUNCTIONS. THE FUNCTIONS AND THE CORRESPONDING MACROS ARE:        *
*                                                                   *
* COFDEFIN      -       DEFINE A CLASS OF VLF OBJECTS               *
* COFIDENT      -       IDENTIFY AN END USER TO VLF                 *
* COFRETRI      -       RETRIEVE A VLF OBJECT                       *
* COFCREAT      -       CREATE A A VLF OBJECT                       *
* COFNOTIF      -       NOTIFY VLF OF A CHANGE TO AN OBJECT         *
* COFREMOV      -       REMOVE A VLF END USER                       *
* COFPURGE      -       PURGE A VLF CLASS                           *
*                                                                   *
* THIS SAMPLE PROGRAM UTILIZES ALL THE SERVICES OF VLF.             *
*                                                                   *
* THE FOLLOWING STATEMENTS MUST BE PUT INTO THE CURRENT COFVLFXX    *
* PARMLIB MEMBER:                                                   *
*                                                                   *
* CLASS NAME(VLFMESS)          /* CLASS NAME FOR VLFMESS SYSTEM */  *
*       EMAJ(VMESSAGE)         /* MAJOR NAME FOR VLFMESS SYSTEM */  *
*                                                                   *
* ABEND CODES:                                                      *
*                                                                   *
* 901           -       COFDEFIN ERROR                              *
* 902           -       COFIDENT ERROR                              *
* 903/906       -       COFRETRI ERROR                              *
* 904/907       -       COFCREAT ERROR                              *
* 905           -       COFNOTIF ERROR                              *
* 908           -       COFREMOV ERROR                              *
* 909           -       COFPURGE ERROR                              *
*-------------------------------------------------------------------*
* CHANGES:                                                          *
*-------------------------------------------------------------------*
VLFAPPL   CSECT                                                     *
         TITLE 'EQUATES'
R0       EQU   0                       REGISTER 0
R1       EQU   1                       REGISTER 1
R2       EQU   2                       REGISTER 2
R3       EQU   3                       REGISTER 3
R4       EQU   4                       REGISTER 4
R5       EQU   5                       REGISTER 5
R6       EQU   6                       REGISTER 6
R7       EQU   7                       REGISTER 7
R8       EQU   8                       REGISTER 8
R9       EQU   9                       REGISTER 9
R10      EQU   10                      REGISTER 10
R11      EQU   11                      REGISTER 11
R12      EQU   12                      REGISTER 12
R13      EQU   13                      REGISTER 13
R14      EQU   14                      REGISTER 14
```

```
R15      EQU   15                      REGISTER 15
ZERO     EQU   X'ØØ'                   ZERO
SPACE    EQU   C' '                    SPACE
SIGNF    EQU   X'FØ'                   POSITIVE SIGN
WANTALL  EQU   X'FF'                   FLAG SETTINGS
WANTLNK  EQU   X'Ø1'                   GET LNKLST INFO
WANTLPA  EQU   X'Ø2'                   GET LPALST INFO
WANTAPF  EQU   X'Ø4'                   GET APFLST INFO
WANTTSO  EQU   X'Ø8'                   GET TSO INFO
VLFAPPL  AMODE 31
VLFAPPL  RMODE ANY
         BAKR  R14,Ø                   SAVE CALLERS ARS + GPRS
*                                      IN THE LINKAGE STACK
         USING VLFAPPL,R12             INFORM THE ASSEMBLER
         LAE   R12,Ø(R15,Ø)            SETUP PROGRAM BASE REGISTER
         L     R9,=AL4(WORKALEN)       WORK AREA LENGTH
         STORAGE OBTAIN,                                              X
               LENGTH=(R9),            STORAGE LENGTH                 X
               ADDR=(R1Ø),             STORAGE ADDRESS                X
               SP=Ø,                   SUBPOOL                        X
               KEY=8,                  KEY                            X
               LOC=BELOW,              STORAGE BELOW THE LINE         X
               COND=NO,                ABEND IF NO STORAGE            X
               RELATED=(FREEWORK,'FREE WORK AREA')
         LAE   R13,Ø(R1Ø,Ø)            @ THE WORKAREA
         USING SAVEAREA,R13            INFORM THE ASSEMBLER
         LA    RØ,SAVEAREA             @ THE WORKAREA
         ICM   R1,B'1111',=AL4(WORKALEN) LENGTH
         SR    R14,R14                 ZEROFILL
         SR    R15,R15                 PROPAGATE
         MVCL  RØ,R14                  CLEAR THE AREA
         MVC   PREVSA,=C'F1SA'         PUT ACRONYM INTO SAVEAREA
*                                      TO INDICATE STATUS SAVED ON
*                                      THE LINKAGE STACK.
         MVC   MDESETX,MDESETL1        MOVE FOR EXECUTE FORM
         MODESET MF=(E,MDESETX)        SUPV STATE KEY Ø
         BAS   R2,VLF_COFDEFIN         DEFINE THE VLF CLASS
         BAS   R2,VLF_COFIDENT         IDENTIFY A VLF USER
         BAS   R2,VLF_COFCREAT         CREATE OR RETRIEVE AN OBJECT
         BAS   R2,VLF_COFNOFIF         NOFIFY VLF OF OBJECT CHANGE
         BAS   R2,VLF_COFUPDTE         CREATE THE UPDATED OBJECT
         BAS   R2,VLF_COFREMOV         REMOVE A VLF USER
         BAS   R2,VLF_COFPURGE         PURGE A VLF CLASS
RETURN   EQU   *
         MVC   MDESETX,MDESETL2        MOVE FOR EXECUTE FORM
         MODESET MF=(E,MDESETX)        PROB STATE KEY 8
         LAE   R1,Ø(R13,Ø)             ADDRESS TO FREE
         L     R9,=AL4(WORKALEN)       WORK AREA LENGTH
         STORAGE RELEASE,                                             X
               ADDR=(R1),                                             X
               LENGTH=(R9),                                           X
               SP=Ø,                                                  X
               KEY=8,                                                 X
```

```
              COND=NO,                                                 X
              RELATED=(GETWORK,'OBTAIN WORK AREA')
EXIT     EQU   *
         PR                            RESTORE CALLERS AR'S
*                                      GPR'S 2-14 AND RETURN
*                                      TO CALLER
         TITLE 'DEFINE THE VLF CLASS'
VLF_COFDEFIN EQU *
         MVC   VLFCLASS,CLASS_NAME     VLF CLASS NAME
         MVC   VLF_CLASS_MAJORLEN,=AL1(L'VLF_EMAJ) MAJOR NAME LENGTH
         MVC   VLF_CLASS_MINLEN,=AL1(L'VLF_EMIN)   MIN NAME LENGTH
         COFDEFIN CLASS=VLFCLASS,      THE VLF CLASS                  X
               MAJLEN=VLF_CLASS_MAJORLEN, MAJOR CLASS LENGTH          X
               MINLEN=VLF_CLASS_MINLEN,   MIN   CLASS LENGTH          X
               TRIM=ON,                TRIM THE OBJECTS IN THE D/SPACE X
               AUTHRET=YES,            SUPERVISOR STATE               X
               RETCODE=COFDEFIN_RETCODE, RETURN CODE                  X
               RSNCODE=COFDEFIN_RSNCODE, REASON CODE                  X
               MF=(E,VLFDEFIN)
* ON RETURN FROM COFDEFIN R15 WILL CONTAIN A RETURN CODE AND RØØ WILL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE CLASS IS DEFINED TO VLF
* Ø2 = A DEFINE REQUEST FOR THIS CLASS IS ALREADY IN PROGESS
*      OR THE CLASS IS ALREADY DEFINE.
* ØC = THERE WAS NO DESCRIPTION FOR THE CLASS IN THE ACTIVE
*      COFVLFXX PARMLIB MEMBER
* 28 = VLF IS NOT ACTIVE
           C     R15,=F'Ø'             THE CLASS DEFINED TO VLF?
           BER   R2                    YES-
           C     R15,=F'2'             CLASS ALREADY DEFINED?
           BER   R2                    YES-
           ABEND 9Ø1,DUMP              LETS CHECK THE PROBLEM
           BR    R2                    RETURN TO CALLER
           TITLE 'IDENTIFY A VLF USER'
VLF_COFIDENT EQU *
         MVC   VLF_VMAJOR_LIST_NOENTS,=F'1' 1 MAJOR NAME
         MVC   VLF_VMAJOR_NAMES,VLF_EMAJ    MAJOR NAME
         COFIDENT MAJNLST=VLF_VMAJOR_LIST,  EMAJ LIST                 X
               CLASS=CLASS_NAME,         CLASS NAME                   X
               SCOPE=SYSTEM,             ALL TASKS IN THE SYSTEM      X
               UTOKEN=VLF_TOKEN,         USER TOKEN                   X
               RETCODE=COFIDENT_RETCODE, RETURN CODE                  X
               RSNCODE=COFIDENT_RSNCODE, REASON CODE                  X
               MF=(E,VLFIDENT)
* ON RETURN FROM COFIDENT R15 WILL CONTAIN A RETURN CODE AND RØØ WILL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE USER HAS BEEN IDENTIFIED TO VLF WITH THE SPECIFIED
```

```
*      MAJOR NAME SEARCH ORDER
* Ø2 = THE USER IS ALREADY IDENTIFIED TO VLF FOR THIS CLASS
*      OR THE CLASS IS ALREADY DEFINE.
* ØC = THE CLASS HAS NOT BEEN DEFINED TO VLF
*      COFVLFXX PARMLIB MEMBER
* 18 = THERE WAS AN ERROR IN THE PARAMETER LIST
* 28 = VLF IS NOT ACTIVE
         C     R15,=F'Ø'              USER DEFINED TO VLF?
         BER   R2                     YES-
         C     R15,=F'2'              USER ALREADY DEFINED FOR THIS
*                                     CLASS?
         BER   R2                     YES-
         ABEND 9Ø2,DUMP               LETS CHECK IT OUT
         BR    R2                     RETURN TO CALLER
         TITLE 'CREATE A VLF OBJECT'
VLF_COFCREAT EQU *
         MVC   VLF_TLIST_NOENTS,=F'1'  NO OF TLIST ENTRIES
         MVC   VLF_TLIST_PASN,=F'Ø'    ALET OF TLIST(PASN)
         XC    VLF_MINOR_OBJECT,VLF_MINOR_OBJECT INITIALISE
         LA    RIØ,VLF_MINOR_OBJECT    OBJECT AREA ADDRESS
         STCM  RIØ,B'1111',VLF_TLIST_TARGET@ STORE IN PLIST
         MVC   VLF_TLIST_TARGET@_LEN,=AL4(L'VLF_MINOR_OBJECT)
*                                     LENGTH OF OBJECT
         MVC   VLF_TLIST_TASIZE,=AL4(VLF_TLIST_TASIZE_LEN)
*                                     TARGET AREA LIST SIZE
         MVC   VLF_TLIST_OBJSIZE,=F'Ø' INIT RETURNED VLF OBJECT SIZE
         MVC   VLF_TLIST_CINDEX,=F'Ø'  INIT CONCATENATION INDEX
         COFRETRI MINOR=VLF_EMIN,      VLF MINOR NAME                  X
               UTOKEN=VLF_TOKEN,       USER TOKEN                      X
               TLIST=VLF_TLIST,        TARGET AREA LIST                X
               TLSIZE=VLF_TLIST_TASIZE, TARGET AREA LIST SIZE          X
               OBJSIZE=VLF_TLIST_OBJSIZE, VLF RETURNED OBJECT SIZE     X
               CINDEX=VLF_TLIST_CINDEX, CONCATENATION INDEX            X
               RETCODE=COFRETRI_RETCODE, RETURN CODE                   X
               RSNCODE=COFRETRI_RSNCODE, REASON CODE                   X
               MF=(E,VLFRETRI)
* ON RETURN FROM COFRETRI R15 WILL CONTAIN A RETURN CODE AND RØØ WIL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE VLF OBJECT WAS SUCCESSFULLY RETRIEVED.
* Ø2 = A VLF OBJECT HAS BEEN RETRIEVED THAT MIGHT BE THE CORRECT
*      OBJECT FOR THE USER, BUT THE OBJECT MIGHT ALSO EXIST IN
*      EARLIER MAJOR NAMES IN THE USER'S MAJOR NAME LIST.
* Ø4 = THE VLF OBJECT WAS RETRIEVED, BUT THE TARGET AREAS DID NOT
*      RECEIVE THE ENTIRE OBJECT
* Ø6 = A VLF OBJECT HAS BEEN RETRIEVED THAT MIGHT BE THE CORRECT
*      OBJECT FOR THE USER, BUT THE OBJECT MIGHT ALSO EXIST IN
*      EARLIER MAJOR NAMES IN THE USER'S MAJOR NAME LIST.
*      ADDITIONALLY, THE TRAGET AREAS DID NOT RECEIVE THE ENTIRE
*      OBJECT.
* Ø8 = THE OBJECT DOES NOT EXIST IN VLF.
```

