173

MVS update

February 2001

In this issue

MVS Update

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

North American office
Xephon/QNA
PO Box 350100,
Westminster, CO 80035-0100
USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions

Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Editor
Jaime Kaminski

Disclaimer

MVS Update on-line

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Updating the LLA from a batch program

OVERVIEW

Refreshing the LLA can be a difficult process to manage at times, in particular if it contains user libraries that need to be refreshed regularly. Unless the entire LLA address space is bounced, any change needs to be specified in SYS*x*.PARMLIB. This is easy enough, except that users do not normally have access to this dataset, meaning that the systems programmer (not to mention the change manager!) has to be involved. The LLACOPY macro can be used to overcome this problem but, as is the case with most good things in OS/390, it requires the skills of an Assembler programmer to implement.

There are three ways to refresh the LLA. These include adding a library to the LLA list in PARMLIB and bouncing the entire started task, selectively adding, removing, or refreshing libraries by using the CSVLLAxx PARMLIB members, or by calling the LLACOPY macro. This program is an example of making use of the macro to do a REFRESH for a library that is already in the LLA. The user has the option to do any of the following:

- REFRESH an entire library.
- REFRESH one or more members of a library.
- Tell the LLA that one or more members have been deleted from the library and that they should be removed from the LLA's directory.

The LLACOPY macro is the only documented API in the LLA service. It works very simply – all it requires is that the library to be refreshed is allocated and that register 1 is pointing at a BLDL list of the library in memory. The BLDL list is effectively a list telling LLA which members to refresh. Using the macro has an underlying 'flaw' though – if a member had been deleted from the library on the disk, a BLDL of the library does not contain the member name. This means that, when the LLACOPY macro is issued, LLA is not 'told' to refresh the deleted member and it remains in the LLA / VLF directories. The name has to be specifically included in the list. This is important to remember when this utility is used because it is the only way to indicate to LLA that a member has been deleted. Our list of examples on how to use this program includes an example on how to delete a member in the LLA.

The program requires a DD-statement by the name of LLADSN to point to the dataset that needs to be refreshed (the user must have at least READ access to the dataset). Specifying the members to be updated can be done in one of two ways – either though a SYSIN DD statement or, in the case of a single or all (but not just a few) members, by means of a passed parameter in the format ,'PARM='*' '. Examples of calling the utility are shown below:

```
//S1 EXEC PGM=LLABATCH,MEMBER='*'       -Refreshes all members
//LLADSN  DD     DSN=dataset.to.be.refreshed,DISP=SHR

//S1 EXEC PGM=LLABATCH,MEMBER='memname' -Refreshes one member
//LLADSN  DD     DSN=dataset.to.be.refreshed,DISP=SHR

//S1 EXEC PGM=LLABATCH
//LLADSN  DD     DSN=dataset.to.be.refreshed,DISP=SHR
//SYSIN   DD *                           -Refreshes all members
*
//

//S1 EXEC PGM=LLABATCH
//LLADSN  DD     DSN=dataset.to.be.refreshed,DISP=SHR
//SYSIN   DD *
MEMØ1              -Refreshes members MEMØ1,
MEMØ2              MEMØ2 and MEMØ3
MEMØ3
//

//S1 EXEC PGM=LLABATCH
//LLADSN  DD     DSN=dataset.to.be.refreshed,DISP=SHR
//SYSIN   DD *
MEMØ1               -Refreshes members MEMØ1
MEMØ2(D)           and MEMØ4 and deletes
MEMØ3(D)           members MEMØ2 and MEMØ3
MEMØ4
//
```

Note that the program does not support the option to refresh all members and also delete members in the same step. This can be done in two steps, alternatively the option can be coded into the program by those keen enough to do so.