```
* ØC = THE CLASS TO WHICH THE USER IS IDENTIFIED IS NOT CURRENTLY
*      DEFINED.
Ø2140Ø2Ø
* 1Ø = AN UNKNOWN USER TOKEN WAS SPECIFIED.
* 28 = VLF IS NOT ACTIVE
         C     R15,=F'Ø'              OBJECT RETRIEVED?
         BER   R2                     YES-
         C     R15,=F'8'              OBJECT NOT IN VLF?
         BE    CREATE_VLF_OBJECT      YES- CREATE IT
         ABEND 9Ø3,DUMP               LETS CHECK IT OUT
         BR    R2                     RETURN TO CALLER
CREATE_VLF_OBJECT EQU *
         MVC   VLF_OBJPRT_PARTS,=F'1'  NO OF OBJECT PARTS LIST ENTS
         MVC   VLF_OBJPRT_PASN,=F'Ø'   ALET OF OBJPRT(PASN)
         LA    R1Ø,VLF_MINOR_OBJECT    OBJECT AREA ADDRESS
         STCM  R1Ø,B'1111',VLF_OBJPRT_DATA_PART@ STORE IN PLIST
         MVC   VLF_OBJPRT_DATA_PLEN,=AL4(L'VLF_MINOR_OBJECT)
         MVC   VLF_OBJPRT_LEN,=AL4(VLF_OBJPRT_SIZE)
*                                     LENGTH OF THE OBJECTS PART LIST
         MVC   VLF_MINOR_OBJECT,VLF_MESSAGE OBJECT DATA
         COFCREAT MAJOR=VLF_EMAJ,     VLF MAJOR NAME                  X
               MINOR=VLF_EMIN,        VLF MINOR NAME                  X
               UTOKEN=VLF_TOKEN,      USER TOKEN                      X
               REPLACE=YES,           REPLACE IF THERE                X
               OBJPRTL=VLF_OBJPRTL,   OBJECT PARTS LIST               X
               OBJPLSZ=VLF_OBJPRT_LEN,                                X
               RETCODE=COFCREAT_RETCODE, RETURN CODE                  X
               RSNCODE=COFCREAT_RSNCODE, REASON CODE                  X
               MF=(E,VLFCREAT)
* ON RETURN FROM COFCREAT R15 WILL CONTAIN A RETURN CODE AND RØØ WILL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE VLF OBJECT WAS SUCCESSFULLY RETRIEVED.
* Ø2 = NO VLF OBJECT WAS CREATED.
* Ø4 = THE REQUESTED MAJOR NAME IS NOT IN THE USERS'S SEARCH ORDER.
* ØC = THE CLASS TO WHICH THE USER IS IDENTIFIED IS NOT CURRENTLY
*      DEFINED.
* 1Ø = AN UNKNOWN USER TOKEN WAS SPECIFIED.
* 1C = THERE WAS NOT ENOUGH STORAGE AVAILABLE TO CREATE THIS OBJECT.
* 28 = VLF IS NOT ACTIVE
         C     R15,=F'Ø'              OBJECT CREATED?
         BE    VLF_COFCREAT           YES- LETS RETRIEVE IT
         ABEND 9Ø4,DUMP               LETS CHECK IT OUT
         BR    R2                     RETURN TO CALLER
         TITLE 'NOTIFY VLF OF OBJECT CHANGE'
VLF_COFNOFIF EQU *
         COFNOTIF FUNC=UPDMINOR,      UPDATE MINOR OBJECT             X
               MAJOR=VLF_EMAJ,        MAJOR NAME                      X
               MINLIST=VLF_EMIN,      MINOR LIST                      X
               CLASS=CLASS_NAME,      CLASS NAME                      X
               RETCODE=COFNOTIF_RETCODE, RETURN CODE                  X
               RSNCODE=COFNOTIF_RSNCODE, REASON CODE                  X
```

```
               MF=(E,VLFNOTIF)
          C    R15,=F'Ø'              VLF REFLECTS VLF CHANGE?
          BER  R2                     YES-
          ABEND 9Ø5,DUMP              LETS CHECK IT OUT
          BR   R2                     RETURN TO CALLER
          TITLE 'CREATE THE UPDATED VLF OBJECT'
VLF_COFUPDTE EQU *
          MVC  VLF_TLIST_NOENTS,=F'1' NO OF TLIST ENTRIES
          MVC  VLF_TLIST_PASN,=F'Ø'   ALET OF TLIST(PASN)
          XC   VLF_MINOR_OBJECT,VLF_MINOR_OBJECT INITIALISE
          LA   R1Ø,VLF_MINOR_OBJECT   OBJECT AREA ADDRESS
          STCM R1Ø,B'1111',VLF_TLIST_TARGET@ STORE IN PLIST
          MVC  VLF_TLIST_TARGET@_LEN,=AL4(L'VLF_MINOR_OBJECT)
*                                     LENGTH OF OBJECT
          MVC  VLF_TLIST_TASIZE,=AL4(VLF_TLIST_TASIZE_LEN)
*                                     TARGET AREA LIST SIZE
          MVC  VLF_TLIST_OBJSIZE,=F'Ø' INIT RETURNED VLF OBJECT SIZE
          MVC  VLF_TLIST_CINDEX,=F'Ø' INIT CONCATENATION INDEX
          COFRETRI MINOR=VLF_EMIN,    VLF MINOR NAME                  X
               UTOKEN=VLF_TOKEN,      USER TOKEN                      X
               TLIST=VLF_TLIST,       TARGET AREA LIST                X
               TLSIZE=VLF_TLIST_TASIZE, TARGET AREA LIST SIZE         X
               OBJSIZE=VLF_TLIST_OBJSIZE, VLF RETURNED OBJECT SIZE    X
               CINDEX=VLF_TLIST_CINDEX, CONCATENATION INDEX           X
               RETCODE=COFRETRI_RETCODE, RETURN CODE                  X
               RSNCODE=COFRETRI_RSNCODE, REASON CODE                  X
               MF=(E,VLFRETRI)
* ON RETURN FROM COFRETRI R15 WILL CONTAIN A RETURN CODE AND RØØ WIL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE VLF OBJECT WAS SUCCESSFULLY RETRIEVED.
* Ø2 = A VLF OBJECT HAS BEEN RETRIEVED THAT MIGHT BE THE CORRECT
*      OBJECT FOR THE USER, BUT THE OBJECT MIGHT ALSO EXIST IN
*      EARLIER MAJOR NAMES IN THE USER'S MAJOR NAME LIST.
* Ø4 = THE VLF OBJECT WAS RETRIEVED, BUT THE TARGET AREAS DID NOT
*      RECEIVE THE ENTIRE OBJECT
* Ø6 = A VLF OBJECT HAS BEEN RETRIEVED THAT MIGHT BE THE CORRECT
*      OBJECT FOR THE USER, BUT THE OBJECT MIGHT ALSO EXIST IN
*      EARLIER MAJOR NAMES IN THE USER'S MAJOR NAME LIST.
*      ADDITIONALLY, THE TARGET AREAS DID NOT RECEIVE THE ENTIRE
*      OBJECT.
* Ø8 = THE OBJECT DOES NOT EXIST IN VLF.
* ØC = THE CLASS TO WHICH THE USER IS IDENTIFIED IS NOT CURRENTLY
*      DEFINED.
* 1Ø = AN UNKNOWN USER TOKEN WAS SPECIFIED.
* 28 = VLF IS NOT ACTIVE
          C    R15,=F'Ø'              OBJECT RETRIEVED?
          BER  R2                     YES-
          C    R15,=F'8'              OBJECT NOT IN VLF?
          BE   UPDATE_VLF_OBJECT      YES- CREATE UPDATED OBJECT
          ABEND 9Ø6,DUMP              LETS CHECK IT OUT
```

```
         BR    R2                      RETURN TO CALLER
UPDATE_VLF_OBJECT EQU *
         MVC   VLF_OBJPRT_PARTS,=F'1'  NO OF OBJECT PARTS LIST ENTS
         MVC   VLF_OBJPRT_PASN,=F'Ø'   ALET OF OBJPRT(PASN)
         LA    R1Ø,VLF_MINOR_OBJECT    OBJECT AREA ADDRESS
         STCM  R1Ø,B'1111',VLF_OBJPRT_DATA_PART@ STORE IN PLIST
         MVC   VLF_OBJPRT_DATA_PLEN,=AL4(L'VLF_MINOR_OBJECT)
         MVC   VLF_OBJPRT_LEN,=AL4(VLF_OBJPRT_SIZE)
*                                     LENGTH OF THE OBJECTS PART LIST
         MVC   VLF_MINOR_OBJECT,VLF_UPD_MES  THE UPDATED OBJECT
         COFCREAT MAJOR=VLF_EMAJ,      VLF MAJOR NAME                  X
               MINOR=VLF_EMIN,         VLF MINOR NAME                  X
               UTOKEN=VLF_TOKEN,       USER TOKEN                      X
               REPLACE=YES,            REPLACE IF THERE                X
               OBJPRTL=VLF_OBJPRTL,    OBJECT PARTS LIST               X
               OBJPLSZ=VLF_OBJPRT_LEN,                                 X
               RETCODE=COFCREAT_RETCODE, RETURN CODE                   X
               RSNCODE=COFCREAT_RSNCODE, REASON CODE                   X
               MF=(E,VLFCREAT)
* ON RETURN FROM COFCREAT R15 WILL CONTAIN A RETURN CODE AND RØØ WIL
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = THE VLF OBJECT WAS SUCCESSFULLY RETRIEVED.
* Ø2 = NO VLF OBJECT WAS CREATED.
* Ø4 = THE REQUESTED MAJOR NAME IS NOT IN THE USER'S SEARCH ORDER.
* ØC = THE CLASS TO WHICH THE USER IS IDENTIFIED IS NOT CURRENTLY
*      DEFINED.
* 1Ø = AN UNKNOWN USER TOKEN WAS SPECIFIED.
* 1C = THERE WAS NOT ENOUGH STORAGE AVAILABLE TO CREATE THIS OBJECT.
* 28 = VLF IS NOT ACTIVE
         C     R15,=F'Ø'               OBJECT CREATED?
         BE    VLF_COFUPDTE            YES- LETS RETRIEVE IT
         ABEND 9Ø7,DUMP                LETS CHECK IT OUT
         BR    R2                      RETURN TO CALLER
         TITLE 'REMOVE A VLF USER'
VLF_COFREMOV EQU *
         COFREMOV UTOKEN=VLF_TOKEN,    USER TOKEN                      X
               RETCODE=COFREMOV_RETCODE, RETURN CODE                   X
               RSNCODE=COFREMOV_RSNCODE, REASON CODE                   X
               MF=(E,VLFREMOV)
         LTR   R15,R15                 TOKEN OKAY?
         BZR   R2                      YES-
         ABEND 9Ø8,DUMP                LETS CHECK IT OUT
         BR    R2                      RETURN TO CALLER
         TITLE 'PURGE A VLF CLASS'
VLF_COFPURGE EQU *
         COFPURGE CLASS=CLASS_NAME,    CLASS NAME                      X
               RETCODE=COFPURGE_RETCODE, RETURN CODE                   X
               RSNCODE=COFPURGE_RSNCODE, REASON CODE                   X
               MF=(E,VLFPURGE)
* ON RETURN FFROM COFPURGE R15 WILL CONTAIN A RETURN CODE AND RØØ WIL
```

```
* CONTAIN A REASON CODE. REFER TO THE IBM MANUAL AUTHORISED ASSEMBLER
* REFERENCE FOR A COMPLETE DESCRIPTION.
*
* SOME IMPORTANT RETURN CODES TO CHECK FOR ARE:
* ØØ = SUCCESSFUL COMPLETION. THE CLASS IS NO LONGER DECRIBED TO VLF.
* Ø2 = THE SPECIFIED CLASS WAS NOT DESCRIBED IN THE ACTIVE COFVLFXX
*      MEMBER.
* 28 = VLF IS NOT ACTIVE
         LTR   R15,R15                 CLASS PURGED?
         BZR   R2                      YES-
         ABEND 9Ø9,DUMP                LETS CHECK IT OUT
         BR    R2                      RETURN TO CALLER
         TITLE 'LTORG AREA'
         LTORG
         TITLE 'STORAGE AREAS'
CLASS_NAME  DC  CL7'VLFMESS'           VLF CLASS NAME
VLF_EMAJ    DC  CL8'VMESSAGE'          VLF MAJOR NAME(VMESSAGE)
VLF_EMIN    DC  CL8'VMESSØØ1'          VLF MIN NAME LENGTH(VMESSNNN)
VLF_MESSAGE DC  CL256'VLFMESS*** THIS IS VLF CLASS=VLFMESS, VLF MAJOR=VX
               MESSAGE, VLF MINOR=VMESSØØ1. VLFMESS***'
VLF_UPD_MES DC  CL256'VLFMESS*** THIS IS THE UPDATED VLF CLASS=VLFMESS,X
               VLF MAJOR=MESSAGE, VLF MINOR=VMESSØØ1. UPDATED **'
         TITLE 'MACRO LIST AREA'
LINKL    LINK SF=L
LINKLEN  EQU   *-LINKL                 LENGTH
CALLL    CALL ,(,,,,,,,),MF=L
CALLLEN  EQU   *-CALLL                 LENGTH
MDESETL1 MODESET KEY=ZERO,MODE=SUP,MF=L LIST FORM OF MODESET
MSETLEN1 EQU   *-MDESETL1              LENGTH OF PARAMETER LIST
MDESETL2 MODESET KEY=NZERO,MODE=PROB,MF=L  LIST FORM OF MODESET
MSETLEN2 EQU   *-MDESETL2              LENGTH OF PARAMETER LIST
         TITLE 'WORKAREA DSECT'
WORKAREA DSECT
SAVEAREA DS    CL72                    SAVEAREA
PREVSA   EQU   SAVEAREA+4,4            @ OF PREVIOUS SAVEAREA
DW       DS    D                       WORK AREA
         DS    ØF                      ALIGNMENT
VLF_MINOR_OBJECT DS   CL256            MINOR OBJECT
         COFDEFIN MF=(L,VLFDEFIN)      VLFDEFIN RE-ENTRANT AREA
VLFCLASS           DS CL8              VLF CLASS
VLF_CLASS_MAJORLEN DS X                MAJOR CLASS LENGTH
VLF_CLASS_MINLEN   DS X                MIN CLASS LENGTH
COFDEFIN_RETCODE   DS F                RETURN CODE
COFDEFIN_RSNCODE   DS F                REASON CODE
         DS    ØF                      ALIGNMENT
         COFIDENT MF=(L,VLFIDENT)      VLFIDENT RE-ENTRANT AREA
VLF_TOKEN              DS CL16         USER TOKEN
VLF_VMAJOR_LIST        DS ØX           MAJOR CLASS LIST
VLF_VMAJOR_LIST_NOENTS DS AL4          MAJOR CLASS LIST
VLF_VMAJOR_NAMES       DS CL(L'VLF_EMAJ) LIST OF VLF MAJOR NAMES
COFIDENT_RETCODE       DS F            RETURN CODE
COFIDENT_RSNCODE       DS F            REASON CODE
         DS    ØF                      ALIGNMENT
```