LLABATCH

```
**********************************************************************
* This program will refresh one or more (or all) members of the data
* set allocated to LLADSN. A single member name or "*" (all members)
* can be specified with a JCL PARM. A single member, a list
* of members or "*" (all members) can be specified as parameters in
* a SYSIN file. If a SYSIN card is specified, it takes preference
* over a PARM= statement on the EXEC card.
*
* The program will also refresh LLA for one or more deleted members.
* This is requested by adding (D) at the back of a member name. This
* is however only supported for the SYSIN DD specification, not when
* the name is passed as a parmeter.
*
* If a member is specified (without (D)) but does not exist, an error is
* given but the program still attempts to refresh the other members.
*
* In order to speed up the call to LLA, only one LLACOPY macro is ever
* issued. This is the sequence of events:
*
* 1) Build a sorted list of the requested member(s) to be refreshed
* 2) Do a BLDL to get a list of all members in the PDS
* 3) Allocate a new list and merge matching entries from 1) and 2)
* 4) Issue LLACOPY for the new list established in 3)
*
*          MODE:     Program runs authorized, is non-reentrant
*         INPUT:     Member name or "*" to indicate all members,
*                    ignored if a SYSIN DD-statement is specified.
*        OUTPUT:     None
*         AMODE:     31
*         RMODE:     24
* Caller's mode:     Any (This program does a BAKR / PR)
*Error messages:     Messages related to invalid/ incomplete parms
*                    Messages related to refresh failures
*Called routns :     None
*      DD-cards:     SYSIN DD-card optional. Contains list of members
*                    to refresh or "*" to refresh all members.
*                    (D) at back of member name indicates member has
*                    been deleted from dataset and should be refreshed
*  Special regs:     None
*                    R12= Base register
*                    R13= Pointer to getmained area and save areas
LLABATCH CSECT
LLABATCH AMODE 31
LLABATCH RMODE 24
         BAKR  R14,Ø              .Save Caller's Status
         BALR  R12,Ø
         USING LOAD,12
*********************************************************************
```

```
*        Main driver routine
**********************************************************************
LOAD     L     R4,Ø(R1)            .Ptr to compressed command text addr
STORAGE  L     R3,STORSIZE
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY
         LR    R2,R1                .Point to getmained area
         XR    R9,R9
         XR    R8,R8
         MVCL  R2,R8               .Propagate binary zeros
         USING STORAREA,R1
         ST    R13,SAVEAREA+4
         LR    R13,R1
         DROP  R1
         USING STORAREA,R13        .Addressability to getmained area
         BAS   R14,GETPARM         .Go get parm or SYSIN
         BAS   R14,RFRSHDSN        .Go refresh the dataset in LLA
FREETABL L     R2,TABSTART         .Where the table starts
         LTR   R2,R2               .Was it allocated?
         BZ    FREEPLST            .No
         L     R3,MEMAREA          .Load size of table to allocate
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
*
FREEPLST L     R2,PARTLIST         .Where the table starts
         LTR   R2,R2               .Was it allocated?
         BZ    FREEWRKA            .No
         L     R3,PARTSIZE         .Load size of table to allocate
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
*
FREEWRKA L     R4,RETCODE          .Pick up return code
         L     R3,STORSIZE         .Size of area to free
         LR    R2,R13              .Address of area to free
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
         LR    R15,R4              .Copy return code
TOCALLER PR    ,                   .=>Caller
         DS    ØD                  .Align
**********************************************************************
*        This routine reads the parm or the SYSIN DD statement
**********************************************************************
GETPARM  BAKR  R14,Ø
RDJFCB1  RDJFCB SYSIN
         LTR   R15,R15             .Did user code a SYSIN card?
         BNZ   CHKPARM             .No, look at input parameters
*
SYSINOK  OPEN  SYSIN
         TM    SYSIN+48,X'1Ø'      .Did the file open?
         BO    SYSINGM             .Yes
         WTO  'LLABATCH(E): -Unexpected error on SYSIN file',ROUTCDE=11
         ABEND ØØØ1
*
SYSINGM  L     R3,MEMAREA          .Load size of table to allocate
         STORAGE OBTAIN,LENGTH=(3),LOC=BELOW
         LTR   R15,R15             .Did we get the storage?
         BZ    LBL                  .Yes
NOSTOR   WTO   'LLABATCH(E): -Could not obtain storage',ROUTCDE=11
         ABEND ØØØ2
*
LBL      LR    R4,R1                .Start of Table
         ST    R4,TABSTART          .The size of the table
         ST    R3,TABSIZE           .The size of the table
         XR    R2,R2                .Clear the counter
SYSINLP  GET   SYSIN
         CLI   Ø(R1),C'*'           .All members?
         BNE   MOVENAME             .Yes
         OI    ALLMEMIN,YES         .Set all-member flag
         B     SYSINLP              .Go set flag & get out
MOVENAME MVC   Ø(8,R4),Ø(R1)        .Move the name into the table
         BAS   R14,DELCHECK         .Go see if the member is to be dlted
         LA    R4,9(R4)             .Point to next slot in table
         LA    R2,1(R2)             .Bump up counter
         C     R2,MAXMEMS           .Have we reached the max?
         BNH   SYSINLPX             .No
         WTO  'LLABATCH(E): -Only 2ØØØØ member names allowed in SYSIN',X
               ROUTCDE=11
         ABEND ØØØ3
*
SYSINLPX B     SYSINLP              .Do for all SYSIN cards

ENDSYSIN CLOSE SYSIN
CHK#IN   LTR   R2,R2                .Did we get any cards?
         BNZ   NUMMSG               .Yes
         TM    ALLMEMIN,YES         .User ONLY asked for all members, OK
         BO    GETPARMX             .Get out
         WTO  'LLABATCH(E): -SYSIN card but no member names specified',X
               ROUTCDE=11
         ABEND ØØØ4
*
NUMMSG   ST    R2,NUMSYSIN          .Remember number of table entries
         TM    ALLMEMIN,YES         .Did user also ask for all (*)?
         BNO   GOSORT
         WTO  'LLABATCH(E): -Conflicting request to do all(*) members  X
               and selective list of members',ROUTCDE=11
         ABEND ØØØ5
*
GOSORT   BAS   R14,SORTTAB          .LLACOPY requires sorted list
         B     GETPARMX             .Get out
CHKPARM  CLC   Ø(2,R4),=H'Ø'        .Has a parm been passed?
         BNE   PARMPASS             .Yes
         WTO  'LLABATCH(E): -No SYSIN DD-card or parameters',ROUTCDE=11
         ABEND ØØØ6
*
PARMPASS XR    R2,R2
         ICM   R2,3,Ø(R4)           .Length of passed parm
         LA    R4,2(R4)             .Point to start of data
PARMASTR CLI   Ø(R4),C'*'           .All members?
```

```
         BNE   SELECT              .No
         OI    ALLMEMIN,YES        .Yes, set flag
         B     GETPARMX            .Don't scan further
SELECT   WTO   'LLABATCH(I): -Selection from input parm',             X
               ROUTCDE=11
         CH    R2,=H'8'            .Maximum length of member name
         BNH   REDUCE
         WTO   'LLABATCH(W): -Only 1 member taken from PARM card',     X
               ROUTCDE=11
         LA    R15,4
         ST    R15,RC
         OC    RETCODE,RC
         LA    R2,8                .Reduce the length to 8
REDUCE   BCTR  R2,Ø                .Correct the length
         MVC   MEMNAME,=8X'4Ø'     .Make blanks in case length< 8 bytes
         EX    R2,MOVEINST         .Update the move instruction
         OI    ONEMEM,YES          .Set flag
         B     GETPARMX
MOVEINST MVC   MEMNAME(Ø),Ø(R4)    .Pick up the member name from parm
GETPARMX PR
*********************************************************************
*        This routine scans for "(D)" in the member name
*********************************************************************
DELCHECK BAKR  R14,Ø
         LA    R2,1(R1)            .Start scanning from second byte
         LA    R3,77               .Max # bytes to scan
DELLOOP  CLC   Ø(3,R2),=C'(D)'     .Does it contain delete option?
         BE    DELFOUND            .Yes
         LA    R2,1(R2)            .Bump up pointer
         BCT   R3,DELLOOP          .Scan entire card
         B     DELCHECX            .Not found, get out
DELFOUND MVC   Ø(8,R4),=8C' '      .Clear the name
         MVI   8(R4),C'D'          .Move a "D" into the 8th byte
         SR    R2,R1               .Offset where we found "(D)"
         BCTR  R2,Ø                .Correct the length for MVC
         STC   R2,MVCNAME+1        .Update MVC instruction
MVCNAME  MVC   Ø(Ø,R4),Ø(R1)       .Move the name back into field
DELCHECX PR
*********************************************************************
*        This routine sorts the table containing the member names
*********************************************************************
SORTTAB  BAKR  R14,Ø
         CLC   NUMSYSIN,=F'1'      .Only one entry?
         BNH   SORTTABX            .Yes, no need to sort
         L     R1,NUMSYSIN         .Number of entries in table
         LR    R2,R1
         BCTR  R2,Ø
         XR    R3,R3               .Outer loop counter
OUTLOOP  EQU   *
         XR    R4,R4               .Inner loop counter
INLOOP   EQU   *
         LR    R5,R3
         MH    R5,=H'9'            .Length of an entry
         A     R5,TABSTART
         LR    R6,R4
         MH    R6,=H'9'
         A     R6,TABSTART
         CLC   Ø(9,R5),Ø(R6)       .In correct sequence?
         BNL   LOOPEND             .Yes
         MVC   TEMPNAME,Ø(R5)
         MVC   Ø(9,R5),Ø(R6)       .Swap the 2 names
         MVC   Ø(9,R6),TEMPNAME
LOOPEND  LA    R4,1(R4)
         CR    R4,R2
         BL    INLOOP
         LA    R3,1(R3)
         CR    R3,R1
         BL    OUTLOOP
SORTTABX PR
**********************************************************************
*        This routine opens the data set, calls LLACOPY and closes
*        the dataset.
**********************************************************************
RFRSHDSN BAKR  R14,Ø
RDJFCB2  RDJFCB LLADSN
         LTR   R15,R15             .Is the DD-statement there?
         BZ    MOVEDSN             .Yes
         WTO   'LLABATCH(E): -LLADSN DD-statement missing',ROUTCDE=11
         ABEND ØØØ8
*
MOVEDSN  EQU   *
         USING INFMJFCB,JFCB
         MVC   DSNAME,JFCBDSNM     .Pick up the dataset name
OPLLADSN OPEN  LLADSN
         TM    LLADSN+48,X'1Ø'     .Did the file open?
         BO    OPENOK              .Yes
         WTO   'LLABATCH(E): -Could not open LLADSN',ROUTCDE=11
         ABEND ØØØ9,DUMP
*
OPENOK   BAS   R14,BUILDLST        .Read directory to get member names
         LTR   R15,R15             .Did it work?
         BZ    RESETMD1            .Yes, dataset is ready for update
NOLIST   WTO   'LLABATCH(E): -Could not do a BLDL for the dataset',   X
               ROUTCDE=11
         ABEND ØØ1Ø
*
RESETMD1 EQU   *
CHKALL   TM    ALLMEMIN,YES        .Are we doing all members?
         BNO   PARTIAL             .No, partial refresh only
         LA    R3,BLDLLIST         .Yes, use entire list
         B     REFRESH
PARTIAL  WTO   'LLABATCH(I): -Partial refresh to be done',ROUTCDE=11
GOBLDLST BAS   R14,PARTLBLD        .Go build a partial list
         L     R3,PARTLIST         .Address of the partial list
         USING PRTDSECT,R3         .Addressability to partial list
         CLC   PARTNUM,=H'Ø'       .Any qualifying entries?
         BNZ   REFRESH             .Yes
         WTO 'LLABATCH(W): -No members selected for LLACOPY',ROUTCDE=11
         LA    R15,12
         ST    R15,RC
         OC    RETCODE,RC
         B     CLOSDIR             .Get out
REFRESH  BAS   R14,LLACPMAC        .Issue LLACOPY macro
         LTR   R15,R15             .Successful?
         BZ    GIVEOKMS            .Yes
NOTOKMS  LR    R2,R15              .Preserve our return code
         MVC   WTOAREA(ERWTOLEN),ERWTOMS
         CVD   R15,DOUBLE          .The return code from LLACOPY
         UNPK  DOUBLE(4),DOUBLE+6(2)
         OI    DOUBLE+3,X'FØ'      .Make printable
         MVC   WTOAREA+5Ø(4),DOUBLE
         CVD   RØ,DOUBLE           .The reason code from LLABATCH
         UNPK  DOUBLE(4),DOUBLE+6(2)
         OI    DOUBLE+3,X'FØ'       .Make printable
         MVC   WTOAREA+63(4),DOUBLE
         LA    R1,WTOAREA
         WTO   MF=(E,(1)),ROUTCDE=11
         LR    R15,R2              .Reload the return code
         B     CLOSDIR             .Get out
GIVEOKMS EQU   *
         MVC   WTOAREA(OKWTOLEN),OKWTOMS
         MVC   WTOAREA+48(44),DSNAME
         LA    R1,WTOAREA
         WTO   MF=(E,(1)),ROUTCDE=11
CLOSDIR  ST    R15,RC              .Return code already > Ø?
         OC    RETCODE,RC          .Yes
CLOSIT   CLOSE LLADSN
RFRSHDSX PR
*********************************************************************
*        This routine builds a list of all the members in the PDS
*********************************************************************
BUILDLST BAKR  R14,Ø
         LA    R7,MEMLIST          .Point to start of member list
         XR    R8,R8               .Number-of-members counter
READDIR  LA    R2,LLADSN           .Get next block
         GET   (2)                 .Get next block
         LR    R4,R1               .Starting address of input data
         LA    R4,2(R4)            .First member name = offset 2
         LH    R2,Ø(R1)            .Number of active bytes in block
         SH    R2,=H'2'            .Subtract length field
MEMLOOP  CLC   Ø(8,R4),=8X'FF'     .Is this the last name?
         BE    ENDOFLST            .Yes
         CLI   Ø(R4),X'ØØ'         .Last name in the block?
         BNH   READDIR             .Yes, it is - go read directory
         MVC   Ø(8,R7),Ø(R4)
         LA    R7,12(R7)           .Bump up the list pointer
         LA    R8,1(R8)            .Count number of entries
         C     R8,=F'65535'        .Max number of members allowed
         BNH   CALCOFST            .Calculate offset
         WTO   'LLABATCH(E): -Dataset contains more than 65535 members,X
               dynamic refresh not supported',ROUTCDE=11
         LA    R15,8
         ST    R15,RC
         OC    RETCODE,RC
         B     BUILDLSX            .Get out
*
* Dir entry: XXXXXXXX___Y + No of bytes.   XXXXXXXX = Name
*           |__________________________|    Y X 2 = No of bytes
*
CALCOFST IC    R6,11(R4)           .Load y into reg 6
         SLL   R6,27               .Push other bits out of reg 6
         SRL   R6,26               .Move back + multiply by 2
         LA    R6,12(R6)           .Add fixed 12 bytes of dir entry
         SR    R2,R6
         AR    R4,R6
         LTR   R2,R2               .Still bytes left?
         BNP   READDIR             .Go read next directory block
         B     MEMLOOP             .Get next member name in this block
ENDOFLST STH   R8,NUMENTRY         .Plug into BLDL parm list
         LA    R8,12               .Length of each entry
         STH   R8,ENTLENG          .Plug into BLDL parm list
         XR    R15,R15             .This routine successful
BUILDLSX PR
**********************************************************************
*        This routine merges the SYSIN names or the member name as
*        passed with PARM='MEMNAME' with the BLDL list into a new list
**********************************************************************
PARTLBLD BAKR  R14,Ø
         TM    ONEMEM,YES          .One member passed as a parm?
         BNO   FROMCARD            .No, member list from SYSIN
         LA    R3,1                .Space for 1 member only
         LA    R4,1                .Scan 1 member only
         LA    R5,MEMNAME          .Entry from PARM= is here
         B     CALCSPCE            .Go calculate the space
FROMCARD L     R3,NUMSYSIN         .Number of cards on SYSIN
         L     R4,NUMSYSIN         .Scan this many members
         L     R5,TABSTART         .Entries from SYSIN are here
CALCSPCE MH    R3,=H'12'           .BLDL size for member
         LA    R3,4(R3)            .BLDL list header
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY  Obtain storage to merge
         LTR   R15,R15             .Did we get the storage?
         BZ    KEEPADDR            .Yes
         B     NOSTOR              .Go ABEND
KEEPADDR ST    R1,PARTLIST         .Address of the partial list
         ST    R3,PARTSIZE         .Size of the list
         LR    R3,R1               .Point to start of list
         LA    R3,PENTRIES         .Point to start of list
         XR    R8,R8               .Counter in new table
         LR    R8,R1               .Pointer in new table
         XR    R1Ø,R1Ø             .Match-found counter
BIGLOOP  EQU   *
         LH    R6,NUMENTRY         .Number of members in PDS
         LA    R9,MEMLIST          .Start of BLDL obtained from PDS
         CLI   8(R5),C'D'          .Is member marked for deletion?
         BNE   SMLLOOP             .No, go scan for it
         MVC   Ø(8,R3),Ø(R5)       .Move the name in
         MVC   8(4,R3),=4X'ØØ'     .Move binary zeroes in
         LA    R3,12(R3)           .Point to next slot
         LA    R1Ø,1(R1Ø)          .Add 1 to match-found counter
         B     BUMPUP              .Yes, won't be in BLDL list
SMLLOOP  CLC   Ø(8,R5),Ø(R9)       .Compare SYSIN with BLDL entry
         BE    MATCHFND            .Same
         LA    R9,12(R9)           .Bump up BLDL list pointer
         BCT   R6,SMLLOOP          .Do for each member in BLDL list
         MVC   NOMATCH+29(8),Ø(R5) .Move the member name into the WTO
NOMATCH  WTO  'LLABATCH(W): -Member xxxxxxxx is not in the PDS, refreshX
               could not be done',ROUTCDE=11
         OC    RETCODE,=F'4'       .Set return code to 4
         B     BUMPUP
MATCHFND EQU   *
         LA    R1Ø,1(R1Ø)          .Add 1 to match-found counter
         MVC   Ø(12,R3),Ø(R9)      .Move entry into the partial list
         LA    R3,12(R3)           .Bump up the pointer
BUMPUP   LA    R5,9(R5)            .Bump up SYSIN list pointer
         BCT   R4,BIGLOOP          .Do for each member in SYSIN
         L     R3,PARTLIST         .Start of partial list
         STH   R1Ø,PARTNUM         .Number of qualifying entries
         LA    R1Ø,12              .Size of each entry
         STH   R1Ø,PENTLENG        .Plug the length of each entry
PARTLBLX PR
*********************************************************************
*        This routine does the actual LLACOPY
*********************************************************************
LLACPMAC BAKR R14,Ø
         MODESET MODE=SUP,KEY=ZERO
         MVC   LLAAREA(LLAMACL),LLAMAC
         LA    R2,LLAAREA
         XR    R6,R6
         ICM   R6,3,Ø(R3)          .(Save) the original number
         XR    R5,R5
         ICM   R5,3,Ø(R3)          .Pick up # of entries to cater for
         CH    R5,=X'7FFF'         .More than 32K?
         BNH   LLALOCL1            .No, proceed
         SRL   R5,1                .Divide it by 2
         STH   R5,Ø(R3)            .Update the number to refresh
         OI    SPLTCOPY,YES        .Set flag to show we are doing 2
LLALOCL1 LLACOPY DCB=LLADSN,BLDLLIST=(R3),MF=(E,(2))
         ST    R15,RC
         OC    RETCODE,RC
         CH    R15,=H'4'           .RC > 4?
         BH    RESETMD2            .Yes, get out
         TM    SPLTCOPY,YES        .Is there a second half?
         BNO   RESETMD2            .No, get out
         SR    R6,R5               .Subtract # already done
         MH    R5,=H'12'           .Multiply length with # done
         AR    R3,R5               ,Move 'start of list' pointer
         STCM  R6,3,Ø(R3)          .The number of entries in 2nd half
         MVC   2(2,R3),=X'ØØØC'    .Plug the entry length
         LA    R2,LLAAREA
LLALOCL2 LLACOPY DCB=LLADSN,BLDLLIST=(R3),MF=(E,(2))
         ST    R15,RC
RESETMD2 MODESET MODE=PROB,KEY=NZERO
LLALOCLX OC    RETCODE,RC
DOPR     PR
**********************************************************************
*        Constants follow
**********************************************************************
LLADSN   DCB   RECFM=FB,LRECL=256,DSORG=PS,MACRF=GL,DDNAME=LLADSN,     *
               BLKSIZE=256,EODAD=ENDOFLST,EXLST=JFCB@
SYSIN    DCB   RECFM=FB,DSORG=PS,MACRF=GL,DDNAME=SYSIN,                *
               LRECL=8Ø,EODAD=ENDSYSIN,EXLST=JFCB@
         DS    ØF
OKWTOMS  WTO   'LLABATCH(I): -LLA refresh completed for DSN=           X
                                            ',ROUTCDE=11,MF=L
OKWTOLEN EQU   *-OKWTOMS
ERWTOMS  WTO  'LLABATCH(E): -Unsuccessful LLA refresh, RCODE=xxxx, ReasX
               on=YYYY',ROUTCDE=11,MF=L
ERWTOLEN EQU   *-ERWTOMS
LLAMAC   LLACOPY MF=(L,BLDLLST)
LLAMACL  EQU   *-LLAMAC
STORSIZE DC    AL4(LISTSIZE+GETMSIZE)
TABSIZE  DC    AL4(LISTSIZE)
LOWVALS  DC    F'Ø'
         DS    ØF
JFCB@    DC    X'87',AL3(JFCB)     .Address of JFCB work area
JFCB     DS    CL176
MEMAREA  DS    ØF
         DC    AL4(MEMSPCE)
MEMSPCE  EQU   18ØØØØ              .Space for 2Ø ØØØ members in table
*                                  .Each entry = 9 bytes: 8-byte name
*                                  . + 1 byte set to "D" if member has
*                                  . to be deleted.
MAXMEMS  DS    ØF
         DC    AL4(MEMSPCE/9)      .Maximum # of entries
         LTORG
**********************************************************************
*        DSECTS follow
**********************************************************************
STORAREA DSECT
SAVEAREA DS    18F                 .General savearea
PARMSTRT DS    F                   .Start address of passed parms
RC       DS    F                   .Return code received
```

```
RETCODE  DS    F                     .Return code passed to caller
TABSTART DS    F                     .Start of table with parm members
PARTLIST DS    F                     .Start of partial list
PARTSIZE DS    F                     .Size of partial list
DOUBLE   DS    D                     .Double word work area
*
NUMSYSIN DS    F                     .Number of SYSIN cards
DSNAME   DS    CL44                  .Name of the dataset for LLACOPY
MEMNAME  DS    CL8                   .Name of the member name from parm
TEMPNAME DS    CL9                   .Temporary name holder used to sort
ALLMEMIN DS    C                     .Flag to indicate "all members"
ONEMEM   DS    C                     .Flag to indicate one mem from parm
WTOAREA  DS    CL(OKWTOLEN)          .Work area for WTO macro
         DS    ØF
LLAAREA  DS    CL(LLAMACL)           .Area for LLACOPY macro
         DS    ØD
BLDLLIST DS    ØF    |
NUMENTRY DS    H     |——             .BLDL parameter list. Don't move
ENTLENG  DS    H     |
MEMLIST  DS    ØH    |
SPLTCOPY DS    C                     .Flag to indicate doing 2 LLACOPYs
LISTSIZE EQU   2+2+(12*65535)        .Allow for up to X'FFFF' members
GETMSIZE EQU   *-STORAREA
PRTDSECT DSECT                       .DSECT for partial list
PARTNUM  DS    H                     .Number of entries in list
PENTLENG DS    H                     .Length of each entry in list
PENTRIES DS    ØH                    .Start of entry names
*
NO       EQU   X'ØØ'
YES      EQU   X'Ø4'
POSTPONE EQU   X'84'
         IKJDAPØ8
         IEFJFCBN
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

Systems Programmer (UK) © Xephon 2001

Wait function for REXX programs

Sometimes it is necessary to wait for a period of time in your program. In Assembler programs this is not a problem because you can code an STIMER macro to wait for a desired time. We wanted to know how to achieve this with a REXX program.

You can do this in a REXX program by calling a load module in it. Another method is to write an Assembler function. I developed a function for this purpose. This function gets a parameter from the caller. The format of this parameter is hhmmss, as a waittime, then it waits without using resources. Sometimes you have to stop waiting. By using STIMER the only way to do this is Cancel the program. This is not desirable so I changed the structure of the program.

This function attaches another program to the wait process. The main part waits for the end of the attached program or operator command from the console. If either event happens then it returns to the caller. The return value of the function can be NOP (attached TCB ended), STOP (STOP command entered by the operator), or a value that is entered by the MODIFY command. Also it can return PARMERROR (wait time for the TCB is not given), FORMATERROR (parameter format should be hhmmss), and PARMNOTNUMERIC (parameter should be numeric). The source of the program is shown below.

WAITREXX

```
*******************************************************************
**    Module Name : WAITREXX                                     **
**                                                               **
**    Type        : Function                                     **
**    Parameters  : Parm1 ( Wait time as HHMMSS Format )         **
**                                                               **
**    It gets a parameter. Format of this paramater is HHMMSS.   **
**    It waits for a time period that is given to the function   **
**    as this parameter. It attaches a subroutine to this module.**
**    This subroutine uses STIMER macro for wait process. The    **
**    function also waits for MODIFY command. If time ECB is     **
**    posted (STIMER) macro, it returns NOP. If modify command   **
**    is entered, then this function returns modify command.     **
**    User can also enter STOP command to function, then it      **
**    returns STOP.                                              **
**                                                               **
**    Return values of this function.                                   **
**                                                                      **
**    Return          Reason                                  Message  **
**    ———————— ———————————————— ———— **
**    NOP             No Error                                   -     **
**    STOP            No Error ( STOP Command issued )           -     **
**    PARMERROR       Parameter should be given              WAITERR1 **
**    FORMATERROR     Parameter format should be HHMMSS     WAITERR2 **
**    PARMNOTNUMERIC Parameter should be numeric           WAITERR3 **
**    -               Invalid MODIFY command                 WAITERR4 **
**    ?????????????? No Error, modify command value            -     **
*******************************************************************
WAITREXX CSECT
WAITREXX AMODE 31
WAITREXX RMODE ANY
*******************************************************************
         REQUATE                     .Rename registers
*******************************************************************
         SAVE (14,12),,'WAITREXX,Date:&SYSDATE,Time:&SYSTIME,Ergun'
         BASR  R12,Ø
         USING *,R12
         LA    R2,SAVE
         ST    R13,4(R2)
         ST    R2,8(R13)
         LR    R13,R2
*******************************************************************
         LR    R11,R1
         USING EFPL,R11              .Addressability of Ext.Func.Parm.
         L     R1Ø,EFPLARG
         USING ARGTABLE_ENTRY,R1Ø    .Addressability of argument table
         L     R8,EFPLEVAL
         L     R9,Ø(Ø,R8)
         USING EVALBLOCK,R9          .Addressability of EvalBlock
         L     R8,ARGTABLE_ARGSTRING_PTR
         C     R8,=F'-1'             .Parameter exists?
         BNE   CHECKLEN              .If yes, check length
ERR1     $REXXMSG  WAITERR1,34        .No parameter supplied
         LA    R5,9                  .Length of return value
         ST    R5,EVALBLOCK_EVLEN    .Put length into variable
         MVC   EVALBLOCK_EVDATA(9),=C'PARMERROR' .Return Value
         B     SETRC                 .Return to REXX pgm.
*******************************************************************
CHECKLEN L     R7,ARGTABLE_ARGSTRING_LENGTH
         C     R7,=F'6'              .Parameter length should be 6
         BE    CHECKNUM              .If yes, check numeric?
ERR2     $REXXMSG  WAITERR2,42       .Parameter length error
         LA    R5,11                 .Length of return value
         ST    R5,EVALBLOCK_EVLEN    .Put length into variable
         MVC   EVALBLOCK_EVDATA(11),=C'FORMATERROR'
         B     SETRC                 .Return to REXX pgm.
*******************************************************************
CHECKNUM EQU   *
```

```
         MVC   INPARM(6),Ø(R8)       .Get wait time in HHMMSS format
         NC    FULLZERO(8),INPARM    .Is parameter numeric?
         CLC   FULLZERO,=C'ØØØØØØØØ'
         BE    ATC_TCB               .If yes, attach TCB
ERR3     $REXXMSG  WAITERR3,36       .Parameter does not numeric
         LA    R5,14                 .Length of return value
         ST    R5,EVALBLOCK_EVLEN    .Put length into variable
         MVC   EVALBLOCK_EVDATA(14),=C'PARMNOTNUMERIC'
         B     SETRC                 .Return to REXX pgm.
*
*--------- SUPPLIED PARAMETER OK, CONTINUE --------
*
ATC_TCB  LA    R5,ADRRESPA           .Address of Response area
         EXTRACT (R5),FIELDS=COMM    .Command Schedular Comm. List
         L     R5,ADRRESPA           .Get returned address
         USING COM,R5                .Let Assembler know IEZCOM
         ATTACH EP=WAITTCB,ECB=TCBECB,JSTCB=YES,PARAM=INPARM
         ST    R1,SAVETCB            .R1 points address of attached TCB
         ICM   R3,15,COMCIBPT        .Address of CIB
         USING CIB,R3                .Let Assembler know IEZCIB
         BZ    NOCIB                 .BIF not present
         BAL   R14,FREECIB           .Free CIB
NOCIB    DS    ØH
         QEDIT ORIGIN=COMCIBPT,CIBCTR=1
         L     1,COMECBPT
         O     1,HIBITON
         ST    1,ECBS+4
WAIT     DS    ØH
         WAIT  1,ECBLIST=ECBS        .Wait for an ECB post
         TM    TCBECB,X'4Ø'          .Attached ended?
         BO    TCBENDED              .If yes, process it
         B     CIBCIB                .No, Modify or Stop entered
TCBENDED LA    R5,3                  .Return
         ST    R5,EVALBLOCK_EVLEN    .Evalblock
         MVC   EVALBLOCK_EVDATA(3),=C'NOP'
         DETACH SAVETCB              .Detach TCB
         BAL   R14,FREECIB
         B     SETRC                 .REXX program1
CIBCIB   L     R3,COMCIBPT           .Check for Stop or Modify
         CLI   CIBVERB,CIBSTOP       .Stop entered?
         BNE   DOCIB                 .If not, get modify value
STOPRTN  LA    R5,4                  .Length of the return value
         ST    R5,EVALBLOCK_EVLEN    .Put it into EvalBlock
         MVC   EVALBLOCK_EVDATA(4),=C'STOP'
         DETACH SAVETCB              .Detach TCB
         BAL   R14,FREECIB
         B     SETRC                 .Return to REXX Pgm.
DOCIB    DS    ØH
         CLI   CIBVERB,CIBMODFY      .Check for Modify command
         BNE   NOCIB
         LA    4,CIBDATA             .Address of Modify command
         LH    8,CIBDATLN            .Length of the modify command
```

```
SKIPBLNK DS    ØH                      .Skip blanks in the command
         CLI   Ø(4),X'4Ø'
         BNE   ENDSKPBL
         LA    4,1(4)
         BCT   8,SKIPBLNK
ENDSKPBL C     8,=F'Ø'                 .Check length of the command
         BE    INVPARM                 .If no command entered, warning
         ST    R8,EVALBLOCK_EVLEN      .Length of the return value
         LA    R7,EVALBLOCK_EVDATA
         EX    R8,MOVEPARM
         DETACH SAVETCB                .Detach TCB
         BAL   R14,FREECIB
         B     SETRC                   .Return to REXX program
MOVEPARM MVC   Ø(Ø,R7),Ø(R4)           .Move command into EvalBlock
INVPARM  DS    ØH
ERR4     $REXXMSG  WAITERR4,45         .Invalid modify command
         BAL   14,FREECIB
         B     NOCIB
SETRC    LA    R15,Ø                   .Set return code of the function
         L     R13,SAVE+4
         L     R14,12(,R13)
         LM    R1,R12,24(R13)
         BR    R14                     .Go back to caller
FREECIB  DS    ØH
         QEDIT ORIGIN=COMCIBPT,BLOCK=(3)
         BR    14
*******************************************************************
** Parameters of the Function                                    **
*******************************************************************
         DS    ØF
SAVE     DS    18F                     .Save area for the function
ADRRESPA DS    A                       .Address of the response area
TCBECB   DS    A
SAVETCB  DS    A                       .Address of Attached TCB
ECBS     DC    A(TCBECB)
         DS    A
*                                      .Error messages of the function
WAITERR1 DC    CL34'WAITERR: Parameter should be given'
WAITERR2 DC    CL42'WAITERR: Parameter format should be HHMMSS'
WAITERR3 DC    CL36'WAITERR: Parameter should be numeric'
WAITERR4 DC    CL45'WAITERR: Invalid parameter for MODIFY command'
INPARM   DC    CL8'ØØØØØØØØ'           .Input parameter
FULLZERO DC    CL8'ØØØØØØØØ'           .Check for numberic
HIBITON  DC    X'8ØØØØØØØ'             .For ECB post processing
         DS    ØF
         $REXXMSG  VARS=YES            .Declare variables for warn.Msg.
         LTORG
         IRXEFPL                       .Mapping macro for Ext.Func.Parm
         IRXARGTB                      .Mapping macro for argument table
         IRXEVALB                      .Mapping macro for EvalBlock
COM      DSECT
         IEZCOM
```

```
CIB      DSECT
         IEZCIB
         END
```

WAITTCB

Attached program for Wait purposes:

```
*******************************************************************
**    Module Name : WAITTCB                                      **
**    Type        : Program                                      **
**    Parameters  : Parm1 ( Wait time as HHMMSS Format )         **
**                                                               **
**    It get's a parameter. Format of this parameter is HHMMSS.  **
**    It waits for a time that is given to this program as a     **
**    parameter. It executes STIMER macro. At the end of the     **
**    wait time it returns to caller.                            **
*******************************************************************
WAITTCB  CSECT
WAITTCB  AMODE 31
WAITTCB  RMODE 24
         REQUATE                  .Rename registers
         SAVE  (14,12),,'WAITTCB,DATE:&SYSDATE,TIME:&SYSTIME,ERGUN'
         BASR  R12,Ø
         USING *,R12
         LA    R2,SAVE
         ST    R13,4(R2)
         ST    R2,8(R13)
         LR    R13,R2
         L     R2,Ø(R1)           .R1 points to parameter
         MVC   WAITTIME,Ø(R2)     .Move wait time into WAITTIME
         STIMER WAIT,DINTVL=WAITTIME .Wait till the end
         L     R13,SAVE+4         .Return to caller
         ST    R15,16(,R13)
         LM    R14,12,12(R13)
         BR    R14
         DS    ØF
SAVE     DS    18F                .Save area of the program
WAITTIME DC    C'ØØØØØØØØ'        .Wait time as hhmmsstt
         END
```

$REXXMSG

Macro for REXX error messages:

```
*/***START OF SPECIFICATIONS******************************************/
*/* MACRO-NAME = $REXXMSG                                            */
*/* DESCRIPTIVE NAME = Displays a message in a REXX program, by      */
*/*                    using IRXSAY macro                            */
*/***END OF SPECIFICATIONS********************************************/
```

```
         MACRO
&L       $REXXMSG &MSG,&LENG,&VARS=NO
         AIF   ('&VARS' EQ 'YES').LABØØØ1
         ST    R7,SAVEREG             Keep content of work register
         LA    R7,&LENG               Load message length
         ST    R7,MSGLEN              Length is used in IRXSAY
         LA    R7,&MSG                Address of output message
         ST    R7,MSGADDR             Address is used in IRXSAY
         LINK  EP=IRXSAY,PARAM=(WRITE,MSGADDR,MSGLEN),VL=1
         L     R7,SAVEREG             Restore content of work register
         AGO   .LABØØØ2
.LABØØØ1 ANOP
WRITE    DC    CL8'WRITE   '          Function to be performed
MSGLEN   DS    F                      Output message length
MSGADDR  DS    A                      Output message length
SAVEREG  DS    F                      To keep content of work register
.LABØØØ2 ANOP
         MEND
```

Sample use of the Function in a REXX program:

```
/**REXX**************************************************************/
/* WAITTIME : Sample REXX program for usage of WAITREXX function.   */
/********************************************************************/
Pull WaitTime                     /* Get Initial WaitTime from user */
If WaitTime = '' Then             /* No parameter given             */
  WaitTime = 'ØØØ1ØØ'             /* Default WaitTime, ØØØ1ØØ       */
ReturnValue = ''                  /* To Control looping             */
Do Until ReturnValue = |STOP|
  Say 'Time is now ' || Time() || ', program will wait ' ||,
    WaitTime || ' ( As HHMMSS format )'
  RETURNVALUE = WAITREXX(WaitTime)
  If RETURNVALUE = |NOP| Then
    NOP
  Else If RETURNVALUE = <STOP< Then
    NOP
  Else
    If Pos(RETURNVALUE,'PARMERROR-FORMATERROR-PARMNOTNUMERIC') ¬= Ø Then
      Do
        Say 'Error in parameter'
        Exit(2Ø)
      End
  Else
    WaitTime = RETURNVALUE
End
Exit
```

Ergun Ozel
Project Manager
Vizyon Bilgi Islem (Turkey)

Automated and interactive library update using edit macros

INTRODUCTION

Often there is a need to change strings in a library in an MVS environment. But changing a specific string in libraries with many members, it would be a very hard task to do without using an edit macro.

For example, assume that we have several JCL libraries related to a product, and that members of these libraries are referring to a back-level product qualifier, which is BBIOA.V514, and all these occurrences have to be substituted with BBIOA.V611, which is the new version qualifier.

I have developed a small utility which consists of REXX programs, edit macros, several panels, and a message library member to automate this sort of task, in a panel-driven and interactive way.

WHAT IS THE ALGORITHM?

The edit macro used in this article is run against all members of a PDS. A loop in the REXX program uses a LISTDS command output to obtain the PDS dataset members' names. For each member, an ISPEXEC EDIT command is issued with the initial MACRO keyword.