```
         COFRETRI MF=(L,VLFRETRI)      VLFRETRI RE-ENTRANT AREA
VLF_TLIST              DS ØX           VLF TLIST AREA
VLF_TLIST_NOENTS       DS F            NO OF TLIST ENTRIES
VLF_TLIST_PASN         DS F            ALET OF TLIST(PASN)
VLF_TLIST_TARGET@      DS F            TARGET @ OF OBJECT AREA
VLF_TLIST_TARGET@_LEN  DS F            TARGET LENGTH OF OBJECT AREA
VLF_TLIST_TASIZE_LEN   EQU *-VLF_TLIST SIZE OF THE TARGET AREA LIST
VLF_TLIST_TASIZE       DS F            TARGET AREA LIST SIZE
VLF_TLIST_OBJSIZE      DS F            RETURNED VLF OBJECT SIZE
VLF_TLIST_CINDEX       DS F            CONCATENATION INDEX
COFRETRI_RETCODE       DS F            RETURN CODE
COFRETRI_RSNCODE       DS F            REASON CODE
         DS    ØF                      ALIGNMENT
         COFCREAT MF=(L,VLFCREAT)      VLFCREAT RE-ENTRANT AREA
VLF_OBJPRTL            DS ØX           VLF OBJPRTL AREA
VLF_OBJPRT_PARTS       DS F            NO OF OBJPRT PARTS
VLF_OBJPRT_PASN        DS F            ALET OF OBJPRT(PASN)
VLF_OBJPRT_DATA_PART@  DS F            TARGET @ OF OBJECT AREA
VLF_OBJPRT_DATA_PLEN   DS F            TARGET OBJECT AREA LENGTH
VLF_OBJPRT_SIZE        EQU *-VLF_OBJPRTL SIZE OF THE OBJPRTL PLIST
VLF_OBJPRT_LEN         DS F            LENGTH OF THE OBJECTS PART LIST
COFCREAT_RETCODE       DS F            RETURN CODE
COFCREAT_RSNCODE       DS F            REASON CODE
         DS    ØF                      ALIGNMENT
         COFNOTIF MF=(L,VLFNOTIF)      VLFNOFIF RE-ENTRANT AREA
COFNOTIF_RETCODE       DS F            RETURN CODE
COFNOTIF_RSNCODE       DS F            REASON CODE
         DS    ØF
Ø4520030
         COFREMOV MF=(L,VLFREMOV)      VLFREMOV RE-ENTRANT AREA
COFREMOV_RETCODE       DS F            RETURN CODE
COFREMOV_RSNCODE       DS F            REASON CODE
         DS    ØF
Ø4560028
         COFPURGE MF=(L,VLFPURGE)      VLFPURGE RE-ENTRANT AREA
COFPURGE_RETCODE       DS F            RETURN CODE
COFPURGE_RSNCODE       DS F            REASON CODE
         DS    ØF                      ALIGNMENT
MDESETX  DS    XL(MSETLEN1)            MODESET EXECUTE FORM
         DS    ØF
LINKAREA DS    CL(LINKLEN)             LINK AREA
         DS    ØF
CALLAREA DS    CL(CALLLEN)             PARM LIST AREA
*                                      INFORMATION AREA
WORKALEN EQU   *-WORKAREA              WORK AREA LENGTH
         TITLE 'MVS MAPPINGS'
         CVT DSECT=YES,LIST=NO,PREFIX=NO CVT
         IHAPSA DSECT=YES,LIST=NO        PSA
         END   VLFAPPL
```

Rem F Perretta
Senior Systems Programmer
Millenium Computer Consultancy (UK)

An improved MQSeries batch trigger monitor

THE PROBLEM

At my company we have discovered a need to trigger batch jobs from MQSeries queues. IBM supplies, in support pack MA12, what they call a sample batch trigger monitor. This program runs as a started task and waits for a trigger message on an initiation queue and then triggers a job pointed to by a DD card in its own JCL. The program as implemented can only start one job for each batch trigger monitor that is running. We found this to be too narrowly focused so I set out to design a more versatile replacement.

A SOLUTION

The revised monitor we produced runs as a started task and awaits a trigger message on a single initiation queue as did IBM's, but the revised monitor does not rely on a DD card to get the JOB card JCL to submit to the internal reader. Instead, I decided to use the trigger message itself to supply the JCL. To do this you define a QLOCAL as trigger first with its initiation queue as the queue that the trigger monitor is monitoring. You then define a PROCESS that has the JOB JCL coded in the APPLICID, ENVRDATA, and the USERDATA fields. The APPLICID field allows 256 bytes and the ENVRDATA and USERDATA fields allow for 128 bytes each. These fields are broken up into 64-byte records so you can code eight different JCL cards that this program will deliver to the internal reader. That should be enough to start most jobs.

Using this method you can define as many triggered QLOCALs as you have jobs you want to start, and point them all to a single initiation queue that this program is watching. With all the JCL provided in the PROCESS definition no DD cards are needed in the trigger monitor started task and no modifications are needed in the program when a new job is needed to be started.

PROGRAM LOGIC

When the trigger monitor is started it first reads the PARM input to determine the name of the initiation queue to listen on and the name of the Queue Manager to connect to. It then connects to the Queue Manager and opens the initiation queue. While it is doing this, it is putting out little informational messages to the SYSPRINT DD. The program then goes into a GET wait on the initiation queue until it is awakened by a trigger message. When a trigger message arrives, the monitor wakes up and parses the message for the JCL it is to submit to the internal reader. After doing this, it then goes back into a GET wait on the initiation queue.

STOPPING THE TRIGGER MONITOR

To stop the trigger monitor you should put a message on the initiation queue which is of type REPORT and feedback of MQFB-QUIT. The program CKTIEND, which is part of support pac MA12, can generate this message. You can download CKTIEND from the IBM Web site (http://www.software.hosting.ibm.com/ts/mqseries/txppacs/ma12.html). The JCL to run CKTIEND follows the program source.

DPKUT200

```
DYNAM,LIB,OBJECT,RENT,RES,APOST
* ---------------------------------------------------------------- *
 IDENTIFICATION DIVISION
* ---------------------------------------------------------------- *
 PROGRAM-ID. DPKUT200.
 DATE-WRITTEN  AUG, 1998.
* ---------------------------------------------------------------- *
 ENVIRONMENT DIVISION
* ---------------------------------------------------------------- *
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SYSPRINT ASSIGN TO UT-S-SYSPRINT.
     SELECT INTRDR   ASSIGN TO UT-S-INTRDR.
* ---------------------------------------------------------------- *
 DATA DIVISION
* ---------------------------------------------------------------- *
 FILE SECTION.
 FD  SYSPRINT
     BLOCK CONTAINS 0 RECORDS
     RECORDING MODE IS F.
 01  PR-PRINT-REC.
     05  PR-CARRIAGE-CONTROL     PIC X.
```

```
     05  PR-PRINT-DATA           PIC X(132).
 FD  INTRDR
     BLOCK CONTAINS  0 RECORDS
     RECORDING MODE IS F.
 01  ID-INTRDR-DATA              PIC X(80).
* ---------------------------------------------------------------- *
 WORKING-STORAGE SECTION
* ---------------------------------------------------------------- *
*    GENERAL WORK FIELDS
 01  ID-INPUT-DATE.
     05  ID-YEAR                 PIC 99.
     05  ID-MONTH                PIC 99.
     05  ID-DAY                  PIC 99.
 01  IT-INPUT-TIME.
     05  IT-HOURS                PIC 99.
     05  IT-MINUTES              PIC 99.
     05  IT-SECONDS              PIC 99.
     05  IT-HUNDRETHS            PIC 99.
 01  RC-RETURN-CODE              PIC S9(04) BINARY  VALUE ZERO.
 01  WS-WORKING-STORAGE.
     05  WS-MQCONN               PIC X(8)  VALUE 'CSQBCONN'.
     05  WS-MQOPEN               PIC X(8)  VALUE 'CSQBOPEN'.
     05  WS-MQINQ                PIC X(8)  VALUE 'CSQBINQ'.
     05  WS-MQGET                PIC X(8)  VALUE 'CSQBGET'.
     05  WS-MQPUT1               PIC X(8)  VALUE 'CSQBPUT1'.
     05  WS-MQCMIT               PIC X(8)  VALUE 'CSQBCOMM'.
     05  WS-MQBACK               PIC X(8)  VALUE 'CSQBBACK'.
     05  WS-MQCLOSE              PIC X(8)  VALUE 'CSQBCLOS'.
     05  WS-MQDISC               PIC X(8)  VALUE 'CSQBDISC'.
*    LINES OF THE PRINT REPORT
01  H1-HEADER-1.
     05  FILLER                  PIC X(10) VALUE SPACES.
     05  H1-MM                   PIC 99.
     05  FILLER                  PIC X     VALUE '/'.
     05  H1-DD                   PIC 99.
     05  FILLER                  PIC X     VALUE '/'.
     05  H1-YY                   PIC 99.
     05  FILLER                  PIC X(8)  VALUE SPACES.
     05  FILLER                  PIC X(21) VALUE
                                           'Batch Trigger Monitor'.
     05  FILLER                  PIC X(85) VALUE SPACES.
 01  H2-HEADER-2.
     05  FILLER                  PIC X(5)  VALUE SPACES.
     05  FILLER                  PIC X(29) VALUE
                                 '        Queue Manager Name: '.
     05  H2-MQM-NAME             PIC X(48) VALUE SPACES.
     05  FILLER                  PIC X(50) VALUE SPACES.
 01  H3-HEADER-3.
     05  FILLER                  PIC X(17) VALUE SPACES.
     05  FILLER                  PIC X(17) VALUE
                                              ' Trigger Queue: '.
     05  H3-QUEUE-NAME           PIC X(48) VALUE SPACES.
     05  FILLER                  PIC X(50) VALUE SPACES.
```

```
 Ø1  H4-HEADER-4.
     Ø5  H4-HOURS                PIC X(2).
     Ø5  FILLER                  PIC X VALUE ':'.
     Ø5  H4-MINUTES              PIC X(2).
     Ø5  FILLER                  PIC X VALUE ':'.
     Ø5  H4-SECONDS              PIC X(2).
     Ø5  FILLER                  PIC X(3)  VALUE SPACES.
     Ø5  H4-HEADER-4-DATA        PIC X(8Ø) VALUE SPACES.
 Ø1  H5-HEADER-5.
     Ø5  FILLER                  PIC X(11) VALUE SPACES.
     Ø5  H5-HEADER-5-DATA        PIC X(69) VALUE SPACES.
*    DATA FIELDS DERIVED FROM THE PARM FIELD
 Ø1  IP-INPUT-PARMS.
     Ø5  IP-MQM                      PIC X(48) VALUE SPACES.
     Ø5  IP-OBJECT                   PIC X(48) VALUE SPACES.
*    MQM API FIELDS
 Ø1  AF-API-FIELDS.
     Ø5  AF-BUFFER-LENGTH            PIC S9(9) BINARY  VALUE 8ØØ.
     Ø5  AF-HCONN                    PIC S9(9) BINARY.
     Ø5  AF-OPTIONS                  PIC S9(9) BINARY.
     Ø5  AF-HOBJ                     PIC S9(9) BINARY.
     Ø5  AF-DATA-LENGTH              PIC S9(9) BINARY.
     Ø5  AF-COMPCODE                 PIC S9(9) BINARY.
     Ø5  AF-REASON                   PIC S9(9) BINARY.
     Ø5  AF-MESSAGE-DATA.
         1Ø MQTM-STRUCTURE.
             15 AF-STRUCID       PIC X(4) VALUE 'TM  '.
             15 AF-VERSION       PIC S9(9) BINARY VALUE 1.
             15 AF-QNAME         PIC X(48) VALUE SPACES.
             15 AF-PROCESSNAME PIC X(48) VALUE SPACES.
             15 AF-TRIGGERDATA PIC X(64) VALUE SPACES.
             15 AF-APPLTYPE      PIC S9(9) BINARY VALUE Ø.
             15 AF-APPLID        PIC X(256) VALUE SPACES.
             15 AF-ENVDATA       PIC X(128) VALUE SPACES.
             15 AF-USERDATA      PIC X(128) VALUE SPACES.
 Ø1 AA-APPLID-ARRAY.
     Ø5 AA-APPLID-ARRAY-1.
        1Ø AA-1-SLASH                 PIC XX.
        1Ø FILLER                     PIC X(62).
     Ø5 AA-APPLID-ARRAY-2.
        1Ø AA-2-SLASH                 PIC XX.
        1Ø FILLER                     PIC X(62).
     Ø5 AA-APPLID-ARRAY-3.
        1Ø AA-3-SLASH                 PIC XX.
        1Ø FILLER                     PIC X(62).
     Ø5 AA-APPLID-ARRAY-4.
        1Ø AA-4-SLASH                 PIC XX.
        1Ø FILLER                     PIC X(62).
 Ø1 EA-ENVDATA-ARRAY.
     Ø5 EA-ENVDATA-ARRAY-1.
        1Ø EA-1-SLASH                PIC XX.
        1Ø FILLER                     PIC X(62).
     Ø5 EA-ENVDATA-ARRAY-2.
```

```
           10 EA-2-SLASH                  PIC XX.
           10 FILLER                      PIC X(62).
       01 UA-USERDATA-ARRAY.
           05 UA-USERDATA-ARRAY-1.
              10 UA-1-SLASH               PIC XX.
              10 FILLER                   PIC X(62).
           05 UA-USERDATA-ARRAY-2.
              10 UA-2-SLASH               PIC XX.
              10 FILLER                   PIC X(62).
      *    ERROR AND INFORMATION MESSAGES
       01  MESSAGES.
           05  M-MESSAGE-0.
               10  FILLER              PIC X(53) VALUE
                     ' ********** TERMINATION MESSAGE RECEIVED FROM: '.
               10  M-USERID            PIC X(8) VALUE SPACES.
               10  FILLER              PIC X(71) VALUE SPACES.
           05  M-MESSAGE-1.
               10  FILLER              PIC X(10)  VALUE SPACES.
               10  FILLER              PIC X(122) VALUE
              '********** NO DATA PASSED TO PROGRAM. PROGRAM REQUIRES A
      -       'QUEUE MANAGER NAME AND A QUEUE NAME. **********'.
           05  M-MESSAGE-2.
               10  FILLER              PIC X(25)  VALUE SPACES.
               10  FILLER              PIC X(107) VALUE
              '********** NO QUEUE MANAGER NAME PASSED TO PROGRAM - DEFA
      -       'ULT USED *****'.
           05  M-MESSAGE-3.
               10  FILLER              PIC X(38) VALUE SPACES.
               10  FILLER              PIC X(94) VALUE
              '********** NO QUEUE NAME PASSED TO PROGRAM. **********'.
           05  M-MESSAGE-4.
               10  FILLER              PIC X(13) VALUE SPACES.
               10  FILLER              PIC X(32) VALUE
                   '********** AN ERROR OCCURRED IN '.
               10  M-MSG4-TYPE         PIC X(10).
               10  FILLER              PIC X(20) VALUE
                   '. COMPLETION CODE = '.
               10  M-MSG4-COMPCODE     PIC Z(8)9.
               10  FILLER              PIC X(15) VALUE ' REASON CODE ='.
               10  M-MSG4-REASON       PIC Z(8)9.
               10  FILLER              PIC X(24) VALUE ' **********'.
      *    The following copy files define API control blocks.
       01  MQM-OBJECT-DESCRIPTOR.
           COPY CMQODV.
       01  MQM-MESSAGE-DESCRIPTOR.
           COPY CMQMDV.
       01  MQM-GET-MESSAGE-OPTIONS.
           COPY CMQGMOV.
      *    Copy file of constants (for filling in the control blocks)
      *    and return codes (for testing the result of a call)
       01  MQM-CONSTANTS.
           COPY CMQV.
      *    RETURN VALUES
```

```
 Ø1  RV-RETURN-VALUES.
     Ø5  RV-CSQ4-OK              PIC S9(4) VALUE Ø.
     Ø5  RV-CSQ4-WARNING         PIC S9(4) VALUE 4.
     Ø5  RV-CSQ4-ERROR           PIC S9(4) VALUE 8.
*    END OF JOB INDICATOR
 Ø1  E-ENDJOB.
     Ø5  E-END-JOB               PIC 9 VALUE IS Ø.
         88 E-END-OF-JOB         VALUE IS 1.
     Ø5  EO-TRUE                 PIC 9 VALUE 1.
*    ADDITIONAL JOB CARDS TO WRITE OUT
 Ø1  AC-ADDITIONAL-CARDS.
     Ø5  AC-SUBMITED-BY-CARD.
         1Ø FILLER               PIC X(4)  VALUE '//* '.
         1Ø FILLER               PIC X(76) VALUE
                        'JOB SUBMITTED BY DPKUT2ØØ'.
     Ø5  AC-EOF-CARD             PIC X(8Ø) VALUE
         '/*EOF'.
 LINKAGE SECTION.
 Ø1  P-PARMDATA.
     Ø5  P-PARM-LEN              PIC S9(Ø3) BINARY.
     Ø5  P-PARM-STRING           PIC X(1ØØ).
*
     EJECT
* ---------------------------------------------------------------- *
 PROCEDURE DIVISION USING P-PARMDATA
* ---------------------------------------------------------------- *
 A-MAIN SECTION.
* ---------------------------------------------------------------- *
* This section receives the names of the queue manager and the  *
* queue from the PARM statement in the JCL. It opens the queue, *
* reads all the messages, and then prints them.                 *
* ---------------------------------------------------------------- *
*    Open the print file, initialize the fields for the
*    header date and the page number, and print the first
*    line of the header
     OPEN OUTPUT SYSPRINT.
     ACCEPT ID-INPUT-DATE FROM DATE.
     MOVE ID-MONTH TO H1-MM.
     MOVE ID-DAY TO H1-DD.
     MOVE ID-YEAR TO H1-YY.
*
     MOVE H1-HEADER-1 TO PR-PRINT-DATA.
     WRITE PR-PRINT-REC AFTER ADVANCING PAGE.
*
*    If no data was passed, create a message, print, and exit
*
     IF P-PARM-LEN = Ø THEN
        MOVE M-MESSAGE-1 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC
        MOVE RV-CSQ4-WARNING TO RC-RETURN-CODE
        GO TO A-MAIN-END
     END-IF.
*
```

```
*    Separate into the relevant fields any data passed in the
*    PARM statement
*
     UNSTRING P-PARM-STRING DELIMITED BY ALL ','
                            INTO IP-MQM
                                 IP-OBJECT.
*    Move the data (spaces if nothing is entered) into the
*    relevant print fields
     MOVE IP-MQM     TO H2-MQM-NAME.
     MOVE IP-OBJECT TO H3-QUEUE-NAME.
*    Print a message if the queue manager name is missing, the
*    default queue manager will be used
*
     IF IP-MQM = SPACES OR IP-MQM = LOW-VALUES THEN
        MOVE M-MESSAGE-2 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC
     END-IF.
*    Print a message if the queue name is missing and exit from
*    the program
     IF IP-OBJECT = SPACES OR IP-OBJECT = LOW-VALUES THEN
        MOVE M-MESSAGE-3 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC
        MOVE RV-CSQ4-WARNING TO RC-RETURN-CODE
        GO TO A-MAIN-END
     END-IF.
*    Print the remaining header lines
     MOVE H2-HEADER-2 TO PR-PRINT-DATA.
     WRITE PR-PRINT-REC AFTER ADVANCING 2.
     MOVE H3-HEADER-3 TO PR-PRINT-DATA.
     WRITE PR-PRINT-REC AFTER ADVANCING 1.
*    Connect to the specified queue manager.
     CALL WS-MQCONN USING IP-MQM
                          AF-HCONN
                          AF-COMPCODE
                          AF-REASON.
*    Test the output of the connect call.  If the call failed,
*    print an error message showing the completion code and
*    reason code
     IF (AF-COMPCODE NOT = MQCC-OK) THEN
        MOVE 'CONNECT'     TO M-MSG4-TYPE
        MOVE AF-COMPCODE   TO M-MSG4-COMPCODE
        MOVE AF-REASON     TO M-MSG4-REASON
        MOVE M-MESSAGE-4 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC
        MOVE RV-CSQ4-ERROR TO RC-RETURN-CODE
        GO TO A-MAIN-END
     END-IF.
*
*    Initialize the object descriptor (MQOD) control block.
*    (The copy file initializes all the other fields)
      MOVE IP-OBJECT TO MQOD-OBJECTNAME.
*    Initialize the working storage fields required to open
*    the queue
```

```
*      AF-OPTIONS IS SET TO OPEN THE QUEUE FOR INPUT
*      AF-HOBJ     is set by the MQOPEN call and is used by the
*                  MQGET and MQCLOSE calls
*
     MOVE MQOO-INPUT-AS-Q-DEF TO AF-OPTIONS.
*    Open the queue.
     CALL WS-MQOPEN USING AF-HCONN
                          MQOD
                          AF-OPTIONS
                          AF-HOBJ
                          AF-COMPCODE
                          AF-REASON.
*    Test the output of the open call.  If the call failed, print
*    an error message showing the completion code and reason code
     IF (AF-COMPCODE NOT = MQCC-OK) THEN
        MOVE 'OPEN'        TO M-MSG4-TYPE
        MOVE AF-COMPCODE   TO M-MSG4-COMPCODE
        MOVE AF-REASON     TO M-MSG4-REASON
        MOVE M-MESSAGE-4 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC
        MOVE RV-CSQ4-ERROR TO RC-RETURN-CODE
        GO TO A-MAIN-DISCONNECT
     END-IF.
*    SET THE MQMD ATTRIBUTES
     MOVE MQGMO-WAIT TO MQGMO-OPTIONS.
     ADD MQGMO-NO-SYNCPOINT TO MQGMO-OPTIONS.
     MOVE MQWI-UNLIMITED TO MQGMO-WAITINTERVAL.
     PERFORM WITH TEST AFTER
             UNTIL AF-COMPCODE NOT = MQCC-OK
                   OR E-END-OF-JOB
* SET ATTRIBUTES THAT MUST BE RESET EACH TIME
        MOVE MQMI-NONE TO MQMD-MSGID
        MOVE MQCI-NONE TO MQMD-CORRELID
*       PRINT A LINE SAYING WE'RE WAITING FOR WORK
        ACCEPT IT-INPUT-TIME FROM TIME
        MOVE IT-HOURS TO H4-HOURS
        MOVE IT-MINUTES TO H4-MINUTES
        MOVE IT-SECONDS TO H4-SECONDS
        MOVE 'Waiting for Triggering Message' TO
             H4-HEADER-4-DATA
        MOVE H4-HEADER-4 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC AFTER ADVANCING 2
*       Get the next message
        CALL WS-MQGET USING AF-HCONN
                            AF-HOBJ
                            MQMD
                            MQGMO
                            AF-BUFFER-LENGTH
                            AF-MESSAGE-DATA
                            AF-DATA-LENGTH
                            AF-COMPCODE
                            AF-REASON
        IF (AF-COMPCODE = MQCC-OK) THEN
```

```
            IF MQMD-MSGTYPE = MQMT-REPORT AND
               MQMD-FEEDBACK = MQFB-QUIT
               THEN
                  MOVE MQMD-USERIDENTIFIER TO M-USERID
                  MOVE M-MESSAGE-Ø TO PR-PRINT-DATA
                  MOVE EO-TRUE TO E-END-JOB
               ELSE
                  PERFORM PROCESS-MESSAGE
            END-IF
         END-IF
      END-PERFORM.
      IF NOT AF-COMPCODE = MQCC-OK
      THEN
         MOVE 'GET'          TO M-MSG4-TYPE
         MOVE AF-COMPCODE    TO M-MSG4-COMPCODE
         MOVE AF-REASON      TO M-MSG4-REASON
         MOVE M-MESSAGE-4 TO PR-PRINT-DATA
      END-IF.
      WRITE PR-PRINT-REC
* Close the queue
      MOVE MQCO-NONE TO AF-OPTIONS.
      CALL WS-MQCLOSE USING AF-HCONN
                            AF-HOBJ
                            AF-OPTIONS
                            AF-COMPCODE
                            AF-REASON.
*     Test the output of the MQCLOSE call.  If the call failed,
*     print an error message showing the completion and reason code
      IF (AF-COMPCODE NOT = MQCC-OK) THEN
         MOVE 'CLOSE'        TO M-MSG4-TYPE
         MOVE AF-COMPCODE    TO M-MSG4-COMPCODE
         MOVE AF-REASON      TO M-MSG4-REASON
         MOVE M-MESSAGE-4 TO PR-PRINT-DATA
         WRITE PR-PRINT-REC
         MOVE RV-CSQ4-ERROR TO RC-RETURN-CODE
      END-IF.
 A-MAIN-DISCONNECT.
* Disconnect from the queue manager
      CALL WS-MQDISC USING AF-HCONN
                           AF-COMPCODE
                           AF-REASON.
*     Test the output of the disconnect call.  If the call failed,
*     print an error message showing the completion code and
*     reason code
      IF (AF-COMPCODE NOT = MQCC-OK) THEN
         MOVE 'DISCONNECT'   TO M-MSG4-TYPE
         MOVE AF-COMPCODE    TO M-MSG4-COMPCODE
         MOVE AF-REASON      TO M-MSG4-REASON
         MOVE M-MESSAGE-4 TO PR-PRINT-DATA
         MOVE RV-CSQ4-ERROR TO RC-RETURN-CODE
         WRITE PR-PRINT-REC
      END-IF.
 A-MAIN-END.
*     Set the return code
```

```
     MOVE RC-RETURN-CODE TO RETURN-CODE.
*    Close the print file and stop
     CLOSE SYSPRINT.
     STOP RUN.
 PROCESS-MESSAGE SECTION.
*  ------------------------------------------------------------------
*   THIS SECTION IS CALLED WHEN A TRIGGER MESSAGE HAS BEEN READ.
*   IT WILL READ THE TRIGGER MESSAGE TO CREATE THE JCL TO SEND
*   TO THE INTERNAL READER.
*  ------------------------------------------------------------------
*       PRINT A LINE SAYING WE'VE GOT WORK
     ACCEPT IT-INPUT-TIME FROM TIME
     MOVE IT-HOURS TO H4-HOURS
     MOVE IT-MINUTES TO H4-MINUTES
     MOVE IT-SECONDS TO H4-SECONDS
     MOVE 'Received Trigger Message; Generated JCL follows.' TO
             H4-HEADER-4-DATA
        MOVE H4-HEADER-4 TO PR-PRINT-DATA
        WRITE PR-PRINT-REC AFTER ADVANCING 1
     OPEN OUTPUT INTRDR.
*  Now, Grab the fields of the trigger message to construct JCL
* MOVE JOB CARDS TO INTERNAL READER AND OUTPUT IT
     MOVE AF-APPLID TO AA-APPLID-ARRAY.
     IF AA-1-SLASH = '//'
          MOVE AA-APPLID-ARRAY-1 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE AA-APPLID-ARRAY-1 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     IF AA-2-SLASH = '//'
          MOVE AA-APPLID-ARRAY-2 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE AA-APPLID-ARRAY-2 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     IF AA-3-SLASH = '//'
          MOVE AA-APPLID-ARRAY-3 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE AA-APPLID-ARRAY-3 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     IF AA-4-SLASH = '//'
          MOVE AA-APPLID-ARRAY-4 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE AA-APPLID-ARRAY-4 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     MOVE AF-ENVDATA TO EA-ENVDATA-ARRAY.
     IF EA-1-SLASH = '//'
          MOVE EA-ENVDATA-ARRAY-1 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE EA-ENVDATA-ARRAY-1 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
```