```
    (Ispexec Edit Dataset(...) Macro(...) command. )
```

When the main REXX CHGREXX is called, it will display the panel PANEL0 to get all 'Change' command parameters from the user, including the PO dataset to be updated (it is called as SOURCE dataset), 'From' string, 'To' string, etc. Then, a backup copy of the SOURCE dataset is created for backout purposes. (It is called as BKPCPY dataset).

Later on, CHGREXX reads all members of the SOURCE dataset and calls the edit macro CHGMAC within an 'ISPEXEC EDIT DATASET' command for each member read. If that member has at least one occurrence of the ‘From string’, then those strings are changed and a ‘Save’ operation is performed. If the member is saved successfully, ISPF statistics of that member are changed as well. The ‘Save’ command in the edit macro sets the ID field (the user ID that last modified the data) as the TSO user-id. But the REXX CHGREXX sets the UID field as CHGMAC so that, at a later time, the user can easily distinguish members which were changed by the edit macro while editing the dataset.

Then the result of the edit macro execution is written, member by member, into an ISPF table, which will be displayed in Panel 3 in the next step.

Then CHGREXX compares the SOURCE dataset, which may have have been updated, with the BKPCPY dataset, which is the original version of the SOURCE dataset, by a SuperC utility and displays the differences found to the user. It is meant to show the user differences between the initial and the final contents of the SOURCE dataset. Finally, Panel 2 is displayed and the user is asked if she/he accepts the changes just made by the utility. Depending on his/her response, the BKPCPY dataset is kept or deleted. The Panel 1 displays an explanatory message to the user during the execution of the utility. There is also a help panel, which is Panel H.

HOW TO RUN THE REXX CHGREXX

To make the use of this application easy, I recommend putting all REXXs, panels, the edit-macro, and the message member into a single library. In this article, all of them are members of the dataset EXP.CTM.REXX. Then it will be sufficient just to call the REXX CHGREXX in the following way:

```
TSO EX 'EXP.CTM.REXX(CHGREXX)'
```

The REXX CHGMAC will do all necessary allocations for ISPF message, panels, the REXX, and the macro by using a series of ALTLIB and LIBDEF commands.

Another way, of course, is to put all panel definition members into an ISPPLIB, the REXX and the edit-macro into a SYSEXEC, and the message member into an ISPMLIB concatenation library. Note that if this method is chosen, all ALTLIB and LIBDEF commands, which are not necessary, have to be removed from the REXX CHGREXX. Then you can call the REXX simply by its name:

```
TSO %CHGREXX
```

But for easy maintenance, I would recommend using the first approach.

A SHORT OVERVIEW ON EDIT MACROS

You can use edit macros, which look like ordinary editor commands, to extend and customize the editor. You create an edit macro by placing a series of commands into a dataset or member of a partitioned dataset. Then you can run those commands as a single macro by typing the defined name in the 'Command ===>' line in an edit session or you call an edit macro from a REXX. The second method not only provides us with the capability of running an edit macro many times in a loop, but also saves us time by not having to edit each member to be able to run the edit macro against that member.

Edit macros can be either CLISTs or REXX EXECs written in the CLIST or REXX command language, or program macros written in a programming language (such as FORTRAN, PL/I, or COBOL).

Edit macros have access to the dialog manager and system services. Because edit macros are CLISTs, or REXX EXECs, programs, they have unlimited possibilities. Edit macros can be used to:

- Perform repeated tasks
- Simplify complex tasks
- Pass parameters
- Retrieve and return information.

The REXX edit macros must include a REXX comment line (/* REXX */) as the first line of each edit macro to distinguish them from CLIST edit macros. This comment line can contain other words or characters if necessary, but it must include the string REXX.

REXX edit macros must be in partitioned datasets. REXX edit macros can exist in the following concatenations: SYSUEXEC, ALTLIB (for datasets activated as EXECs), and SYSEXEC. Datasets in these concatenations can contain only REXX EXECs.

A REXX edit macro is made up of REXX statements. Each statement falls into one of the following categories:

- Edit macro commands
- CLIST or REXX command procedure statements and comments
- ISPF and PDF dialog service requests
- TSO commands.

You can run PDF edit macros in batch by submitting JCL which allocates all of the necessary ISPF libraries, and runs a command that calls the EDIT service with an initial macro. This initial macro can do anything that can be done by an initial macro in an interactive session. However, in batch, the macro should end with an ISREDIT END or ISREDIT CANCEL statement. These statements ensure that no attempt is made to display the edit screen in batch.

We can pass parameters to an edit macro. A parameter can be either a simple string or a quoted string. It can be passed by using the standard method of putting variables into shared and profile pools (use VPUT in dialogs and VGET in initial macros and *vice versa*). This method is best suited to parameters passed from one dialog to another, as in an edit macro. The edit macro and the REXX used in this utility use the VGET/VPUT commands to communicate with one another. For more information on edit macros see the IBM book *ISPF Edit and Edit Macros*.

SOME NOTES ON THE UTILITY

A 'From string' and a 'To string' have to be specified in PANEL1 of the application. If the string is a simple or delimited string *(a string that begins and ends with apostrophes or quotes),* the characters are treated as being both upper and lower case even if caps mode is off. For example, this command:

```
CHANGE ALL 'CONDITION NO. 1' '........'
```

This changes the following:

- CONDITION NO. 1
- Condition No. 1

- condition no. 1
- coNDitION nO. 1

Also, all of the following commands have the same effect:

```
CHANGE 'Edit Commands' '........'
CHANGE 'EDIT COMMANDS' '........'
CHANGE 'edit commands' '........'
```

For this reason, if you want the change command to be satisfied by an exact character-by-character match, lower case alphabetic characters matching only with lower case alphabetic characters, and upper case alphabetic characters matching only with uppercase, a character string must be used. For example, to change ‘aBc’ to ‘AbC’ you have to use the command:

```
Change C'aBc' 'AbC' ALL
```

Besides character strings, you can use picture strings to change a particular kind of character without regard for the specific character involved. For example to change any lower case character to upper case, use the command:

```
CHG p'<' p'>'
```

To change a string of hexadecimal digits, you can use a hex string. For example, CHG 'c1c2'x 'a1a2'x. To change a character string regardless of whether alphabetic characters are upper or lower case, use a text string. For example, the following command changes the text «spf» to caps:

```
CHG t'spf' SPF
```

To limit the strings that are found, you can use qualifying parameters. This way, you will specify additional characteristics of string-1 (from string) by using the operands PREFIX, SUFFIX, CHARS, and WORD.

- CHARS – locates string-1 anywhere the characters match. This is the default.
- PREFIX – locates string-1 at the beginning of a word.
- SUFFIX – locates string-1 at the end of a word.
- WORD String-1 – is delimited on both sides by blanks or other non-alphanumeric characters.

The col-1 and col-2 operands allow the user to search only a portion of each line, rather than the entire line. These operands, which are numbers separated by a comma or by at least one blank, show the starting and ending columns for the search. The following rules apply:

- If you specify neither col-1 nor col-2, the search continues across all columns within the current boundary columns (BOUNDS line).
- If you specify col-1, the editor finds the string only if the string starts in the specified column.
- If you specify both col-1 and col-2, the editor finds the string only if it is entirely within the specified columns.
- If the second column specified is larger than the record size, this is an error condition and the edit macro of the utility (CHGMAC) will substitute the second column with the record length of the dataset being changed.

ADDITIONAL NOTES AND CONCLUSION

1. This edit-macro with its accompanying REXX program can be used in the migration processes. For example, it is helpful to make multi-changes in the libraries when a new version of a product is implemented. Also you can make the edit macro a little more complex by using more edit primary commands such as ‘Change’ or ‘Exclude’. For this, Panel 1 has to be accommodated to the changes made on the edit-macro since they work together. For example, to incorporate one more Change command, you would have to prepare a continuation panel to Panel 1.

   Another advantage of using this utility is that it permits us to change strings of up to 255 characters in length. Remember that an ordinary change command on the ISPF edit command line is not able to do that due to the limitation of the edit command line.

2. If it seems that the PO dataset has too many members and most of them are candidates to have updates when the REXX CHGREXX runs, special care must be taken regarding S37 abend. The REXX CHGREXX compresses the SOURCE dataset at the beginning of the utility as a precaution. For this reason, it

is recommended that you enlarge the PO dataset to tolerate possible growth due to many possible ‘Isredit Save’ commands in the edit macro (CHGMAC). However, if you see an S37 abend condition on the result panel Panel 3 at the end of the execution, you can compress the SOURCE dataset in another split-screen and re-execute the utility.

3 If it is not necessary to work interactively, you can make changes in the PO libraries without using the panel as well. On the other hand you can make changes in several PO datasets at the same time. The REXX program BATCHREX and the edit mcro BATCHMAC are given as an example abridged codes and do not contain any error recovery control.

To execute, use the command:

```
TSO EX 'EXP.CTM.REXX(BATCHREX)'
```

REXX: CHGREXX

```
/*REXX*/
/*------------------------------------------------------------------*/
/* AUTOMATED AND INTERACTIVE LIBRARY UPDATE BY USING EDIT MACRO      */
/*                                                                   */
/* REXX program       : ChgRexx                                      */
/* Edit macro called  : ChgMac                                       */
/*------------------------------------------------------------------*/
Status = Msg('Off')
/*------------------------------------------------------------------*/
/* Let EXEC process errors. This way we can check the Return Codes   */
/* and take the appropriate actions.                                 */
/*------------------------------------------------------------------*/
|Ispexec Control Errors Return|
/*-----------------------------------------------------------------*/
/* Make necessary library allocations for REXX/Panel/Message members.*/
/*-----------------------------------------------------------------*/
|Altlib  Activate Application(Exec) Da(Exp.Ctm.Rexx)|
|Ispexec Libdef Ispplib Dataset     Id(Exp.Ctm.Rexx)|
|Ispexec Libdef Ispmlib Dataset     Id(Exp.Ctm.Rexx)|
/*------------------------------------------------------------------*/
/* Clean some profile variables.                                     */
/*------------------------------------------------------------------*/
|Ispexec Verase (Mes,Mem,Col1,Col2,Rc1,Rc2,Rc3,Cnt,Chg,Err,Msg1,Msg2)|
REPEAT:
/*------------------------------------------------------------------*/
/* Display the panel PanelØ to get the dataset name, and |Ispf Edit  */
/* command| parameters.                                              */
```

```
/*-------------------------------------------------------------------*/
«Ispexec Display Panel(PanelØ)»
/*-------------------------------------------------------------------*/
/* Control that if PFØ3 or PFØ4 key is pressed on the PanelØ. If so,  */
/* deallocate the libraries allocated by Libdef & Altlib commands.    */
/*-------------------------------------------------------------------*/
«Ispexec Vget (Spfkey,Dsn,Col1,Col2,From,To) Profile»
If (Spfkey = PFØ3 3 Spfkey = PFØ4) Then Do
                       «Ispexec Setmsg Msg(EdtmØØ6)»
                       «Altlib Deactivate Application(Exec)»
                       «Ispexec Libdef Isplib»
                       «Ispexec Libdef Ispmlib»
                       Exit
                      End
/*-------------------------------------------------------------------*/
/* Check if the dset name entered on PanelØ is an existing PO dset.   */
/* From now on, we will be calling this dataset as 'SOURCE dset.'     */
/*-------------------------------------------------------------------*/
IF Sysdsn(Dsn) = 'OK' Then
             Do
              X = Listdsi(Dsn)
              If X ¬= Ø Then
                Say 'Some Listdsi info not available. Function code =' X
                Else Do
                      Dsorg   = Sysdsorg
                      If Dsorg ¬= PO Then
                                     Do
                                       «Ispexec Setmsg Msg(EdtmØØ9D)»
                                       Signal Repeat
                                     End
                     End
             End
             Else Do
                   «Ispexec Setmsg Msg(EdtmØØ9C)»
                   Signal Repeat
                  End
/*-------------------------------------------------------------------*/
/* Check whether the SOURCE dset has any member?                      */
/*-------------------------------------------------------------------*/
 x = Outtrap('Var.')
 «Listds» Dsn «Members»
 x = Outtrap('Off')        /* Turns trapping OFF */
  Cnt = Var.Ø-6
  «Ispexec Vput Cnt Profile»
  If Cnt = Ø Then Do
                   «Ispexec Setmsg Msg(EdtmØØ9B)»
                   Signal Repeat
                  End
/*-------------------------------------------------------------------*/
/* Check that if From or To string that are entered on PanelØ is      */
/* greater than the record length of the SOURCE dataset.              */
```

```
/*--------------------------------------------------------------------*/
 L2    = Syslrecl
 If Length(From) > L2    Then Do
                                   L1 = Length(From)
                                   «Ispexec Vput (L1,L2) Profile»
                                   «Ispexec Setmsg Msg(EdtmØØ8)»
                                   Signal Repeat
                              End
 If Length(To)   > L2    Then Do
                                   L1 = Length(To)
                                   «Ispexec Vput (L1,L2) Profile»
                                   «Ispexec Setmsg Msg(EdtmØØ9)»
                                   Signal Repeat
                              End
/*--------------------------------------------------------------------*/
/* Check whether Col1 or Col2 is greater then the LRECL of the SOURCE */
/* dataset. If so, do not continue any more. In addition, display     */
/* the LRECL of the SOURCE dset for informative purposes.             */
/*--------------------------------------------------------------------*/
  If (Col2 > L2) 3 (Col1 > L2)  Then Do
                                       «Ispexec Setmsg Msg(EdtmØØ9J)»
                                       Mes = '*** Lrecl : 'L2
                                       «Ispexec Vput Mes Profile»
                                       Signal Repeat
                                     End
/*--------------------------------------------------------------------*/
/* Compress the SOURCE dataset to prevent it from producing S37 abend */
/*--------------------------------------------------------------------*/
Chgiebcp Dsn Dsn
/*--------------------------------------------------------------------*/
/* Allocate a back-up copy of the SOURCE dataset. From now on we      */
/* will be calling it as BKPCPY dataset.                              */
/*--------------------------------------------------------------------*/
Dsn_bkp = Dsn'.BKP'
 «Alloc Da(«Dsn_bkp») Like(«Dsn»)»
 If Rc ¬=Ø Then
            Do
             Say 'BKPCPY dset allocation is not successful'
             If Rc =12 Then Say 'The dset,' Dsn_bkp 'already exists.',
                              'Please check it out.'
              Exit
             End
 «Free Da(«Dsn_bkp»)»   /*   Free BKPCPY dataset. */
/*--------------------------------------------------------------------*/
/* Call the system utility IEBCOPY to copy the SOURCE dataset         */
/* members into the BKPCPY dataset.                                   */
/*--------------------------------------------------------------------*/
Chgiebcp Dsn Dsn_bkp
/*--------------------------------------------------------------------*/
/* At this point, we have BKPCPY dataset built. So we can start       */
/* executing the Edit macro over the SOURCE dataset members to do     */
/* batch string updates.                                              */
/*————————————————————————————————————————————————————————————————————*/

/*————————————————————————————————————————————————————————————————————*/
/* A table which will show edit-macro execution status for each       */
/* member of the SOURCE dataset wll be created.                       */
/*                                                                    */
/* Table_output_library = Ispf profile dataset.(File Name = ISPTABL)  */
/* Table_name = TABLExxx (xxx = 1,...,1ØØ )                           */
/*                                                                    */
/* A user can ask to run this application more than once on a split
*/
/* screen. Each time a different table name will be generated.        */
/*————————————————————————————————————————————————————————————————————*/
 Prfxusr  = Sysvar(sysuid)
 «Alloc Fi(Isptabl) Da('«33 Prfxusr 33 «.Ispf.Ispprof') Shr»