```
          WRITE PR-PRINT-REC.
     IF EA-2-SLASH = '//'
          MOVE AF-ENVDATA TO EA-ENVDATA-ARRAY
          MOVE EA-ENVDATA-ARRAY-2 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE EA-ENVDATA-ARRAY-2 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     MOVE AF-USERDATA TO UA-USERDATA-ARRAY.
     IF UA-1-SLASH = '//'
          MOVE UA-USERDATA-ARRAY-1 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE UA-USERDATA-ARRAY-1 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
     IF UA-2-SLASH = '//'
          MOVE UA-USERDATA-ARRAY-2 TO ID-INTRDR-DATA
          WRITE ID-INTRDR-DATA
          MOVE UA-USERDATA-ARRAY-2 TO H5-HEADER-5-DATA
          MOVE H5-HEADER-5 TO PR-PRINT-DATA
          WRITE PR-PRINT-REC.
*  MOVE THE SUBMITTED BY DPKUT200 CARD
     MOVE AC-SUBMITED-BY-CARD TO ID-INTRDR-DATA.
     WRITE ID-INTRDR-DATA.
     MOVE AC-SUBMITED-BY-CARD TO H5-HEADER-5-DATA
     MOVE H5-HEADER-5 TO PR-PRINT-DATA
     WRITE PR-PRINT-REC.
*  Put the JES termination card.
     MOVE AC-EOF-CARD TO ID-INTRDR-DATA.
     WRITE ID-INTRDR-DATA.
     MOVE AC-EOF-CARD TO H5-HEADER-5-DATA
     MOVE H5-HEADER-5 TO PR-PRINT-DATA
     WRITE PR-PRINT-REC.
*  Close the output INTRDR file.
     CLOSE INTRDR.
 PROCESS-MESSAGE-END.
     EXIT.
```

JCL

```
//QBTTRGRT EXEC PGM=DPKUT200,PARM='QKST,BATCH.INIT.Q'
//STEPLIB  DD  DSN=PKMO.LOADLIB,DISP=SHR
//         DD  DSN=SYS2.MQSERIES.SCSQAUTH,DISP=SHR
//         DD  DSN=SYS2.MQSERIES.SCSQLOAD,DISP=SHR
//SYSPRINT DD  SYSOUT=*,
//             DCB=(DSORG=PS,RECFM=VBA,LRECL=133,BLKSIZE=137)
//INTRDR   DD  SYSOUT=(*,INTRDR)
```

Bruce Borchardt
Senior Systems Programmer (USA)

ISPF command tool (part 2)

Last month we considered the reasons for deploying the ISPF command tool shown below in preference to the standard IBM ISPF 3.9 Command Table Utility. This month the EXEC for the command tool is provided in its entirety to complement the panels provided last month. This utility will allow users to manage their command tables more effectively, making it easy to display and modify the active ISPF commands with Version 4.2 of ISPF.

Editor's note: due to a printing error on page 57 of the November edition of MVS Update a spurious copy of the first page of the ISPFCMDS EXEC was printed. Please note this when downloading from the Web and use the following EXEC instead.

ISPFCMDS EXEC

```
/*============================>> REXX <<=================================*/
/*  ISPFCMDS   : This EXEC displays the contents of the currently     */
/*               active command tables. It also enables the user to   */
/*               (temporarily) update all the command tables and to   */
/*               (permanently) save the User or Site table to disk.   */
/*                                                                    */
/*  Externals  : panels ISPFCMDn (n=0,1,3)           (table display) */
/*                      ISPFCMD2                                      */
/*                      ISPFCMD4, ISPFCMD5    (copies of IBM panels ) */
/*                     (ISPUCMX,  ISPUCMXR)   (<---- the IBM panels ) */
/*                      ISPFCMHn (n=1 to 6)          (HELP panels  ) */
/*               msgs   ISRZ002                (standard IBM message) */
/*                                                                    */
/*               Incorporating the (possible) 4 levels of CMD tables  */
/*               that could be active with ISPF 4.2 or later versions */
/*                                                                    */
/*  Version    : 3.8          Last Updated:  August '98              */
/*====================================================================*/
  USRPREF = 'MY'  /* Default User command table prefix, should be the */
                  /* ......... same as the value in ISRCONFG module.  */
                  /*--------------------------------------------------*/
  Address ISPEXEC                          /* Commands -> ISPEXEC     */
  "CONTROL ERRORS RETURN"                  /* Handle return codes here */
  "VGET (ZAPPLID ZUCTPREF ZSCTPREF ZSCTSRCH ZSCREEN) SHARED"
  Do n = 0 to 9  Until rc > 0       /* Create unique table names      */
     tsuf = ZSCREEN||n              /* Up to 10 suffixes per ZSCREEN  */
     cmds_table = 'CMDSTB'||tsuf    /* Table for list of commands     */
     "TBQUERY" cmds_table           /* Check if table already exists  */
```

```
   End
cmdtbs_table = 'CMDTBS'||tsuf     /* Table for cmd table statistics */
Call LOAD_CMDSTABL                 /* Load commands into cmds_table */
CMDSCAN = ''                       /* Display ALL commands          */
Call SETUP_SCAN
Call SORT_CMDSTABL('C')             /* Sort into command order      */
Call SETUP_OVER                     /* Mark overriding commands     */
Call SORT_CMDSTABL('S')             /* Sort back to original order  */
ZCMD = ''
SELCMD = 'RESIZE'                   /* Initial command for panel to */
"CONTROL NONDISPL ENTER"            /* .... ensure no Pop-up Window */
csrrow = Ø
updated = Ø                         /* User has not updated cmds    */
CMDSACT = ''                        /* Intially use brief display   */
MODELIN2 = 'OMIT'                   /* Omit the line from )MODEL    */
tbdispl_rc = Ø                      /* RC from TBDISPL              */
Do tdloop = 1 to 9999,
   While tbdispl_rc < 8             /* RC = 8:  END, RETURN, CAN    */
  /*-----------------------------------------------------------------*/
  /* Display a list of active Commands.                              */
  /*   - This panel has a variable model line so it can show either */
  /*    one or two lines per command depending on the user's choice */
  /*-----------------------------------------------------------------*/
  "TBDISPL" cmds_table "PANEL(ISPFCMDØ) AUTOSEL(NO) CSRROW("csrrow")"
   tbdispl_rc = RC
   If tbdispl_rc > 4 Then           /* User pressed PF3, or ERROR   */
      Leave tdloop                  /* Exiting from ISPFCMDS        */
   /*-----------------------------------------------------------------*/
   /* Process any line commands  (ZTDSELS = no. of selected rows)   */
   /*-----------------------------------------------------------------*/
   Do ZTDSELS                       /* Only if some rows selected.  */
      Select
        When SEL = 'I' Then               /*  Insert a new command */
           Call INSERT_CMD
        When SEL = 'R' Then               /*  Repeat a command     */
           Call REPEAT_CMD
        When SEL = 'D' Then               /*  Delete the command   */
           Call DELETE_CMD
        When SEL = 'U' | SEL = 'E' Then   /*  Update the command   */
           Call UPDATE_CMD
        When SEL = 'V' | SEL = 'B' Then   /*  View a command       */
           Call VIEW_CMD
        When SEL = 'S' | SEL = 'X' Then   /*  Execute the command  */
           Call EXECUTE_CMD
        Otherwise                         /*  Invalid option       */
        End
      If updated Then Do            /* If the user updated the list */
         Call SORT_CMDSTABL('C')
         Call SETUP_OVER            /* ... Mark the overriding cmds */
         Call SORT_CMDSTABL(CUR#SORT)     /* Restore the sort order */
         updated = Ø
```

```
      End
    If ZTDSELS = 1 Then Leave     /* Finished all selected rows   */
    SEL = ''
   "TBDISPL" cmds_table           /* Get next selected row        */
    tbdispl_rc = RC
    If tbdispl_rc > 8 Then        /* Some kind of ERROR           */
       Leave tdloop               /* Exiting from ISPFCMDS        */
    End
/*-------------------------------------------------------------------*/
/* Process primary commands                                          */
/*-------------------------------------------------------------------*/
Parse Upper Var ZCMD cmd cmdparm
Select
 When cmd = '1' Then Do                  /** Switch displays     **/
    If CMDSACT = '' Then CMDSACT = '/'
                    Else CMDSACT = ''
    End
 When cmd = '2' Then                     /** Display table info  **/
    Call DISPLAY_TABLINFO
 When cmd = '3' | cmd = 'SAVE' Then      /** Save user commands  **/
   Call SAVE_USERTABL
 When cmd = 'SAVESITE' Then              /** Save site commands  **/
   Call SAVE_SITETABL
 When cmd = 'NEWTABLE' Then              /** Create user table   **/
   Call CREATE_USERTABL
 When cmd = 'L' Then                     /** Locate a command    **/
    Call LOCATE_ROW
 When cmd = 'SORT' Then                  /** Sort the commands   **/
    CMDSORT = cmdparm
 When cmd = 'REF' Then Do                /** Refresh the list    **/
   Call LOAD_CMDSTABL             /* Recreate cmds_table          */
   Call SORT_CMDSTABL('C')        /* Sort into command order      */
   Call SETUP_OVER                /* Mark overriding commands     */
   Call SORT_CMDSTABL(CUR#SORT)   /* Sort back to current order   */
   Call SETUP_SCAN                /* New scan is required         */
   End
 Otherwise
    csrrow = Ø
 End
/*-------------------------------------------------------------------*/
/* Process any changes to the selected tables or the cmd scan        */
/*-------------------------------------------------------------------*/
Select
 When CUR#TABS ¬= T1','T2','T3','T4 Then
   Call SELECT_TABLES             /* Update CMDSEL vars           */
 When CUR#SCAN ¬= CMDSCAN Then
   Call SETUP_SCAN                /* Setup row scan for TBDISPL   */
 When Left(CUR#SORT,1) ¬= Left(CMDSORT,1) Then
   Call SORT_CMDSTABL(CMDSORT)    /* Sort the displayed commands  */
 When cmd = '' | cmd = '1' | cmd = '2' | cmd = '3' | cmd = 'SAVE' ,
    | cmd = 'SAVESITE' Then Do    /* Maintain top row displayed   */
```

```
        "TBTOP" cmds_table
        "TBSKIP" cmds_table "NUMBER("ZTDTOP")"
        End
      Otherwise NOP
      End
    End                              /* End of tdloop (Do loop)      */
  /*-------------------------------------------------------------------*/
  /* Clean up and exit                                                 */
  /*-------------------------------------------------------------------*/
  "TBEND" cmdtbs_table
  "TBEND" cmds_table                 /* Finished with the tables     */
  If tbdispl_rc > 8 Then Do
     ZERRSM = ''                     /* No short message             */
     ZERRLM   '***  Error: TBDISPL of table' cmds_table',',
              'using panel ISPFCMDØ has FAILED, rc =' tbdispl_rc
     ZERRALRM = 'YES .WINDOW=LRESP'  /* Alarm, user must press ENTER */
     "SETMSG MSG(ISRZØØ2)"           /* Standard IBM message         */
     End
  Return

/*===================================================================*/
/*====                     SUBROUTINES                           ====*/
/*===================================================================*/
/*===================================================================*/
/* Load the temporary CMDS table from the open command tables        */
/* (in the correct order)                                            */
/*-------------------------------------------------------------------*/
LOAD_CMDSTABL:
    CMDSEL = '/'              /* Rows to be selected by the SCAN      */
    CMDOVER = ''              /* Command overrides initially blank    */
    CMDNUM = Ø                /* Command order numbers                */
  /*-------------------------------------------------------------------*/
  /*  Create a new temporary CMDS table                                */
  /*-------------------------------------------------------------------*/
    "TBEND" cmds_table        /* Remove any existing temp table first */
    "TBCREATE" cmds_table ,
       "NAMES(CMDTAB CMDTORD CMDOVER CMDSEL CMDTRUNC CMDNUM",
             "ZCTVERB ZCTTRUNC ZCTDESC, ZCTACT) NOWRITE"
  /*-------------------------------------------------------------------*/
  /*  Copy appl table into cmds_table                                  */
  /*-------------------------------------------------------------------*/
    to_table = cmds_table       /* Table to be copied to             */
    If ZAPPLID ¬= 'ISP' Then Do /* The ISP table is not an appl table */
       from_table = ZAPPLID"CMDS"
       CMDTAB = ZAPPLID
       CMDTORD = 1              /* The command table order           */
       Call COPY_TABLE          /* Copy from_table into cmds_table   */
       If result = Ø
          Then T1 = '/'         /* Indicates the table was copied    */
          Else T1 = ' '         /* Indicates the table was not copied */
```

```
     End
/*-------------------------------------------------------------------*/
/*  Copy User table into cmds_table                                  */
/*-------------------------------------------------------------------*/
  If ZUCTPREF ¬= '' Then Do   /* User table prefix, but only if open*/
     from_table = Strip(ZUCTPREF)"CMDS"
     CMDTAB = ZUCTPREF
     CMDTORD = 2              /* The command table order           */
     Call COPY_TABLE          /* Copy from_table into cmds_table   */
     If result = Ø
        Then T2 = '/'
        Else T2 = ' '
     End
  Else T2 = ' '               /* Set to blank if no user table open */
/*-------------------------------------------------------------------*/
/*  Copy third table into cmds_table                                 */
/*-------------------------------------------------------------------*/
  If ZSCTSRCH = 'A' Then Do
     from_table = "ISPCMDS"                        /* System table */
     CMDTAB = "ISP"
     End
  If ZSCTSRCH = 'B' Then Do
     from_table = Strip(ZSCTPREF)"CMDS"            /* Site table   */
     CMDTAB = ZSCTPREF
     End
  If CMDTAB = '' Then T3 = ''
  Else Do
     CMDTORD = 3              /* The command table order           */
     Call COPY_TABLE          /* Copy from_table into cmds_table   */
     If result = Ø
        Then T3 = '/'
        Else T3 = ' '
     End
/*-------------------------------------------------------------------*/
/*  Copy fourth table into cmds_table                                */
/*-------------------------------------------------------------------*/
  If ZSCTSRCH = 'B' Then Do
     from_table = "ISPCMDS"                        /* System table */
     CMDTAB = "ISP"
     End
  If ZSCTSRCH = 'A' Then Do
     from_table = Strip(ZSCTPREF)"CMDS"            /* Site table   */
     CMDTAB = ZSCTPREF
     End
  If CMDTAB = '' Then T4 = ''
  Else Do
     CMDTORD = 4              /* The command table order           */
     Call COPY_TABLE          /* Copy from_table into cmds_table   */
     If result = Ø
        Then T4 = '/'
```

```
          Else T4 = ' '
       End
    "TBTOP" cmds_table
    Return
/*====================================================================*/
/* Copy a command table into the temporary CMDS table.                */
/* This is called from LOAD_CMDSTABL, once for each open command      */
/* table.  It is also called from SAVE_USERTABL and SAVE_SITETABL.     */
/*--------------------------------------------------------------------*/
COPY_TABLE:
    "TBVCLEAR" from_table
    If rc > 0 Then Return 4          /* Table not open -> return      */
    "TBTOP" from_table
    "TBSKIP" from_table
    skip_rc = rc
    Do While skip_rc = 0
       CMDTRUNC = Copies('-',ZCTTRUNC)
       CMDNUM = CMDNUM + 10          /* Number the rows (for sorting) */
       "TBADD" to_table "MULT(200)"  /* Get storage for 200 rows      */
       "TBSKIP" from_table
       skip_rc = rc
       End
    Return 0

/*====================================================================*/
/* Set up argument for the SCAN in the TBDISPL panel (ISPFCMD0)       */
/*--------------------------------------------------------------------*/
SETUP_SCAN:
    "TBVCLEAR" cmds_table
    If CMDSCAN = '' Then
       ZCTVERB = '*'
    Else If Right(CMDSCAN,1) ¬= '*'
       Then ZCTVERB = CMDSCAN'*'
       Else ZCTVERB = CMDSCAN
    CMDSEL   = '/'
    "TBSARG" cmds_table "NAMECOND(ZCTVERB,EQ,CMDSEL,EQ)"
    Return

/*====================================================================*/
/* Check the correct value is in CMDSEL in every row of cmds_table    */
/*  - the Tx variables are updated by the user and can be either      */
/*    ' ' or '/'.  CMDSEL column is used for the table SCAN.          */
/*--------------------------------------------------------------------*/
SELECT_TABLES:
    "TBTOP" cmds_table
    "TBSKIP" cmds_table
    Do Until rc > 0                   /* rc=8 when it's at the bottom */
       Select
          When CMDTORD = 1 Then
             If CMDSEL ¬= T1 Then Do       /* If the value is not OK  */
                CMDSEL = T1
```

```
                "TBPUT" cmds_table "ORDER"  /* ... update the row     */
                End
            When CMDTORD = 2 Then
               If CMDSEL ¬= T2 Then Do
                  CMDSEL = T2
                  "TBPUT" cmds_table "ORDER"
                  End
            When CMDTORD = 3 Then
               If CMDSEL ¬= T3 Then Do
                  CMDSEL = T3
                  "TBPUT" cmds_table "ORDER"
                  End
            When CMDTORD = 4 Then
               If CMDSEL ¬= T4 Then Do
                  CMDSEL = T4
                  "TBPUT" cmds_table "ORDER"
                  End
            Otherwise
            End
         "TBSKIP" cmds_table
         End
      Return

/*====================================================================*/
/* Locate a command starting with the cmdparm value                   */
/* - if no match:  find the (first) command that should be after      */
/*   the desired one, then position the list on the row before that   */
/*--------------------------------------------------------------------*/
LOCATE_ROW:
      CMDSEL = '/'            /* Only locate from the selected commands */
      If Right(cmdparm,1) ¬= '*'
         Then ZCTVERB = cmdparm'*'
         Else ZCTVERB = cmdparm
      "TBSCAN" cmds_table "NOREAD",
         "ARGLIST(ZCTVERB,CMDSEL) CONDLIST(EQ,EQ) ROWID(csrrow)"

      If RC > Ø Then Do               /* No matching command row found  */
        "TBSCAN" cmds_table "NOREAD",
          "ARGLIST(ZCTVERB,CMDSEL) CONDLIST(GT,EQ)"
        "TBSKIP" cmds_table "NOREAD NUMBER(-1) ROWID(csrrow)"
         ZERRSM = 'Command "'cmdparm'" not found'
         ZERRLM = ''
         ZERRALRM = 'NO'              /* No alarm with the message      */
         "SETMSG MSG(ISRZØØ2)"        /* Standard IBM message           */
         End
      Return

/*====================================================================*/
/* Delete the command from the live command table (& cmds_table)      */
/*--------------------------------------------------------------------*/
DELETE_CMD:
```

```
    del_table = CMDTAB'CMDS'
    "TBTOP" del_table
    "TBSCAN" del_table "ARGLIST(ZCTVERB,ZCTTRUNC,ZCTDESC,ZCTACT)"
    "TBDELETE" del_table
    "TBDELETE" cmds_table
    ZERRSM = 'Command deleted'
    ZERRLM = 'Command' ZCTVERB 'was deleted'
    ZERRALRM = 'NO'                 /* No alarm with the message      */
    "SETMSG MSG(ISRZØØ2)"           /* Standard IBM message           */
    updated = 1                     /* Command list was updated       */
    Return

/*====================================================================*/
/* Copy the ZCTxxx vars into the ZETxxx vars (for the IBM panels)     */
/*--------------------------------------------------------------------*/
COPY_ZCTVARS:
    ZETTRUNC = ZCTTRUNC                          /* Truncation       */
    ZETDESC = Overlay(ZCTDESC,Copies(' ',8Ø))    /* Description      */
    ZETDESC1 = Left(ZETDESC,6Ø)
    ZETDESC2 = Right(ZETDESC,2Ø)
    ZETACT = Overlay(ZCTACT,Copies(' ',24Ø))     /* Command action   */
    ZETACT1  = Left(ZETACT,6Ø)
    ZETACT2  = Substr(ZETACT,61,6Ø)
    ZETACT3  = Substr(ZETACT,121,6Ø)
    ZETACT4  = Right(ZETACT,6Ø)
    Return

/*====================================================================*/
/* Use copy of IBM panel: ISPUCMX to update a command                 */
/* - allow saving the command via PF3 (as is done by IBM)             */
/* - do some validity checking                                        */
/*   IBM state that a command must begin with A-Z, so that is what    */
/*   this tool enforces, however my testing shows that only a few     */
/*   characters are really invalid, as in the following code:         */
/*     vi = Pos(Left(ZETVERB,1),':;Ø123456789=')                      */
/*     If vi > Ø Then vi = 1   /* vi=Ø (valid) , vi=1 (invalid) */    */
/*--------------------------------------------------------------------*/
PANEL_ISPUCMX:
    "CONTROL DISPLAY SAVE"          /* Save the TBDISPL display       */
    "ADDPOP"                        /* Next panel in Pop-Up window    */
    Do Until ZUCMKEY ¬= ''
       "DISPLAY PANEL(ISPFCMD4)"    /* Copy of IBM cmd update panel   */
       If RC = 8 & ZUCMKEY = '' Then       /* user 'PF3' ie END       */
          ZUCMKEY = 'END'           /* ..... allow update to be saved */
       If ZUCMKEY = 'CANCEL' Then Leave    /* user 'CANCEL'           */
       tn = Datatype(ZETTRUNC,'N')    /* Check Trunc is numeric       */
       vc = Datatype(Left(ZETVERB,1),'M') /* Check first char in Verb */
       /* vc = 1  *** uncomment this line to allow ANY command name   */
       vl = Length(ZETVERB)         /* Get length of the command Verb */
       If ¬vc | ¬tn | ,             /* If invalid Verb | Trunc value  */
          vl < 2 Then Do            /* Verb must be 2 - 8 characters. */
```

```
         ZUCMKEY = ''              /* Force redisplay of the panel   */
         ZERRSM = ''
         If ¬tn Then               /* tn=Ø if Trunc not numeric      */
            ZERRLM = '** Truncation value must be numeric **'
         If ¬vc Then               /* vc=Ø if invalid first char     */
            ZERRLM = '** Command name (Verb) must start with',
                     'A - Z **'
         If ZETVERB = '' Then
            ZERRLM = '** Command name (Verb) MUST be specified **'
         If vl = 1 Then
            ZERRLM = '** Command name (Verb) must be at least',
                     '2 characters **'
         ZERRALRM = 'YES'          /* No alarm with the message      */
         "SETMSG MSG(ISRZØØ2)"     /* Standard IBM message           */
         End
      Else                         /* Verb length >= 2               */
         If ZETTRUNC > vl Then     /* If Trunc > Verb length, the    */
            ZETTRUNC = vl          /* .... command entry is invalid  */
      End
   "REMPOP"                        /* Remove the Pop-Up window       */
   Return

/*====================================================================*/
/* Repeat a command (before the selected row)                         */
/*--------------------------------------------------------------------*/
REPEAT_CMD:
   ZERRSM = ''
   ZERRLM = '** Repeat command' ZCTVERB ,
            'in the' CMDTAB'CMDS table **'
   ZERRALRM = 'NO'                 /* no alarm with the message      */
   "SETMSG MSG(ISRZØØ2)"           /* standard IBM message           */
   ZETVERB = ZCTVERB
   Call COPY_ZCTVARS               /* copy ZCTxx vars -> ZETxx vars  */
   Call PANEL_ISPUCMX              /* use ISPUCMX panel for update   */
   If ZUCMKEY = 'END' Then Do      /*** UPDATE ***/
      upd_table = CMDTAB'CMDS'
      "TBTOP" upd_table
      "TBSCAN" upd_table "ARGLIST(ZCTVERB,ZCTTRUNC,ZCTDESC,ZCTACT)"
      ZCTVERB = ZETVERB
      ZCTTRUNC = ZETTRUNC
      ZCTDESC = Strip(ZETDESC1||ZETDESC2)        /* strip the blanks */
      ZCTACT  = Strip(ZETACT1||ZETACT2||ZETACT3||ZETACT4)
      "TBSKIP" upd_table "NOREAD NUMBER(-1)"     /* position cursor  */
      "TBADD" upd_table            /* Insert new row in CMD table    */
      CMDTRUNC = Copies('-',ZCTTRUNC)
      CMDNUM = CMDNUM - 1
      "TBSKIP" cmds_table "NOREAD NUMBER(-1)"
      "TBADD" cmds_table "ORDER"   /* Insert new row in temp table   */
      updated = 1                  /* Command list was updated       */
      End
   Else Do
```

```
       ZERRSM = 'Repeat cancelled'
       ZERRLM = 'No new command was created'
       ZERRALRM = 'NO'              /* No alarm with the message      */
       "SETMSG MSG(ISRZ002)"        /* Standard IBM message           */
       End
    "CONTROL DISPLAY RESTORE"       /* Restore the TBDISPL display    */
    Return

/*====================================================================*/
/* Insert a new command (after the selected row)                      */
/*--------------------------------------------------------------------*/
INSERT_CMD:
    ZERRSM = ''
    ZERRLM = '** Inserting a new command after' ZCTVERB ,
             'in the' CMDTAB'CMDS table **'
    ZERRALRM = 'NO'                 /* No alarm with the message      */
    "SETMSG MSG(ISRZ002)"           /* Standard IBM message           */
    ZETVERB = ''                    /* User must supply command name  */
    Call COPY_ZCTVARS               /* Copy ZCTxx vars -> ZETxx vars  */
    Call PANEL_ISPUCMX              /* Use ISPUCMX panel for update   */
    If ZUCMKEY = 'END' Then Do      /*** UPDATE ***/
       upd_table = CMDTAB'CMDS'
       "TBTOP" upd_table
       "TBSCAN" upd_table "ARGLIST(ZCTVERB,ZCTTRUNC,ZCTDESC,ZCTACT)"
       ZCTVERB = ZETVERB
       ZCTTRUNC = ZETTRUNC
       ZCTDESC = Strip(ZETDESC1||ZETDESC2)        /* strip the blanks */
       ZCTACT  = Strip(ZETACT1||ZETACT2||ZETACT3||ZETACT4)
       "TBADD" upd_table            /* Insert new row in CMDS table   */
       CMDTRUNC = Copies('-',ZCTTRUNC)
       CMDNUM = CMDNUM + 1
       "TBADD" cmds_table "ORDER"   /* Insert new row in temp table   */
       updated = 1                  /* Command list was updated       */
       End
    Else Do
       ZERRSM = 'Insert cancelled'
       ZERRLM = 'No new command was created'
       ZERRALRM = 'NO'              /* No alarm with the message      */
       "SETMSG MSG(ISRZ002)"        /* Standard IBM message           */
       End
    "CONTROL DISPLAY RESTORE"       /* Restore the TBDISPL display    */
    Return