 Ran = Random(1,1ØØ)
 Table_name = 'TABLE' 33 ran
 «Ispexec Tbcreate» Table_name «Names(Mem Chg Err Msg1 Rc1 Rc2 Msg2)»
/*———————————————————————————————————————————————————————————————————*/
/* Get members of SOURCE dset and execute the edit-macro in a loop.  */
/*———————————————————————————————————————————————————————————————————*/
 Do i = 7  To Var.Ø    /* Loop-Martapv */
   Mem = Strip(Var.i)
   Call Editmac
   Rc1 = Result
   «Ispexec Vput (Mem,Rc1) Profile»
   «Ispexec Vget (Chg,Err)      Profile»
   «Ispexec Vget (Msg2,Rc2,Rc3) Profile»  /*  Get «ISPEXEC EDIT» and  */
                                          /*  «SAVE» command Rc.      */
   Select
    When (Rc1 = Ø )           Then Msg1= 'Member updated.  '
    When ((Rc1=4) & (Rc3>=4)) Then Msg1= 'No save.Abend S37'
    When (Rc1 = 4 )           Then Msg1= 'Member not saved.'
    When (Rc1 = 14)           Then Msg1= 'Member in use.   '
    When (Rc1 = 2Ø)           Then Msg1= 'Severe error.    '
    Otherwise                 Nop
   End
   «Ispexec Vput Msg1 Profile»
/*————————————————————————————————————————————————————————————————————*/
/* Set the ISPF statistics for the SOURCE dataset members that are    */
/* updated by the edit macro. (Changed members will have the |CHGMAC| */
/* string in their ID field. )                                        */
/*————————————————————————————————————————————————————————————————————*/
If RC3 = Ø Then
     Do
      «Alloc Fi(Stats)  Da(«Dsn»)     Shr Reuse»
      «Ispexec Lminit   Dataid(Sta)   Ddname(Stats) Enq(Shr)»
      «Ispexec Lmmstats Dataid(«Sta») Member(«Mem») User(Chgmac)»
      «Ispexec Lmfree   Dataid(«Sta»)»
     End
/*---------------------------------------------------------------------*/
/* Add a new row to the «Edit-macro Result Table».                     */
/*---------------------------------------------------------------------*/
  «Ispexec Tbadd» Table_name

 End  /* End-of-Loop-Martapv */
/*---------------------------------------------------------------------*/
/*  At last, display the «Edit-macro Result Table» and delete the      */
/*  virtual storage copy of this table.                                */
/*---------------------------------------------------------------------*/
    «Ispexec Tbtop»   Table_name
    «Ispexec Tbdispl» Table_name «Panel(Panel3)»
    «Ispexec Tbend»   Table_name

/*---------------------------------------------------------------------*/
/* Compare the SOURCE and BKPCPY datasets with the SuperC utility.     */
/*---------------------------------------------------------------------*/

«Alloc Fi(Newdd) Da(«Dsn»)     Shr Reuse»
«Alloc Fi(Olddd) Da(«Dsn_bkp») Shr Reuse»
«Alloc Fi(Outdd) Space(1,1) Cylinders New Keep Dsorg(Ps)»

/*---------------------------------------------------------------------*/
/* If RC is zero, it means that no change has been made to the SOURCE */
/* dataset. So we can delete the BKPCPY dataset and terminate the     */
/* dialog. If RC is not zero, in this case this means that some       */
/* changes have been made to SOURCE dataset.                           */
/*---------------------------------------------------------------------*/
Address Tso «Call 'Isp.Sisplpa(Isrsupc)' 'LONGL,LINECMP'»
 If Rc = Ø Then
             Do
                «Ispexec Setmsg Msg(EdtmØØ9I)»
                Delete Dsn_bkp
                Signal De_Alloc
             End

«Ispexec Lminit Dataid(Villar) Ddname(Outdd) Enq(Shr)»
 Rc7 = Rc
 If Rc7 ¬= Ø Then
             Do
                «Ispexec Vput Rc7 Profile»
                «Ispexec Setmsg Msg(EdtmØØ9G)»
               Exit
             End
/*---------------------------------------------------------------------*/
/* Call POPUP1 PROC to display an explanatory message.                 */
/*---------------------------------------------------------------------*/
Call Popup1
/*---------------------------------------------------------------------*/
```

```
/* Display the SuperC output to the user.                                  */
/*————————————————————————————————————————————————————————————————————————*/
«Ispexec Browse Dataid(«Villar»)»
/*————————————————————————————————————————————————————————————————————————*/
/* Call POPUP2 to ask the user if he/she agrees with the changes on   */
/* SOURCE dataset. If he/she does, BKPCPY dataset will be deleted.    */
/*————————————————————————————————————————————————————————————————————————*/
Call Popup2
De_Alloc:
«Ispexec Lmfree Dataid(«Villar»)»
«Free File(Newdd,Olddd,Outdd)»
«Altlib Deactivate Application(Exec)»
«Ispexec Libdef Ispplib»
«Ispexec Libdef Ispmlib»

 EXIT         /* End-of-the-main-Rexx */
POPUP1:
 Zwinttl =
 «Ispexec Addpop Row(6) Column(25)»
 «Ispexec Display Panel(Panel1)»
 «Ispexec Vget Spfkey Profile»
 «ispexec Rempop»
RETURN /* End-of-the-procedure-POPUP1 */
POPUP2:
 Zwinttl = CONFIRMATION SCREEN
 «Ispexec Addpop Row(6) Column(25)»
 «Ispexec Display Panel(Panel2)»
 «Ispexec Vget Spfkey Profile»
/*————————————————————————————————————————————————————————————————————————*/
/* Keep displaying the Panel2 until the user enters 'Y' or 'N'.       */
/*————————————————————————————————————————————————————————————————————————*/
 Do While( Spfkey=PFØ3 3 Spfkey = PFØ4 )
   «Ispexec Setmsg Msg(EdtmØØ9A)»
   «ispexec Rempop»
   «Ispexec Browse Dataid(«Villar»)»
   «Ispexec Addpop Row(6) Column(25)»
   «Ispexec Display Panel(Panel2)»
   «Ispexec Vget Spfkey Profile»
 End
 «ispexec Rempop»
 «Ispexec Vget Resp Profile»
 If Resp = 'Y' Then Do
                      Delete Dsn_bkp
                      «Ispexec Setmsg Msg(EdtmØØ9O)»
                   End
              Else «Ispexec Setmsg Msg(EdtmØØ9N)»
RETURN /* End-of-the-procedure-POPUP2 */

EDITMAC:
/*————————————————————————————————————————————————————————————————————————*/
/* Control if the member is still there.                                   */
/*————————————————————————————————————————————————————————————————————————*/
```

```
Rs = Sysdsn(Dsn»(«Mem»)»)
If Rs=»OK» Then Nop
           Else Do
                  Say 'The specified member is not found in dataset.'
                  Return
                End
«Ispexec Edit Dataset('«Dsn»(«Mem»)') Macro(ChgMac)»
Return Rc /* End-of-the-procedure-EDITMAC */
```

REXX : BATCHREX

```
/*REXX*/
/*——————————————————————————————————————————————————————————————————————*/
/* REXX program       : Batchrex                                      */
/* Edit macro called  : Batchmac                                      */
/* Purpose            : Change multiple strings in 3 libraries.    */
/*——————————————————————————————————————————————————————————————————————*/
«Prof nopref»
«Altlib  Activate Application(Exec) Da(Exp.Ctm.Rexx)»
Mpv.Ø = 1
Mpv.1 = Exp.Ctm.Test1
Mpv.2 = Exp.Ctm.Test2
Mpv.3 = Exp.Ctm.Test3
Do i = 1 to 3
 Dsn = Mpv.i
 x = Outtrap('Var.')
 «Listds» Dsn «Members»
 x = Outtrap('Off')       /* Turns trapping OFF */
/*————————————————————————————————————————————————————————————————————————*/
/* Get members of SOURCE dset and execute the edit-macro in a loop.  */
/*————————————————————————————————————————————————————————————————————————*/
Do j = 7  To Var.Ø
  Mem = Strip(Var.j)
  «Ispexec Edit Dataset('«Dsn»(«Mem»)') Macro(Batchmac)»
  End
End
«Altlib Deactivate Application(Exec)» /* Deallocate the REXX library */
Exit
```

REXX PROCEDURE : CHGIEBCP

```
/*REXX*/
/*————————————————————————————————————————————————————————————————————————*/
/* Allocate files needed by IEBCOPY.                                    */
/*————————————————————————————————————————————————————————————————————————*/
Arg Dsn1 Dsn2

«Alloc Fi(Input)    Da(«Dsn1») Shr Reuse»
«Alloc Fi(Output)   Da(«Dsn2») Shr Reuse»
«Alloc Fi(Sysout)   Dummy Reuse»
```

```
«Alloc Fi(Sysprint) Dummy Reuse»
«Alloc Fi(Sysut1)   Unit(VIO) Space(1,1) Cyl New Delete Reuse»
«Alloc Fi(Sysut2)   Unit(VIO) Space(1,1) Cyl New Delete Reuse»
«Alloc Fi(Sysut3)   Unit(VIO) Space(1,1) Cyl New Delete Reuse»
«Alloc Fi(Sysut4)   Unit(VIO) Space(1,1) Cyl New Delete Reuse»
/*——————————————————————————————————————————————————————————————————*/
/* Build the SYSIN control statements of the IEBCOPY utility.        */
/*——————————————————————————————————————————————————————————————————*/
«Alloc Fi(Sysin)    Unit(VIO) Space(1,Ø) Blksize(8Ø) Lrecl(8Ø),
 Recfm(F B) Dsorg(PS) New Delete Reuse»
 «Ispexec Lminit Dataid(Martapv) Ddname(Sysin) Enq(Exclu)»
 Rc5 = Rc
 If Rc5 ¬= Ø Then
              Do
                 «Ispexec Vput Rc5 Profile»
                 «Ispexec Setmsg Msg(EdtmØØ9E)»
                Exit
              End
            Else
              Do
                Card = ' COPY OUTDD=OUTPUT,INDD=INPUT'
                «Ispexec Lmopen  Dataid(«Martapv») Option(Output)»
                «Ispexec Lmput   Dataid(«Martapv»),
                Dataloc(Card) Datalen(8Ø) Mode(Invar)»
                «Ispexec Lmclose Dataid(«Martapv»)»
                «Ispexec Lmfree  Dataid(«Martapv»)»
                Rc4 = Rc
              End
 If Rc4 ¬= Ø Then Do
                   «Ispexec Vput Rc4 Profile»
                   «Ispexec Setmsg Msg(EdtmØØ9F)»
                   Exit
                 End
            Else Do
                   «Ispexec Select Pgm(IEBCOPY)»
                    Rc6 = Rc
                   If Rc6 ¬= Ø Then Do
                                     «Ispexec Vput Rc6 Profile»
                                     «Ispexec Setmsg Msg(EdtmØØ9H)»
                                     Exit
                                    End
/*——————————————————————————————————————————————————————————————————*/
/* Free all Ddnames of Iebcopy.                                      */
/*——————————————————————————————————————————————————————————————————*/
«Free File(Input,Output,Sysin,Sysprint,Sysut1,Sysut2,Sysut3,Sysut4)»
Exit                      /* End-of_procedure */
```

EDIT MACRO : CHGMAC

```
«Isredit Macro»
/*—————————————————————————————————————————————————————————————————*/
```

```
/*                                                                    */
/* Edit macro    : ChgMac                                             */
/* Called from   : ChgRexx                                            */
/* Purpose       : Change strings in a member of a given dataset.     */
/*                                                                    */
/*--------------------------------------------------------------------*/
Status = Msg('Off')
«Ispexec Control Errors Return»    /* Let EXEC process errors       */
Chg = Ø; Err = Ø                   /* Change-count, Error-count = Ø */
/*--------------------------------------------------------------------*/
/* Get the Number Mode.                                               */
/* Edit-macro has to take into account that if Number_Mode is 'ON'    */
/* then the number of editable characters is Lrecl-8. Because right- */
/* end side will include «sequence numbers». (NUMBER FIELDS)          */
/* So if the user enters a Column2 value that is bigger than          */
/* Lrecl-8, It is modified by this edit macro so that we will         */
/* prevent errors.                                                    */
/*--------------------------------------------------------------------*/
«Isredit (Mode) = Number»
/*--------------------------------------------------------------------*/
/* Get the variables from the Profile Variable Pool. These are the    */
/* strings which will be used on the 'Ispf Change' edit command.      */
/*--------------------------------------------------------------------*/
«Ispexec Vget (From,To,Mem,Qual,Col1,Col2,L2) Profile»
If ((Mode = ON) & (Col2>=(L2-8)))  Then  Col2 = L2-8
/*--------------------------------------------------------------------*/
/* Build the limiting keyword.                                        */
/*--------------------------------------------------------------------*/
Select
   When Qual = 1  Then Qual = CHARS
   When Qual = 2  Then Qual = PREFIX
   When Qual = 3  Then Qual = SUFFIX
   When Qual = 4  Then Qual = WORD
   Otherwise      Nop
End
If Length(From) = 1 Then
          «Isredit Change» «'«From»'» To «ALL» Qual Col1 Col2
                    Else
          «Isredit Change» From To «ALL» Qual Col1 Col2
          Rc2 = Rc
«Isredit (Chg,Err) = CHANGE_COUNTS»
Chg = Abs(Chg)
Err = Abs(Err)
If Rc2 = 4 Then Msg2='The string is not found.  '
If Rc2 = 8 Then Msg2='Str2 is longer than Str1. '
If Rc2 =12 Then Msg2='Inconsistent parameters.  '
If Rc2 =2Ø Then Msg2='Severe error.             '
If Rc2 = Ø Then Do
                Msg2='Change macro command Rc=Ø.'
                /*--------------------------------------------------*/
                /* If Error-Count is Ø, then we can save the    */
                /* current member in the SOURCE dataset.        */
                /*———————————————————————————————————————————————*/
                If Err= Ø Then Do
                               «Isredit Save»
                               Rc3 = Rc   /* Rc3 = Save Return Code */
                               End
                End
«Ispexec Vput (Chg,Err,Msg2,Rc2,Rc3) Profile»

/*———————————————————————————————————————————————————————————————————————*/
/* No matter it was saved or not, exit from the member to be able to  */
/* process the next member in the SOURCE dataset.                     */
/*———————————————————————————————————————————————————————————————————————*/
«Isredit Cancel»
Return
```

EDIT MACRO : BATCHMAC

```
«Isredit Macro»
«Isredit Change V312 V41Ø ALL»
«Isredit Change '.LIB312' '.LIB41Ø' ALL»
«Isredit Save»
«Isredit Cancel»
```

ISPF PANEL : PANEL0

```
)ATTR
    < TYPE(INPUT) INTENS(HIGH) COLOR(YELLOW) CAPS(OFF)
    { TYPE(INPUT) INTENS(HIGH) COLOR(GREEN)  CAPS(OFF)
    } TYPE(TEXT)  INTENS(HIGH) COLOR(PINK)
    $ TYPE(TEXT)  INTENS(HIGH) COLOR(PINK)
    % TYPE(TEXT)  INTENS(HIGH) COLOR(TURQ)
    + TYPE(TEXT)  INTENS(LOW)
    _ TYPE(INPUT) INTENS(HIGH)
    # TYPE(INPUT) INTENS(HIGH) COLOR(WHITE)
    [ TYPE(TEXT)  INTENS(HIGH) COLOR(RED) HILITE(REVERSE)
)BODY
%————————————[ AUTOMATED LIBRARY UPDATE %————————————-
%
+
% Please enter the dataset name on which you will make changes:
+ Dataset      %===>_Dsn                                         %
                                                               #Mes
% Please enter a «FROM» and a «TO» string: (Without quotation marks)
+ From string %===>{Z