/*====================================================================*/
/* Update an existing command                                         */
/*--------------------------------------------------------------------*/
UPDATE_CMD:
    ZETVERB  = ZCTVERB
    Call COPY_ZCTVARS               /* Copy ZCTxx vars -> ZETxx vars  */
    Call PANEL_ISPUCMX              /* Use ISPUCMX panel for update   */
    If ZUCMKEY = 'END' Then Do      /*** UPDATE ***/
```

```
      upd_table = CMDTAB'CMDS'
      "TBTOP" upd_table
      "TBSCAN" upd_table "ARGLIST(ZCTVERB,ZCTTRUNC,ZCTDESC,ZCTACT)"
      ZCTVERB = ZETVERB
      ZCTTRUNC = ZETTRUNC
      ZCTDESC = Strip(ZETDESC1||ZETDESC2)        /* Strip the blanks */
      ZCTACT  = Strip(ZETACT1||ZETACT2||ZETACT3||ZETACT4)
      "TBPUT" upd_table            /* Replace row in CMDS table      */
      CMDTRUNC = Copies('-',ZCTTRUNC)
      "TBPUT" cmds_table           /* Replace row in temp table      */
      updated = 1                  /* Command list was updated       */
      End
   Else Do
      ZERRSM = 'Update cancelled'
      ZERRLM = ''
      ZERRALRM = 'NO'              /* No alarm with the message      */
      "SETMSG MSG(ISRZØØ2)"        /* Standard IBM message           */
      End
   "CONTROL DISPLAY RESTORE"       /* Restore the TBDISPL display    */
   Return

/*====================================================================*/
/* View the details of a command, using copy of ISPUCMXR panel        */
/*--------------------------------------------------------------------*/
VIEW_CMD:
   ZETVERB  = ZCTVERB
   Call COPY_ZCTVARS               /* Copy ZCTxx vars -> ZETxx vars  */
   ZERRSM = ''
   ZERRLM = '** View of command' ZCTVERB ,
            'in the' CMDTAB'CMDS table **'
   ZERRALRM = 'NO .WINDOW=LN'      /* No alarm, in long noresp window*/
   "SETMSG MSG(ISRZØØ2)"           /* Standard IBM message           */
   "CONTROL DISPLAY SAVE"          /* Save the TBDISPL display       */
   "ADDPOP"                        /* Next panel in Pop-Up window    */
   "DISPLAY PANEL(ISPFCMD5)"       /* Standard IBM cmd display panel */
   "REMPOP"                        /* Remove the Pop-Up window       */
   "CONTROL DISPLAY RESTORE"       /* Restore the TBDISPL display    */
   Return

/*====================================================================*/
/* Execute a command (&ZPARM is taken from the command line)          */
/*--------------------------------------------------------------------*/
EXECUTE_CMD:
   Select
   When ZCTACT = 'PASSTHRU' Then Do
      ZERRSM = ''
      ZERRLM = ZCTVERB 'can't be invoked here,',
               '(because it's a PASSTHRU command)'
      ZERRALRM = 'YES'             /* Alarm with the message         */
     "SETMSG MSG(ISRZØØ2)"         /* Standard IBM message           */
      Return
```

```
      End
   When ZCTACT = '' Then Do
      ZERRSM = ''
      ZERRLM = 'Command:' ZCTVERB 'can't be executed',
               '(because it has no Action)'
      ZERRALRM = 'YES'             /* Alarm with the message         */
     "SETMSG MSG(ISRZØØ2)"         /* Standard IBM message           */
      Return
      End
   When Word(ZCTACT,1) = 'SELECT' &,
        (CMDOVER = '-' | ZTDSELS > 1) Then Do
      "CONTROL DISPLAY SAVE"       /* Save the TBDISPL display       */
      zp = Pos('&ZPARM',ZCTACT)
      If zp > Ø Then               /* Put ZCMD where &ZPARM was      */
         zctact = Left(ZCTACT,zp-1)||ZCMD||Substr(ZCTACT,zp+6)
      Address ISPEXEC zctact       /* EXECUTE the command            */
      ZCMD = ''                    /* Blank out the command line     */
      "CONTROL DISPLAY RESTORE"    /* Restore the TBDISPL display    */
      End
   When Word(ZCTACT,1) = 'ALIAS' Then Do
      SELCMD = Subword(ZCTACT,2),  /* alias ZCTVERB -> ZCMD in panel */
               ZCMD                /* current ZCMD -> &ZPARM action  */
      "CONTROL NONDISPL ENTER"     /* ENTER the command              */
      Return
      End
   Otherwise
      SELCMD = ZCTVERB             /* ZCTVERB -> ZCMD in panel       */
      "CONTROL NONDISPL ENTER"     /* ENTER the command              */
   End
   Return

/*====================================================================*/
/*  Mark which commands override others                               */
/*  1. '*' marks the active commands, '-' marks overridden commands   */
/*  2. The cmds_table has been first sorted into command verb order   */
/*       and then in the order of the commands.                       */
/*  3. The table is processed starting from the bottom, then          */
/*       checking each row above (to see if it is the same verb).     */
/*--------------------------------------------------------------------*/
SETUP_OVER:
   skip_rc = Ø
   last_verb = ''
   "TBBOTTOM" cmds_table              /* Starting at bottom of table */
   Do until skip_rc > Ø
      If last_verb = ZCTVERB Then Do
         CMDOVER = '*'
         "TBPUT" cmds_table "ORDER"          /* Put '*' into the row */
         "TBSKIP" cmds_table                 /* -> Previous row      */
         CMDOVER = '-'
         "TBPUT" cmds_table "ORDER"          /* Put '-' into the row */
```

```
          "TBSKIP" cmds_table "NUMBER(-1)"    /* Return -> next row   */
          End
       Else If updated = 1 & CMDOVER ¬= ' ' Then Do
          CMDOVER = ' '
          "TBPUT" cmds_table "ORDER"          /* Put ' ' into the row */
          End
       last_verb = ZCTVERB              /* Save the name of last verb */
       "TBSKIP" cmds_table "NUMBER(-1)" /* Get next row (working UP)  */
       skip_rc = rc
       End
    Return

/*====================================================================*/
/*  Sort user table into standard sequence                            */
/*--------------------------------------------------------------------*/
SORT_CMDSTABL:
    Arg cmdsort
    Select
      When Left(cmdsort,1) = 'S' Then Do   /* Sort by original order  */
        cmdsort = 'STANDARD ORDER'
        "TBSORT "cmds_table " FIELDS(CMDNUM,N,A)"
        End
      When Left(cmdsort,1) = 'T' Then Do   /* Sort by table and verb  */
        cmdsort = 'TABLE'
        "TBSORT "cmds_table " FIELDS(CMDTORD,N,A,ZCTVERB,C,A)"
        End
      When Left(cmdsort,1) = 'A' Then Do   /* Sort by action and verb */
        cmdsort = 'ACTION'
        "TBSORT "cmds_table " FIELDS(ZCTACT,C,A,ZCTVERB,C,A)"
        End
      When Left(cmdsort,1) = 'D' Then Do   /* Sort by desc and verb   */
        cmdsort = 'DESCRIPTION'
        "TBSORT "cmds_table " FIELDS(ZCTDESC,C,A,ZCTVERB,C,A)"
        End
      Otherwise                            /* Sort by verb and table  */
        cmdsort = 'COMMAND'
        "TBSORT "cmds_table " FIELDS(ZCTVERB,C,A,CMDNUM,N,A)"
      End
    Return

/*====================================================================*/
/* Put table stats into a table and display it                        */
/*--------------------------------------------------------------------*/
DISPLAY_TABLINFO:
    "TBCREATE" cmdtbs_table "NOWRITE SHARE KEYS(CMDTABL)",
       "NAMES(CMDROWS,CMDDSN,CMDDATE,CMDTIME,CMDUSER)"
    tbcreate_rc = rc                /* rc > Ø if table already exists */
    If tabnames = 'TABNAMES' Then Do
       tabnames =  Strip(ZUCTPREF) TB3 TB4
```

```
      If ZAPPLID ¬= 'ISP' Then tabnames = ZAPPLID tabnames
      End
   Do t = 1 to Words(tabnames)
      cmdtabl = Word(tabnames,t)'CMDS'
      "TBGET" cmdtbs_table
      tbget_rc = rc                        /* rc = 8 if no row found */
      "TBSTATS" cmdtabl "ROWCURR(CMDROWS) STATUS1(status1)",
         "UDATE(CMDDATE) UTIME(CMDTIME) USER(CMDUSER)"
      If status1 = 1 Then Do   /* table exists in file ISPTLIB       */
         CMDROWS = Format(CMDROWS)    /* Remove the leading zeros    */
         If tbget_rc = 8 ,     /* No existing row - get dataset name */
            Then CMDDSN = FIND_MEMBER(cmdtabl 'ISPTLIB')
         "TBMOD" cmdtbs_table              /* Update or add the row  */
         End
      End
   old_tbtop = ZTDTOP                      /* Save old top line no.  */
   "TBTOP" cmdtbs_table
   /*-------------------------------------------------------------------*/
   /* Display the table stats panel                                     */
   /*-------------------------------------------------------------------*/
   "TBDISPL" cmdtbs_table "PANEL(ISPFCMD3) AUTOSEL(NO)"
   If rc > 8 Then Do
      ZERRLM   '***  Error: TBDISPL of table' cmdtbs_table',',
               'using panel ISPFCMD3 has FAILED, rc =' rc
      ZERRSM = ''                     /* No short message             */
      ZERRALRM = 'YES .WINDOW=LRESP' /* alarm, user must press ENTER */
      "SETMSG MSG(ISRZØØ2)"           /* Standard IBM message         */
      End
   ZTDTOP = old_tbtop                 /* Restore top line number      */
   Return

/*====================================================================*/
/* Look for 'member' in the 'file' concatenation                      */
/*  - return the name of the first library with that member, and if   */
/*    it is accessed by a LIBDEF put '***' before the dsname          */
/*--------------------------------------------------------------------*/
FIND_MEMBER:
   Parse Upper Arg member file              /* file = ISPTLIB        */
   If member = ZAPPLID'CMDS' Then Do        /* Application cmds      */
      "QLIBDEF" file "TYPE(libtype) ID(libid)"
      If rc = Ø Then Do                     /* a LIBDEF is active    */
         If libtype = 'DATASET' Then Do i = 1 to Words(libid)
            dsn = Word(Strip(libid,b,"'"),i)
            x = Sysdsn("'"dsn"("member")'")
            If x = 'OK' Then Return '***' dsn
            End
         If libtype = 'LIBRARY' Then Do
            x = Listdsi(libid "FILE")
            x = Sysdsn("'"SYSDSNAME"("member")'")
            If x = 'OK' Then Return '***' SYSDSNAME
```

```
            End
         End
      End
   If dsnlist = 'DSNLIST' Then
      Call FIND_DSNLIST(file)     /* Get list of libraries allocated */
   Do i = 1 To Words(dsnlist)
      dsn = Word(dsnlist,i)
      x = Sysdsn("'"dsn"("member")'")         /* look for the member */
      If x = 'OK' Then Return dsn             /* RETURN the DSNAME   */
      End
   Return  '*** not found in ISPTLIB'         /* Member not found    */

/*====================================================================*/
/* Return a list of the datasets in the specified file allocation     */
/*   - this is called only by FIND_MEMBER                             */
/*--------------------------------------------------------------------*/
FIND_DSNLIST:
   Parse Upper Arg file
   Trace O
   x = Outtrap('var.')            /* Trap TSO messages in stem var.  */
   Address TSO "LISTA ST"         /* List dataset allocations in TSO */
   x = Outtrap('OFF')             /* End message trapping            */
   dsnlist = ''
   Do i = 2 To var.Ø By 2         /* Process the trapped messages    */
      parse var var.i dsn .
      If dsn = 'TERMFILE' | dsn = 'NULLFILE' then i = i - 1
      Else Do
         j = i + 1
         Parse Var var.j newdd disp
         If disp ¬= ' ' Then
            ddname = Strip(newdd)     /* Get a new DDNAME            */
         End
      If ddname = file Then           /* If it's the DDNAME we want  */
         dsnlist = dsnlist dsn        /* .... add the dsn to dsnlist */
      Else                            /* ddname ¬= file              */
         If dsnlist ¬= '' Then        /* If dsnlist has been created */
            Return dsnlist
      End
   Return  ddname 'not found'         /* dsnlist not created         */

/*====================================================================*/
/*  Save User table to disk                                           */
/*--------------------------------------------------------------------*/
SAVE_USERTABL:
   If ZUCTPREF = '' Then Do
      ZERRSM = 'No User commands'
      ZERRLM = 'INVALID: there is no User Command table to be saved'
      ZERRALRM = 'YES'                  /* Alarm with the message    */
      "SETMSG MSG(ISRZØØ2)"             /* Standard IBM message      */
      Return
      End
```

```
    from_table = Strip(ZUCTPREF)'CMDS'
    to_table = 'USERCM'||tsuf /* Try to make a unique table name      */
    "TBEND" to_table          /* Remove any existing temp user table  */
    "TBCREATE" to_table ,     /* Create temporary table in WRITE mode */
       "NAMES(ZCTVERB ZCTTRUNC ZCTACT ZCTDESC) WRITE"
    /*-----------------------------------------------------------------*/
    /* Create copy of User Command Table                               */
    /*-----------------------------------------------------------------*/
    Call COPY_TABLE     /* Copy User cmds to temporary table USERCMDT */
    If Result = Ø Then Do
       /*--------------------------------------------------------------*/
       /* Get valid user_library name & allocate it to CMDULIB        */
       /*--------------------------------------------------------------*/
       If CMDSN2 = '' | CMDSN2 = 'CMDSN2' Then
          CMDSN2 = FIND_MEMBER(from_table 'ISPTLIB')
       /*--------------------------------------------------------------*/
       /* Get confirmation before continuing                          */
       /*--------------------------------------------------------------*/
       conf_rc = CONFIRM_SAVE(from_table CMDSN2)
       If conf_rc > Ø Then Do
          "TBEND" to_table
          Return
          End
       Address TSO "ALLOC FI(CMDULIB) DS('"CMDSN2"') SHR REUSE"
       /*--------------------------------------------------------------*/
       /* Save the temporary table, replacing the existing user table */
       /*--------------------------------------------------------------*/
       "TBSAVE" to_table "NAME("from_table") LIBRARY(CMDULIB)"
       If rc = Ø Then Do
          ZERRSM = 'User cmds saved'
          ZERRLM = 'User command table saved in' CMDSN2 'dataset'
          ZERRALRM = 'NO'               /* No alarm with the message */
          End
       Else Do
          ZERRSM = 'cmds not saved'
          ZERRLM = 'Unable to save cmds in' CMDSN2 'dataset, rc='rc
          ZERRALRM = 'YES'              /* Alarm with the message    */
          End
       "SETMSG MSG(ISRZØØ2)"            /* Standard IBM message      */
       End
    "TBEND" to_table          /* Finished with the temp user table   */
    Return

/*=================================================================*/
/*  Save Site table to disk                                        */
/*-----------------------------------------------------------------*/
SAVE_SITETABL:
    If ZSCTPREF = '' Then Do
       ZERRSM = 'No Site commands'
       ZERRLM = 'INVALID: there is no Site Command table to be saved'
```

```
      ZERRALRM = 'YES'                  /* Alarm with the message    */
      "SETMSG MSG(ISRZØØ2)"             /* Standard IBM message      */
      Return
      End
   from_table = Strip(ZSCTPREF)'CMDS'
   to_table = 'SITECM'||tsuf /* Try to make a unique table name      */
   "TBEND" to_table          /* Remove any existing temp user table  */
   "TBCREATE" to_table ,     /* Create temporary table in WRITE mode */
      "NAMES(ZCTVERB ZCTTRUNC ZCTACT ZCTDESC) WRITE"

   /*-----------------------------------------------------------------*/
   /* Create copy of Site Command Table                               */
   /*-----------------------------------------------------------------*/
   Call COPY_TABLE     /* Copy Site cmds to temporary table SITECMDT */
   If Result = Ø Then Do
      /*--------------------------------------------------------------*/
      /* Get valid user_library name & allocate it to CMDSLIB         */
      /*--------------------------------------------------------------*/
      If CMDSN3 = '' | CMDSN3 = 'CMDSN3' Then
         CMDSN3 = FIND_MEMBER(from_table 'ISPTLIB')
      /*--------------------------------------------------------------*/
      /* Get confirmation before continuing                           */
      /*--------------------------------------------------------------*/
      conf_rc = CONFIRM_SAVE(from_table CMDSN3)
      If conf_rc > Ø Then Do
         "TBEND" to_table
         Return
         End
      Address TSO "ALLOC FI(CMDSLIB) DS('"CMDSN3"') SHR REUSE"
      /*--------------------------------------------------------------*/
      /* Save the temp table, replacing the existing site table       */
      /*--------------------------------------------------------------*/
      "TBSAVE" to_table "NAME("from_table") LIBRARY(CMDSLIB)"
      If rc = Ø Then Do
         ZERRSM = 'Site cmds saved'
         ZERRLM = 'Site command table saved in' CMDSN3 'dataset'
         ZERRALRM = 'NO'                /* No alarm with the message */
         End
      Else Do
         ZERRSM = 'Cmds not saved'
         ZERRLM = 'Unable to save cmds in' CMDSN3 'dataset, rc='rc
         ZERRALRM = 'YES'               /* Alarm with the message    */
         End
      "SETMSG MSG(ISRZØØ2)"             /* Standard IBM message      */
      End
   "TBEND" to_table           /* Finished with the temp site table   */
   Return

/*=================================================================*/
/*  User to Confirm that the new table should be written to disk   */
/*    - invoked from SAVE_USERTABL, SAVE_SITETABL & CREATE_USERTABL */
/*-----------------------------------------------------------------*/
```

```
CONFIRM_SAVE:
    Arg TABNAME LIBNAME            /* Get table name and library name */
    TSTAT = Sysdsn("'"LIBNAME"("TABNAME")'")      /* Already exists?  */
    "ADDPOP"
    "DISPLAY PANEL(ISPFCMD2)"        /* Ask user to confirm or cancel */
    conf_rc = rc          /* User replies: ENTER (rc=Ø) or END (rc=8) */
    "REMPOP"
    If conf_rc > Ø Then Do
       ZERRSM = 'SAVE Cancelled'
       ZERRLM = 'SAVE of Command Table' TABNAME 'was CANCELLED'
       ZERRALRM = 'YES'                   /* Alarm with the message    */
       "SETMSG MSG(ISRZØØ2)"              /* Standard IBM message      */
       End
    Return conf_rc

/*================================================================*/
/*  Create a new User Command Table                               */
/*----------------------------------------------------------------*/
CREATE_USERTABL:
    If ZUCTPREF = '' & USRPREF = '' Then Do
       ZERRSM = ''
       ZERRLM = 'INVALID: User Command Tables are not in use'
       ZERRALRM = 'YES'                   /* Alarm with the message    */
       "SETMSG MSG(ISRZØØ2)"              /* Standard IBM message      */
       Return
       End
    If ZUCTPREF ¬= '' ,     /* ZUCTPREF = user table prefix, iff open */
       Then USRTABL = Strip(ZUCTPREF)"CMDS"
       Else USRTABL = USRPREF"CMDS"      /* Default name               */
    /*----------------------------------------------------------------*/
    /* Put list of ISPTLIB libraries  in table TLIBS                  */
    /*----------------------------------------------------------------*/
    If dsnlist = 'DSNLIST' Then
       Call FIND_DSNLIST('ISPTLIB')  /* Get list of ISPTLIB libraries */
    "TBCREATE TLIBS NAMES(CMDSN1)"   /* Table of libraries in ISPTLIB */
    Do d = 1 to Words(dsnlist)
       CMDSN1 = Word(dsnlist,d)
       "TBADD TLIBS"                 /* Populate the table            */
       End
    "TBTOP TLIBS"
    /*----------------------------------------------------------------*/
    /* Display panel asking user to identify the target library       */
    /*----------------------------------------------------------------*/
    ZTDSELS = Ø
    Do until (rc > Ø | ZTDSELS > Ø)
       "TBDISPL TLIBS PANEL(ISPFCMD1)",    /* Display list of ISPTLIB */
          "AUTOSEL(NO)"                    /* No auto-select of rows  */
       If ZTDSELS > Ø Then Do        /* If user selects a library ... */
          /*-------------------------------------------------------*/
          /* Get confirmation before continuing                    */
          /*-------------------------------------------------------*/
```

```
   conf_rc = CONFIRM_SAVE(USRTABL CMDSN1)
   If conf_rc > Ø Then Do
      "TBEND TLIBS"
      Return
      End
   /* Allocate library to temp ddname */
   Address TSO "ALLOC FI(CMDULIB) DS('"CMDSN1"') SHR REUSE"
   /*-------------------------------------------------------------*/
   /* Create temp table & add 1 dummy row                         */
   /*-------------------------------------------------------------*/
   tu_table = 'USERCM'||tsuf    /* Try to make a unique tbname */
   "TBEND" tu_table       /* Ensure no temp table already open */
   "TBCREATE" tu_table,   /* Create new user table, WRITE mode */
      "NAMES(ZCTVERB ZCTTRUNC ZCTACT ZCTDESC) WRITE"
   ZCTVERB  = '??'
   ZCTTRUNC = 'Ø'
   ZCTACT   = ''              /* Null action = 'do nothing'    */
   ZCTDESC  = '<<< Dummy command for you to modify >>>'
   "TBADD" tu_table           /* Add row to temp table         */
   /*-------------------------------------------------------------*/
   /* Write new user table into the user table library            */
   /*-------------------------------------------------------------*/
   "TBSAVE" tu_table ,        /* Pad 5ØØ% to leave extra space */
      "NAME("USRTABL") LIBRARY(CMDULIB) PAD(5ØØ)"
   If rc = Ø Then Do
      If ZUCTPREF ¬= '' Then Do   /* User table already exists */
         Call UPDATE_USERTABL     /* ... update it             */
         txt = 'Now'              /* Text in ZERRLM message    */
         End
      Else                        /* No existing User table    */
         txt = 'Next you should EXIT from ISPF,',
               'then start ISPF again and your',
               'new command table will be active.  Then'
      ZERRLM = 'A new User Command Table' USRTABL 'was',
               'created in' CMDSN1 'library. ' txt ,
               'use this "ISPF Commands" tool to',
               'update your User Commands table. ',
               'It contains a dummy command for you to modify,',
               'and you can add extra commands too. ',
               'Then use option 3 to SAVE your new commands.'
      ZERRSM = ''
      End
   Else Do
      ZERRSM = 'Table not created'
      ZERRLM = 'Unable to save cmds in' CMDSN1 'dataset, rc='rc
      End
   ZERRALRM = 'YES'               /* Alarm with the message    */
   "SETMSG MSG(ISRZØØ2)"          /* Standard IBM message      */
   "TBEND" tu_table               /* Finished with temp table  */
   End
End
```

```
      "TBEND TLIBS"
    Return

/*=============================================================================*/
/*  Update User Command Table                                                  */
/*  - update existing User table to have only the dummy cmd                    */
/*  - this is only called from CREATE_USERTABL                                 */
/*-----------------------------------------------------------------------------*/
UPDATE_USERTABL:
    "TBTOP" USRTABL
    Do until RC > Ø                              /* Delete all the rows */
       "TBSKIP" USRTABL "NOREAD"
       "TBDELETE" USRTABL
       End
    "TBADD" USRTABL                              /* Add the dummy cmd   */
    /*-------------------------------------------------------------------*/
    /* Update current table of commands too                              */
    /*-------------------------------------------------------------------*/
    If Left(CUR#SORT,1) ¬= 'S' Then   /* If sort order not standard - */
       Call SORT_CMDSTABL('S')        /* Sort back to original order  */
    "TBTOP" cmds_table
    CMDTAB = USRPREF
    "TBSCAN" cmds_table "ARGLIST(CMDTAB) NOREAD"
    Do while RC = Ø                  /* Deleting all the User rows   */
       "TBDELETE" cmds_table
       "TBSCAN" cmds_table "ARGLIST(CMDTAB) NOREAD"
       End
    CMDTORD = 3                   /* No. of the table after User cmds */
    "TBSCAN" cmds_table "ARGLIST(CMDTORD) CONDLIST(GE)"
    CMDNUM = CMDNUM - 5             /* Assumes LE 4 inserted cmds     */
    "TBSKIP" cmds_table "NUMBER(-1) NOREAD"
    CMDTAB = USRPREF
    CMDTORD = 2                     /* User cmds are always 2nd table */
    CMDOVER = ''
    CMDSEL = '/'
    "TBGET" tu_table                /* Get row from temp User table   */
    "TBADD" cmds_table "ORDER"      /* Add the dummy cmd              */
    /*-------------------------------------------------------------------*/
    /* Select (only) the User table upon return to the command list      */
    /*-------------------------------------------------------------------*/
    T1 = ''
    T2 = '/'
    T3 = ''
    T4 = ''
    CMDSCAN = ''               /* Show ALL commands in the User table */
    Return
```

Ron Brown
Systems Programmer (Germany)

Beyond Software has announced its EnterpriseWeb Legacy Application Server (LASER), described as both a Web server and an applications server for legacy applications which run under MVS/ESA or OS/390 platforms.

Its functions are similar to other Web servers, but were built specifically to facilitate direct access to mainframe applications and data. Simple transactions can be enabled in a few lines of code. OS/390 applications can be accessed, and transactions run, with a point and click of a Web browser via any desktop machine.

LASER comes with a software developer's kit for building business logic and wrappers around existing applications built in CICS, IMS, TSO, and DB2 using languages like COBOL, REXX, Nomad2, Adabas, SAS, FOCUS, and 370 Assembler.

It is being aimed at enabling mainframes and programming staff to play an integral role in corporate intranets, extranets, and e-commerce. It obviates the need for dedicated programmers, middle tier servers, support people, and Web developers in order to get enterprise-class Web connectivity to its CICS, IMS, and DB2 applications.

For further information contact:
Beyond Software, 1040 East Brokaw Road, San Jose, CA 95181, USA.
Tel: (408) 436 5900
Fax: (408) 441 7226

* * *

IBM has announced its VisualAge for Java, Enterprise Edition for OS/390. The optional compiler feature can be used in conjunction with the run-time feature to develop compiled and bound Java programs. The run-time feature is required to execute fully-bound Java programs.

The compiler/binder statically compiles Java bytecodes directly into native object code in the same manner as traditional compilers for C/C++, COBOL, and FORTRAN. The Enterprise ToolKit for OS/390 (ET/390) static compilation occurs only once.

Also, the compiler/binder binds the code into an executable or dynamic link library (DLL) that can be run in the OS/390 shell or under the CICS Transaction Server for OS/390.

With export and remote bind, class files can be sent from the workstation to OS/390 for final compilation and binding. ET/390's remote capabilities allow OS/390 system and error messages to be reported at the workstation's VisualAge console.

The debugger can be used remotely on an NT 4.0 client to debug Java programs running in an OS/390 environment. On the OS/390, debug options include interpreted programs running in the JVM and compiled and bound Java programs running natively on OS/390, either in the OS/390 Unix environment or under CICS.

Contact your local IBM representative for further information.

* * *