                                  %
+
+ To   string %===>‹Z


                                    %
+
%Please qualify the search string :(Default is «CHARS»)
%(‹Z%) $1-CHARS  2-PREFIX  3-SUFFIX  4-WORD %
+
% Please enter the Column Limitations : (Optional)
+ Column-1    %===>‹Z           + Column-2    %===>‹Z
}
}Hit%ENTER}to proceed. Hit%PF3}o%PFØ4}key to exit from the application.
)INIT
.CURSOR = DSN
.ZVARS = '( From To Qual Col1 Col2)'
&Qual = '1'

)REINIT
 Refresh(Mes)

)PROC
&MES = ''
&SPFKEY = .PFKEY
  Ver (&Dsn,Nonblank,Msg=EDTMØØØ)
  Ver (&Dsn,Dsname,Msg=EDTMØØ3)

  &A = Trunc(&Dsn,1)
  Ver (&A,Pict,'A')

  Ver (&From,Nonblank,Msm=EDTMØØ1)
  Ver (&To,Nonblank,Msg=EDTMØØ2)
  Ver (&Qual,Nonblank,Msg=EDTMØØ4)
  Ver (&Qual,List,1,2,3,4,Msg=EDTMØØ5)
  Ver (&Col1,Num,Msg=EDTMØØ7)
  Ver (&Col2,Num,Msg=EDTMØØ7)

IF (&Col1 > &Col2 & &Col2 NE '')
   .Msg = EDTMØØ9K

Vput(Spfkey From To Dsn Qual Col1 Col2 ) Profile
)END
```

ISPF PANEL : PANEL1

```
)ATTR
 % TYPE(TEXT)  COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
 @ TYPE(TEXT)  COLOR(RED)   CAPS(OFF) HILITE(USCORE)
 + TYPE(TEXT)  COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(TEXT)  COLOR(GREEN) HILITE(REVERSE)
)BODY WINDOW(65,9)
+
$                   ****  ATTENTION!  ****                 +
+                                                          +
+ You are about the  browse  the  listing of the comparison of +
+ SOURCE and BKPCPY datasets. Please check the changes that   +
+ have been made on the SOURCE dataset. After browsing, you   +
+ will be presented with another panel which asks if you agree +
+ with the changes.                                        +
%                    Hit@<Enter>%to continue.              +
)INIT
)PROC
&SPFKEY = .PFKEY
   Vput Spfkey Profile
)END
```

ISPF PANEL : PANEL2

```
)ATTR
 % TYPE(TEXT)  COLOR(WHITE) CAPS(OFF) HILITE(USCORE) INTENS(LOW)
 @ TYPE(TEXT)  COLOR(RED)   CAPS(OFF) HILITE(USCORE)
 + TYPE(TEXT)  COLOR(TURQ)  CAPS(OFF) JUST(LEFT)
 $ TYPE(TEXT)  COLOR(GREEN) HILITE(REVERSE)
 [ TYPE(INPUT) COLOR(PINK)  CAPS(ON)  HILITE(REVERSE)
)BODY WINDOW(32,9)
+
$       DO YOU ACCEPT THE
$        CHANGES ON THE
$        SOURCE DATASET?
+
+ Yes(YS) / No(N)......:[Z+
+
% Hit@<Enter>%to proceed.+
)INIT
.Zvars = '( Resp )'
&Resp = 'N'
.Cursor = Resp
)PROC
&Spfkey = .Pfkey
Vput Spfkey Profile
Ver  (&Resp,Nonblank,Msg=EDTM009L)
Ver  (&Resp,List,Y,N,Msg=EDTM009M)
Vput (Resp) Profile
)END
```

ISPF PANEL : PANEL3

```
)ATTR DEFAULT(%+_)
     @ TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
     % TYPE(TEXT)   INTENS(HIGH) COLOR(TURQ)
     $ TYPE(OUTPUT) INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) CAPS(OFF)
     * TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(RED)
     # TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
     [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)
)BODY
%-------------[ AUTOMATED LIBRARY UPDATE %------------
%                              [ EXECUTION RESULTS %
+
%Command ===>_ZCMD
+
* DATASET     :$DSN                                       *DATE
:+&ZDATE
* MEMBER-COUNT :$CNT*                                      *USER-ID
:+&ZUSER
* FROM         :$FROM
* TO           :$TO
* COL1 & COL2  :$COL1  *$COL2      +
+
* NOTE:#R2 = Isredit Change command Rc.  R1 = Ispexec Edit command Rc.
+              Hit#<F1>+to get more info on Return Codes.
+
+%MEMBER  %CHANGE%ERROR%MESSAGE1         %RC%RC%MESSAGE2
+%NAME    %COUNT %COUNT%                 %1 %2 %
+%----%---%--%-------- %-%-%-------------
+
)MODEL
 @MEM     @CHG   @ERR  @MSG1             @Z @Z @MSG2
)INIT
 &Zcmd =''
.Help = Panelh
.Zvars = '(Rc1 Rc2)'
)PROC
)END
```

ISPF PANEL : PANELH

```
)ATTR DEFAULT(%+_)
     % TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN)
     < TYPE(TEXT)   INTENS(HIGH) COLOR(PINK)  HILITE(REVERSE)
     * TYPE(TEXT)   INTENS(HIGH) JUST(ASIS)   COLOR(RED)
     [ TYPE(TEXT)   INTENS(HIGH) COLOR(GREEN) HILITE(REVERSE)
)BODY
%-------------[ AUTOMATED LIBRARY UPDATE %------------
%
*  Rc1 Message1   («Ispexec Edit Dataset» command Return Codes)
*  === ==================================================
+  Ø   Member updated.
+  4   Member not saved. / No save. Abend S37'
+  14  Member in use.
+  2Ø  Severe error.
+
<  NOTE :+In case you see «S37» on the Message1 text, it's necessary
+        +to compress or enlarge the SOURCE dataset.
+
+
*  Rc2   Message2 («Isredit Change» command Return Codes)
*  ===== ================================================
+  Ø    Change macro command normal completion.
+  4    The string is not found.
+  8    String-2 is longer than String-1 and
+       substitution was not performed on at least one change.
+ 12    Inconsistent parameters. The string to be found does
+       not fit between the specified columns.
+ 2Ø    Severe error.
+
)INIT
)PROC
)END
```

ISPF MESSAGES : EDTM00

```
* EDTMØØ MESSAGE DEFINITIONS
*
*
EDTMØØØ  'Enter a dataset name. '      .HELP=*          .ALARM=YES
'Enter a dataset in which you''d like to run an edit macro for its each member'
*
EDTMØØ1  'Enter a FROM string.   '      .HELP=*          .ALARM=YES
'Enter a FROM string that will be used for the ''ISPF Change '' Edit command.'
*
EDTMØØ2  'Enter a TO string.     '      .HELP=*          .ALARM=YES
'Enter a TO string that will be used for the ''ISPF Change '' Edit command.'
*
EDTMØØ3  'Invalid Dset-qualifier.'     .HELP=*          .ALARM=YES
'Each qualifier must be 1-8 alphanumeric chars  & first one must be alphabetic.'
*
EDTMØØ4  'Enter 1, 2, 3, or 4.'         .HELP=*          .ALARM=YES
'Enter a number between 1 and 4 which corresponds to the qualifying keyword.'
*
EDTMØØ5  'Out of range.'                .HELP=*          .ALARM=YES
'You should enter a value between 1 and 4 for this field.'
*
EDTMØØ6 'No changes are made.    '      .HELP=*          .ALARM=YES
'No any changes are made in the specified SOURCE dataset.'
*
EDTMØØ7 'Enter a numeric value.  '      .HELP=*          .ALARM=YES
'Column limitation that will be used for ''CHANGE'' command, must be numeric.'
*
EDTMØØ8 '«FROM» string > Lrecl.  '      .HELP=*          .ALARM=YES
'«FROM» string (&L1) can not be longer than the dataset''s Record Length (&L2)'
```

```
*
EDTMØØ9 '«TO» string > Lrecl.'          .HELP=*          .ALARM=YES
'«TO» string (&L1) cannot be longer than the dataset's Record Length (&L2)'
*
EDTMØØ9A 'Hit <PFØ3+Enter> '            .HELP=*          .ALARM=YES
'Hit <Enter> then <PFØ3> to exit from the Comparison output.'
*
EDTMØØ9B 'PO dset has no member.'       .HELP=*          .ALARM=YES
'The PO dset you entered has no member. Please enter it again.'
*
EDTMØØ9C 'Dset does not exist.'         .HELP=*          .ALARM=YES
'The PO dset you entered does not exist. Please enter it again.'
*
EDTMØØ9D 'Dataset is not PO.'          .HELP=*          .ALARM=YES
'The PO dset you entered is not a PO. Please enter it again.'
*
EDTMØØ9E 'Lminit Rc = &Rc5    '         .HELP=*          .ALARM=YES
'1st Lminit is not successful.'
*
EDTMØØ9F 'Lmfree error.'                .HELP=*          .ALARM=YES
'Lmfree error. Rc = &Rc4'
*
EDTMØØ9G 'Lminit Rc = &Rc7    '         .HELP=*          .ALARM=YES
'2nd Lminit is not successful.'
*
EDTMØØ9H 'Iebcopy error.'               .HELP=*          .ALARM=YES
'IEBCOPY Return Code =  &Rc6'
*
EDTMØØ9I 'No change was made.'          .HELP=*          .ALARM=YES
'The dataset was not updated, since there was no string change.'
*
EDTMØØ9J 'Col value is incorrect.'      .HELP=*          .ALARM=YES
'Col1 or Col2 can''t be bigger than the Lrecl of the SOURCE dataset.'
*
EDTMØØ9K 'Col2 must be bigger.'         .HELP=*          .ALARM=YES
'Col2 value must be bigger than Col1 value. Please enter it again.'
*
EDTMØØ9L 'Enter Y or N.       '         .HELP=*          .ALARM=YES
'This field must be either Y or N.'
*
EDTMØØ9M 'Enter Y to accept.  '         .HELP=*          .ALARM=YES
'Enter ''Y'' to accept changes on SOURCE dset and delete BKPCPY dset.'
*
EDTMØØ9N 'BKPCPY dset kept.'            .HELP=*          .ALARM=YES
'BKPCPY dataset is kept. SOURCE dataset has now changes.'
*
EDTMØØ9O 'BKPCPY dset deleted.'         .HELP=*          .ALARM=YES
'BKPCPY dataset is deleted. SOURCE dataset has now changes.'
*
```

Compressing contiguous characters in a file

There are many files that contain a large number of spaces. Program source codes, output listings, and others have a large percentage of contiguous spaces. For transmission purposes, archiving, or other reasons, one might want to compress them to reduce their size.

The two programs shown below do just that. The first, ZIPCHAR, takes a contiguous sequence of a given character and reduces it to a single byte. The reverse program, UNZIPCHA, does the opposite operation and restores the file. The character is the space by default, but you can choose any other character simply by passing it as a parameter to ZIPCHAR. For example, low-value can also be a good candidate for compression, in certain files. If the character you want to compress is not displayable, as is the case with low-value, then you must edit your JCL in hexadecimal to insert it correctly as a parameter.

The technique is simple. The program reads an input file sequentially, as a continuous stream of bytes, and produces an output file. If there are three or more contiguous spaces (or whatever character you requested), then it replaces them by a control byte whose lower seven bits represent the number of contiguous spaces, up to 127. The high-order bit acts as a flag, and in this case it is zero. If there are more than 127 contiguous spaces, a second control byte is used to count for the additional ones, and so on.

All the other characters in the file are preceded by a control byte whose high-order bit is one and whose remaining seven bits represent the number of 'literal' characters that follow. If there are more than 127 non-space characters, then another control byte is used, and so on. This count does not include the control byte itself. When the input file ends, a zero contol byte is written, to mark the end of it; in fact, a zero fullword. The output file is always a fixed 80-byte file. The input file must be a non-VSAM fixed-length file, with any LRECL up to 32KB.

It is your responsibility to know the correct LRECL of the original file when you uncompress it. The compressed file has no information whatsoever about the original record length or where each record ends; it only contains a stream of bytes. The first byte of the compressed file is the character that was compressed. This way the uncompress program always knows what character to expand, so it needs no parameter.

You can compress an already compressed file for a different character. For example, you can compress a file for spaces and then for zeros. In this case, compress first for the character that occurs more, and then for the other(s), to achieve a better compression ratio.

Below is an example job for a double compression. The input file is assumed to have a record length of 2,500. The first step is for the space, the default, so no parameter is needed. The second step is for character zero, assuming the input file has a lot of contiguous spaces and zeros. The expansion job is also shown. Note that in this case no PARM is needed.

```
//JOB11 JOB REGION=1000K,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*        Example for space and "0" compression
//*
//JOBLIB   DD  DISP=SHR,DSN=my.loadlib
//STEP1    EXEC PGM=ZIPCHAR
//SYSPRINT DD  SYSOUT=*
//INFILE   DD  DISP=SHR,DSN=my.input.file
//OUTFILE  DD  DSN=&&TEMP1,DISP=(NEW,PASS),LRECL=80,RECFM=FB,
//             BLKSIZE=27200,SPACE=(TRK,(300,300),RLSE),UNIT=SYSDA
//*
//STEP2    EXEC PGM=ZIPCHAR,PARM='0'
//SYSPRINT DD  SYSOUT=*
//INFILE   DD  DISP=(OLD,DELETE),DSN=&&TEMP1
//OUTFILE  DD  DSN=my.compressed.file,DISP=(NEW,CATLG,DELETE),
//             SPACE=(TRK,(200,200),RLSE),UNIT=SYSDA,
//             LRECL=80,RECFM=FB,BLKSIZE=27200
//*

//JOB22 JOB REGION=1000K,MSGCLASS=X,MSGLEVEL=(1,1)
//*
//*        Example for restoring a twice-compressed file
//*
//JOBLIB   DD  DISP=SHR,DSN=my.loadlib
//STEP1    EXEC PGM=UNZIPCHA
//SYSPRINT DD  SYSOUT=*
//INFILE   DD  DISP=SHR,DSN=my.compressed.file
//OUTFILE  DD  DSN=&&TEMP1,DISP=(NEW,PASS),LRECL=80,RECFM=FB,
//             BLKSIZE=27200,SPACE=(TRK,(300,300),RLSE),UNIT=SYSDA
//*
//STEP2    EXEC PGM=UNZIPCHA
//SYSPRINT DD  SYSOUT=*
//INFILE   DD  DISP=(OLD,DELETE),DSN=&&TEMP1
```

```
//OUTFILE  DD  DSN=my.restored.file,DISP=(NEW,CATLG,DELETE),
//             LRECL=2500,RECFM=FB,BLKSIZE=27500,
//             SPACE=(TRK,(600,600)),RLSE),UNIT=SYSDA
```

ZIPCHAR SOURCE

```
*====================================================================*
* ZIPCHAR - Reads an INPUT file and writes a compressed OUTPUT file  *
*           The char indicated by "&CHAR" is compressed, by default  *
*           Optional parameter: char to compress. The compressed     *
*           char is written in the first byte of the output file.    *
*           Decompress program: UNZIPCHR.                            *
*====================================================================*
&PROGRAM SETC  'ZIPCHAR'
&CHAR    SETC  '40'                 Hexadecimal of default char
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         USING IHADCB,R11
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     GETPARM
         DC    CL16' &PROGRAM 1.6'
         DC    CL8'&SYSDATE'
*
GETPARM  DS    0F
         LR    R2,R1
         OPEN  (SYSPRINT,OUTPUT)
         L     R2,0(0,R2)
         L     R3,0(R2)
         LR    R9,R3
         SRL   R9,16                   Get parm length in R9
         LTR   R9,R9                   Any parm?
         BZ    OPENFILE                No
         SRL   R3,8                    Parm in lower byte of R3
         STC   R3,COMPRE               Store it
         CLI   COMPRE,X'&CHAR'         Is parm the default character?
         BE    OPENFILE                Yes
         MVC   CHANGE1+1(1),COMPRE     No, change CLI instructions.
         MVC   CHANGE2+1(1),COMPRE
         MVC   CHANGE3+1(1),COMPRE
         MVC   CHANGE4+1(1),COMPRE
*
OPENFILE EQU   *
         SR    R7,R7
         OPEN  (INFILE,INPUT)
         LTR   R15,R15
         BNZ   ERRO1
         LA    R11,INFILE          Address IHADCB of input file.
         LH    R1Ø,DCBLRECL        Get maximum length
         LA    R1Ø,4(Ø,R1Ø)        Keep maximum length plus 4
         ST    R1Ø,MAXLRECL
         OPEN  (OUTFILE,OUTPUT)
         LTR   R15,R15
         BNZ   ERRO2
*
         STORAGE OBTAIN,           2*32K plus some safety margin       X
               LENGTH=68ØØØ,                                           X
               ADDR=(R2)
         ST    R2,POINTER1
         A     R2,=F'32768'        Just one 32k record as limit
         ST    R2,POINTER2
*
         LA    R5,OUTBUF1          R5: Address output buffer
         L     R3,POINTER1         R3: Current byte at inbuffer
         XR    R4,R4               R4: Bytes available at inbuffer
         XR    R6,R6               R6: Number chars in out buffer
         XR    R8,R8               R8: Char or space counter
         MVC   Ø(1,R5),COMPRE      Put compressed char in outbuf
         LA    R5,1(Ø,R5)          Inc output pointer
         LA    R6,1(Ø,R6)          And output counter
         BAL   R1Ø,READIN          Go read some records to inbuffer
*=====================================================================*
*  Paragraphs Bnn deal with compressed char, Cnn deal with the others
*=====================================================================*
CØØØ     EQU   *                   Deal with characters:
         BAL   R1Ø,WSPACES         Write spaces first
CØØ      EQU   *                   If next three bytes are spaces,
CHANGE1  CLI   Ø(R3),X'&CHAR'      branch to spaces.
         BNE   CØ1                 If not, consider them chars
CHANGE2  CLI   1(R3),X'&CHAR'
         BNE   CØ1
CHANGE3  CLI   2(R3),X'&CHAR'
         BE    BØØØ
CØ1      EQU   *
         CH    R8,=H'127'          Counter limit attained?
         BL    CØ2                 No, jump
         BAL   R1Ø,WCHARS          Yes, write out chars
CØ2      EQU   *
         LA    R8,1(Ø,R8)          Inc char number
         LA    R3,1(Ø,R3)          Inc current byte pointer
         SH    R4,=H'1'            Dec input buffer still available
         LTR   R4,R4               End of buffer and of file?
         BE    ENDCHARS            Yes, go deal with what is left.
         CH    R4,=H'13Ø'          Input buffer nearing 127?
         BH    CØ3                 No
         BAL   R1Ø,READIN          Yes, read more records
CØ3      EQU   *
```

```
         B     CØØ                 Continue
BØØØ     EQU   *                   Deal with spaces:
         BAL   R1Ø,WCHARS          write characters first
BØØ      EQU   *
CHANGE4  CLI   Ø(R3),X'&CHAR'      Current byte space?
         BNE   CØØØ                No, jump to chars
         CH    R8,=H'127'          Counter limit attained?
         BL    BØ2                 No
         BAL   R1Ø,WSPACES         Yes, write spaces
BØ2      EQU   *
         LA    R8,1(Ø,R8)          Inc space counter
         LA    R3,1(Ø,R3)          Inc current byte
         SH    R4,=H'1'            Dec available buffer
         LTR   R4,R4               End of buffer and file?
         BE    ENDSPACE            Yes, go deal with what is left
         CH    R4,=H'13Ø'          Input buffer nearing 127?
         BH    BØ3                 No
         BAL   R1Ø,READIN          Yes, read more records
BØ3      EQU   *
         B     BØØ                 Continue
*=====================================================================*
*  After input file ended, write out what's left and exit
*=====================================================================*
ENDSPACE EQU   *
         BAL   R1Ø,WSPACES         Put remaining spaces
         B     ENDALL
ENDCHARS EQU   *
         BAL   R1Ø,WCHARS          Put remaining chars in buffer
         B     ENDALL
*
ENDALL   EQU   *
         XR    R1,R1
         ST    R1,Ø(R5)            Put a zero fullword as eof mark
         LA    R6,4(Ø,R6)          Add 4 bytes to output
         BAL   R8,WRITEOUT         Write last buffer or two
         B     EXIT1               and exit
*
EXIT1    EQU   *
         L     R2,POINTER1
         STORAGE RELEASE,                                              X
               LENGTH=68ØØØ,                                           X
               ADDR=(R2)
         CLOSE INFILE
         CLOSE OUTFILE
         CLOSE SYSPRINT
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         XR    R15,R15
         BR    R14
*=====================================================================*
*        Subroutines
```

```
*======================================================================*
WCHARS   EQU   *                     Write out (R8) chars
         LTR   R8,R8                 Zero chars?
         BZR   R1Ø                   Yes, go back
         STCM  R8,B'ØØØ1',Ø(R5)      Store length
         OI    Ø(R5),X'8Ø'           First bit on indcates chars.
         LA    R5,1(Ø,R5)            Inc output pointer
         LA    R6,1(Ø,R6)            Inc output counter
         SR    R3,R8                 R3 points chars to move
         SH    R8,=H'1'
         EX    R8,MVC2               move chars
         LA    R8,1(Ø,R8)            restore length
         AR    R3,R8                 Restore current pointer
         AR    R5,R8                 Advance output pointer
         AR    R6,R8                 R6 number of bytes in outbuf
         CH    R6,=H'8Ø'             record complete?
         BL    WCHARS1               Not yet
         BAL   R8,WRITEOUT           Yes, write out file
WCHARS1  EQU   *
         XR    R8,R8                 Reset zero length
         BR    R1Ø                   Return
WSPACES  EQU   *                     Write out (R8) spaces
         LTR   R8,R8                 If zero, return
         BZR   R1Ø
         STCM  R8,B'ØØØ1',Ø(R5)      write lower byte of R8
         LA    R5,1(Ø,R5)            Inc out pointer
         LA    R6,1(Ø,R6)            Inc number of bytes in outbuf
         XR    R8,R8                 Reset zero length
         CH    R6,=H'8Ø'             record complete?
         BL    WSPACES1              Not yet
         BAL   R8,WRITEOUT           Yes, write out file
WSPACES1 EQU   *
         XR    R8,R8                 Reset zero length
         BR    R1Ø                   Return
READIN   EQU   *                     Read input file subroutine
         CLI   ENDINPUT,C'1'         Input file ended?
         BER   R1Ø                   Yes, return
         LR    R9,R3                 Current input location
         AR    R9,R4                 Add remaining input for next free.
         L     R2,POINTER2           End of read area
         SR    R2,R9                 R2 = Remaining read area
         CH    R2,MAXLRECL           Space for another record?
         BH    READIN1               Yes, jump around move
*                                    Move data to beginning of area:
         L     R9,POINTER1           Point beginning of area
         LR    R7,R3                 Current byte
         SR    R7,R8                 R7 initial byte to move
         SH    R8,=H'1'              Prepare move length
         EX    R8,MVC1               Move data before R3
         LA    R8,1(Ø,R8)            Restore length
         AR    R9,R8                 Point after data moved
         LR    R7,R3                Move data starting at R3
         SH    R4,=H'1'             Prepare move length (R4)
         EX    R4,MVC1              Move data after R3
         LA    R4,1(Ø,R4)           Restore length
         L     R3,POINTER1          Beginning of area
         AR    R3,R8                R3: new current location
         B     READIN
READIN1  EQU   *                    Read input file
         GET   INFILE,(R9)
         AH    R4,DCBLRECL          Add record leng to R4 total leng
         AH    R9,DCBLRECL          Add record leng to R9 pointer
         SH    R2,DCBLRECL          And subtract from space left (R2)
         CH    R2,MAXLRECL          Space for another record?
         BH    READIN1              Yes, jump around move
READEND  EQU   *
         BR    R1Ø                  Return
WRITEOUT EQU   *                    Write output file
         PUT   OUTFILE,OUTBUF1      Write buffer1
         C     R6,=F'16Ø'           Second buffer filled?
         BL    WRITEOU1             No, jump ahead
         PUT   OUTFILE,OUTBUF2      Write buffer 2
         MVC   OUTBUF1,OUTBUF3      Move back buffer 3
         S     R6,=F'16Ø'           from length and pointer
         LA    R5,OUTBUF1
         AR    R5,R6
         BR    R8                   Return
WRITEOU1 EQU   *
         MVC   OUTBUF1,OUTBUF2      Move back buffer 2
         S     R6,=F'8Ø'            Subtract 8Ø bytes
         LA    R5,OUTBUF1
         AR    R5,R6
         BR    R8                   Return
*
ERRO1    EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file'
         B     EXIT1
ERRO2    EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening output file'
         B     EXIT1
ENDFILE  EQU   *
         MVI   ENDINPUT,C'1'        Set eof flag
         B     READEND              Return of read subroutine
*
INFILE   DCB   DSORG=PS,MACRF=(GM),                                    X
               EODAD=ENDFILE,                                          X
               DDNAME=INFILE
*
OUTFILE  DCB   DSORG=PS,MACRF=(PM),                                    X
               LRECL=8Ø,                                               X
               DDNAME=OUTFILE
*
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                    X
```

```
               LRECL=8Ø,                                              X
               DDNAME=SYSPRINT
*
         LTORG
         DS    ØF
MVC1     MVC   Ø(Ø,R9),Ø(R7)
MVC2     MVC   Ø(Ø,R5),Ø(R3)
ENDINPUT DC    C'Ø'                End of input file flag
COMPRE   DC    X'&CHAR'            Char to compress
SAVEA    DS    18F
POINTER1 DS    F                   Beg of read storage
POINTER2 DS    F                   End of read storage
MAXLRECL DS    H                   Max LRECL declared for file
OUTBUF1  DS    CL8Ø
OUTBUF2  DS    CL8Ø
OUTBUF3  DS    CL8Ø
*
         DCBD  DSORG=PS
         YREGS
         END
```

UNZIPCHA SOURCE

```
*=====================================================================*
* UNZIPCHA - Expands files compressed by ZIPCHAR. Reads an INPUT      *
*            compressed 8Ø byte file and writes an OUTPUT.            *
*            Parameter: none.                                         *
*=====================================================================*
&PROGRAM SETC  'UNZIPCHA'
&CHAR    SETC  '4Ø'                Default char
&PROGRAM AMODE 31
&PROGRAM RMODE 24
&PROGRAM CSECT
         SAVE  (14,12)
         LR    R12,R15
         USING &PROGRAM,R12
         USING IHADCB,R11
         ST    R13,SAVEA+4
         LA    R11,SAVEA
         ST    R11,8(R13)
         LR    R13,R11
         B     OPENFILS
         DC    CL16' &PROGRAM 1.6'
         DC    CL8'&SYSDATE'
*
OPENFILS DS    ØF
         OPEN  (SYSPRINT,OUTPUT)
         OPEN  (INFILE,INPUT)
         LTR   R15,R15
         BNZ   ERRO1
         OPEN  (OUTFILE,OUTPUT)
         LTR   R15,R15
         BNZ   ERRO2
         LA    R11,OUTFILE         Address IHADCB of input file.
         MVC   MAXLRECL,DCBLRECL   Keep maximum length
*
         STORAGE OBTAIN,            Enough for 32k                        X
               LENGTH=33ØØØ,                                              X
               ADDR=(R2)
         ST    R2,POINTER1
         L     R3,POINTER1        R3: current output pointer
         XR    R4,R4              R4: Bytes at outbuf
         XR    R6,R6              R6: Bytes at input
         BAL   R1Ø,READIN         Read first record
         MVC   COMPRE,Ø(R5)       First byte is the compressed char
         LA    R5,1(Ø,R5)         Inc input pointer
         SH    R6,=H'1'           Decrease input counter
         CLI   COMPRE,X'&CHAR'    Default char?
         BE    TESTBYTE           Yes
         MVC   CHANGE1+1(1),COMPRE  No, change MVI instruction
*
TESTBYTE EQU   *
         MVC   BYTE,Ø(R5)         Get first byte
         LA    R5,1(Ø,R5)         Advance input pointer
         SH    R6,=H'1'           Decrease input counter
         CLI   BYTE,X'ØØ'         Byte is null?
         BE    EXIT1              Yes, exit
         CLI   BYTE,X'8Ø'         High-order bit 1?
         BL    SPACES             No
*
BYTES    EQU   *
         NI    BYTE,X'7F'         Reduce by 128
         XR    R7,R7              Clear R7
         IC    R7,BYTE            Get value at byte
*
BYTES1   EQU   *
         CR    R7,R6              More chars than left in record?
         BNH   BYTES2             No
         SH    R6,=H'1'           Decrease for execute
         EX    R6,MVC1            Move R6 characters
         LA    R6,1(Ø,R6)         Restore
         AR    R4,R6              Inc output chars (R4)
         AR    R3,R6              Inc output pointer (r3)
         SR    R7,R6              Length that remains yet
         BAL   R1Ø,READIN         Go read more
         B     BYTES1
*
BYTES2   EQU   *
         SH    R7,=H'1'           Decrease for execute
         EX    R7,MVC1            Move R7 characters
         LA    R7,1(Ø,R7)         Restore
         AR    R3,R7              Inc output pointer (r3)
         AR    R4,R7              Inc output chars (R4)
         AR    R5,R7              Inc input pointer
         SR    R6,R7              Decrease input
         B     BYTEEND
*
SPACES   EQU   *
         XR    R7,R7              Clear R7
         IC    R7,BYTE            Get number of copies at byte
*
SPACES1  EQU   *
CHANGE1  MVI   Ø(R3),X'&CHAR'     Put compressed char
         LA    R3,1(Ø,R3)         Inc out pointer
         LA    R4,1(Ø,R4)         Inc out counter
         BCT   R7,SPACES1         Loop for number of spaces
*
BYTEEND  EQU   *
         CR    R5,R8              Input record ended?
         BL    BYTEEND1           No
         BAL   R1Ø,READIN         Yes, read more
*
BYTEEND1 EQU   *
         BAL   R1Ø,TESTOUT        Test for output record
         B     TESTBYTE
*
EXIT1    EQU   *
         L     R2,POINTER1
         STORAGE RELEASE,                                              X
               LENGTH=33ØØØ,                                           X
               ADDR=(R2)
         CLOSE INFILE
         CLOSE OUTFILE
         CLOSE SYSPRINT
         L     R13,SAVEA+4
         LM    R14,R12,12(R13)
         XR    R15,R15
         BR    R14
*===================================================================*
*        Subroutines
*===================================================================*
TESTOUT  EQU   *
         CH    R4,DCBLRECL        Out record complete?
         BLR   R1Ø                Not yet, return
         PUT   OUTFILE,(R2)       Record complete, write it
         SH    R4,DCBLRECL        See whats left
         LTR   R4,R4              Anything?
         BZ    TESTOUT1           No, return
         SR    R3,R4              Move pointer back
         SH    R4,=H'1'
         EX    R4,MVC2            Move extra to beginning of new rec
         LA    R4,1(Ø,R4)
*
TESTOUT1 EQU   *
         LR    R3,R2              Restore current pointer
         AR    R3,R4               Add excess, if any
         B     TESTOUT             and go see if there are more
*
READIN   EQU   *                   Read input
         CLI   ENDINPUT,C'1'       File ended?
         BER   R1Ø                 yes, return
         GET   INFILE              get locate
         LR    R5,R1               R5 input pointer
         LA    R8,8Ø(Ø,R5)         R8 End of input record
         LA    R6,8Ø               R6 8Ø bytes available
         BR    R1Ø                 return
ERRO1    EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening input file'
         B     EXIT1
ERRO2    EQU   *
         PUT   SYSPRINT,=CL8Ø'>>> Error opening output file'
         B     EXIT1
ENDFILE  EQU   *
         MVI   ENDINPUT,C'1'       Set eof flag
         BR    R1Ø
INFILE   DCB   DSORG=PS,MACRF=(GL),                                    X
               EODAD=ENDFILE,                                          X
               DDNAME=INFILE
OUTFILE  DCB   DSORG=PS,MACRF=(PM),                                    X
               DDNAME=OUTFILE
SYSPRINT DCB   DSORG=PS,MACRF=(PM),                                    X
               LRECL=8Ø,                                               X
               DDNAME=SYSPRINT
         LTORG
         DS    ØF
MVC1     MVC   Ø(Ø,R3),Ø(R5)
MVC2     MVC   Ø(Ø,R2),Ø(R3)
BYTE     DS    C
ENDINPUT DC    C'Ø'
COMPRE   DS    C
SAVEA    DS    18F
POINTER1 DS    F
POINTER2 DS    F
MAXLRECL DS    H
         DCBD  DSORG=PS
         YREGS
         END
```

Systems Programmer (UK) © Xephon 2001

Using REXX and the Web without OpenEdition

In *MVS Update* issues 168 and 169 (September and October 2000) I provided some code that allowed REXX to talk LPAR to LPAR (or machine) across TCP/IP using socket calls. While I was developing that function it had struck me that it would be nice to get the same function operating over the Web. Unfortunately I did not know anything about OpenEdition, and I had always assumed that this was required to achieve any form of Web development. Plus I did not have a clue about how to get from the Web to my server address space anyway. Then while reading an Infoman manual I came across a reference to setting up the Infoman Web server which happened to explain how to code the HTTP request to contact a particular server. Apparently this involves specifying the IP address of the relevant host followed by a ':' and the port number of the server address space. Out of curiosity I tried using this technique with the LPARANSR routine of the previous article. A connection occurred, but only 'rubbish' was returned. However, as it was clear that a connection was possible, all that was required now was to resolve the issues of EBCDIC to ASCII conversion, and how to create HTML for shipping down to a PC and for it to be possible to have a Web server without needing O/E and HFS files. This article shows the results of that development, which has resulted in a Web dialog for carrying out a number of functions such as TSO commands, DISKSPACE displays, and TAPE displays. Note that if you wish to implement this code with all functions active, then you will need the RVOLDATA program from the previous article for the DISKSPACE command. Furthermore, if you are Web-naive, as I was when starting this development, then it is worth reviewing that article for some other considerations to watch for when starting the REXX server.

OPERATION

In order to get this server working you will need to start the REXX code as a batch TSO command where the SYSPROC points to the HTMLREXX code, and where the STEPLIB contains the RVOLDATA, UCBTAPE, and, if necessary, the library containing the RXSOCKET load module (again refer to the previous article). To access the server, simply specify the IP:PORT address. Note that the

REXX code supplied contains a comment regarding the use of a META tag to initiate auto-refresh of the TAPE display screen. If this is re-instated to the code and used in conjunction with the additional PC functionality to provide frames access, then it is possible to have one PC screen which ‘looks’ at multiple servers and keeps updating its screens. Overall such functionality may be of most use if using the supplied code as a means of providing your own Web functionality.

This article includes the following items:

- Sample screens as displayed from the server. Please note that the screens shown in this article are modified to keep my site information confidential, and as such what is shown is produced from a display of the HTML and not from the server. However, the format is identical.
- The HTMLREXX source code (this is the actual server code).
- A sample job for starting the server – note that the previous article also includes similar jobs and it also includes examples of FTP jobs for starting the server on multiple machines/LPARs at one go.
- The source code for the tape information retrieval module (UCBTAPE).
- An example of a PC based HTML routine that can be used to initiate frames access over a browser to pull multiple servers together on one screen (five machines are brought together in the example). Note these are not required for basic PC-to-mainframe communication, they are just a potentially useful add-on.

HTMLREXX

```
/* REXX */
/*******************************************************************/
/* This is the basic server REXX for developing Web serving routines.*/
/* It is called from the Web and is passed a command as part of a    */
/* CGI script in the form /?Command=etc. Where etc. is the command   */
/* that is actually required. This command string is then parsed to  */
/* determine what is required.                                        */
/*                                                                    */
/* If the string DISKSPACE is passed then information on the info    */
/* on the currently on-line DASD is returned.       .               */
/*                                                                  */
/* If the string TAPEINFO is passed then information on the info     */
/* on the current tape set-up is returned.          .               */
/*                                                                  */
/* If the string commences with TSO then all that follows this will  */
/* be issued as a TSO command and the data trapped and returned to   */
/* the caller (eg the results of a LISTC UCAT).                   */
/*                                                                  */
/* If the string SHUTDOWN is passed then the server will terminate   */
/*                                                                  */
/* If none of the above is passed then a basic Web page is served to */
/* request a command from a user.                                   */
/*                                                                  */
/* Once the command has been obtained and the CGI bits parsed, the   */
/* remaining string will be parsed based on the assumption that it   */
/* will be of the following form:                                   */
/* label (action)                                                   */
/*                                                                  */
/* Where label is the routine to invoke, and action is the argument  */
/* to that label. See the SELECT statement later to see how to       */
/* implement additional function in this server.                    */
/********************************************************************/
/* */
/* need to trap possible syntax errors in case of incorrect parms */
/* being passed.                                                 */
/* */
SIGNAL ON syntax
/* */
linecount.=Ø
/* initialize control information                                    */
port = '1952'                /* The port used for the service          */
/* */
/* now obtain the name of the LPAR this server is running on */
/* */
CVTECVT=D2X(C2D(STORAGE(1Ø,4))+14Ø) /* point to cvtsysad */
lparname=STRIP(STORAGE(D2X(C2D(STORAGE(CVTECVT,4))+344),8))
/* Begin setup                                                       */
SAY 'RSSERVER: initializing'
/* */
/* a call to scoket will return a string which gives an rcode */
/* followed by the unique name for this task (in this case    */
/* RSSERVER) followed by the maximum number of tasks and      */
/* finally the name of the IP started task.                   */
/* */
x= 'SOCKET'('Initialize','RSSERVER')
IF WORD(x,1)¬='Ø' THEN DO
   SAY 'ERROR while initialising'
   EXIT
   END
/* */
/* We now need to get the host IP address. This is done with a  */
/* gethostid request. In a similar manner to other requests the */
/* first character returned is a success or failure indicator   */
/* and in this case the second word is the IP address           */
/* */
ipaddress='SOCKET'('GetHostId')
/* */
IF WORD(ipaddress,1)¬='Ø' THEN DO
   SAY 'ERROR while getting hostid'
   EXIT
   END
/* */
ipaddress=WORD(ipaddress,2)
/* */
SAY 'RSSERVER: initialized: ipaddress='ipaddress 'port='port
/* */
/* obtain a socket id. This is word 2 of the request. */
/* */
sock = 'SOCKET'('Socket')
/* */
IF WORD(sock,1)\='Ø' THEN DO
   SAY 'ERROR while getting socket'
   EXIT
   END
/* */
sock=WORD(sock,2)
/* */
/* In case IP hasn't cleared itself up by the time the server */
/* restarts, set the reuse option to prevent the server being */
/* unable to start.                                           */
/* */
x = 'SOCKET'('SetSockOpt',sock,'Sol_Socket','So_REUSEADDR','On')
/* */
/* now its time to issue a bind. Only a single character RC */
/* should be returned this time.                            */
/* */
x='SOCKET'('Bind',sock,'AF_INET' port ipaddress)
/* */
IF x¬=Ø THEN DO
   SAY 'error during af_inet'
   EXIT
   END
/* */
/* now time to listen. */
/* */
x='SOCKET'('Listen',sock)
/* */
IF x¬=Ø THEN DO
   SAY 'error during listen'
   EXIT
   END
/* */
/* now set the io control mode to with blocking. */
/* */
x='SOCKET'('Ioctl',sock,'FIONBIO','ON')
/* */
IF x¬=Ø THEN DO
   SAY 'error during set of io control mode'
   EXIT
   END
/* */
x='SOCKET'('Fcntl',sock,'F_SETFL','BLOCKING')
/* */
IF x¬=Ø THEN DO
   SAY 'error during set of io control mode'
   EXIT
   END
/* */
/* Wait for new connections and send lines. The array linecount will */
/* be used to keep track of data sent to each caller.                */
/* */
linecount. =  Ø
/* */
DO FOREVER
/* */
sellist='SOCKET'('SELECT','Write * Read * Exception')
/* */
PARSE UPPER VAR sellist . 'READ' rsock . 'WRITE' wsock . 'EXCEPTION' .
/* */
/* Now receive the information. If the socket id passed is the same */
/* as the one we are listening on, then we need to accept the       */
/* new connection.                                                  */
/* */
IF rsock¬='' THEN DO
   IF rsock=sock THEN DO
      x = 'SOCKET'('Accept',rsock)
      IF WORD(x,1)\='Ø' THEN DO
         SAY 'error adding another socket'
         EXIT
         END
      ELSE rsock=WORD(x,2)
      END
/* */
/* Ensure that the incoming data is converted from ASCII to EBCDIC */
/* */
   x='SOCKET'('SetSockOpt',rsock,'Sol_Socket','So_ASCII','On')
/* */
/* Then read the data */
/* */
   x='SOCKET'('Recv',rsock)
/* */
   PARSE VAR x x . user string
/* */
   IF x¬='Ø' THEN DO
      SAY 'Connection lost'
      x='SOCKET'('Close',rsock)
      END
   ELSE DO
        stringuser.rsock=user
        stringword.rsock=string
        SAY 'User' user 'issued command' string 'at' TIME() DATE('E')
   END
END
/* */
/* Retrieve the command for this socket request and build the    */
/* information in the variable aray msg.wsock.msgnum.            */
/* It is assumed that RESULT will contain the number of lines    */
/* to return to the caller upon return from the subroutine.      */
/* If it doesn't then 1 line to return is assumed.               */
/* Note that by chaining the data to minimize the number of SENDs*/
/* the server will perfom much more effectively. Hence minimize  */
/* the array count accordingly.                                  */
/*                                                               */
/* The array entries are returned one at a time thus allowing the*/
/* linecount for the write socket to drop to zero at which point */
/* the connection is closed.                                     */
/* */
IF wsock¬='' THEN DO
   IF linecount.wsock=Ø THEN DO
      PARSE VAR stringword.wsock '/?Command=' command .
      UPPER command
      PARSE VAR command command '+' data
      CALL data_fix /* deal with % characters and pluses */
      SELECT
        WHEN command='DISKSPACE' THEN CALL diskspace_process
        WHEN command='TSO' THEN CALL tsocmds_process
        WHEN command='TAPEINFO' THEN CALL tapeinfo_process
        WHEN command='SHUTDOWN' THEN SIGNAL shutdown
        OTHERWISE CALL front_screen
      END
      IF RESULT='' THEN linecount.wsock=1
      ELSE linecount.wsock=RESULT
   END
   msgnum=linecount.wsock
   msg=msg.wsock.msgnum
   x='SOCKET'('SetSockOpt',wsock,'Sol_Socket','So_ASCII','On')
   x='SOCKET'('Send',wsock,msg)
   IF WORD(x,1)='Ø' THEN DO
      linecount.wsock = linecount.wsock - 1
      DROP msg.wsock.msgnum
      END
   IF WORD(x,1)¬='Ø' THEN DO /* send failure - cleanup */
      linecount.wsock=Ø      /* indicate no lines      */
      DO x=1 TO msgnum
         DROP msg.wsock.x    /* release storage */
      END
      DROP stringword.wsock
      DROP stringuser.wsock
   END
   IF linecount.wsock=Ø THEN DO
      x='SOCKET'('Close',wsock)
      END
   END
END
/* */
/* Terminate the server and exit */
/* */
shutdown:
x='SOCKET'('Terminate')
SAY 'RSSERVER: Terminated'
EXIT Ø
/* */
/* ================== The processing subroutines ===================*/
/* */
diskspace_process:
/* */
CALL RVOLDATA
/* */
/* first pass back the title line. */
/* */
titleline='Address Volser Free_Extents Free_Cyls Free_Trks Large_Cyl',
    'Large_Trk Index Frag'
y=2
msg.wsock.y='<HTML><HEAD><TITLE>DISKSPACE</TITLE><HEAD>'||,
 '<BODY BGCOLOR="#ØØffff" LINK="#ØØØØff" VLINK="#8ØØØ8Ø">'||,
 '<FONT COLOR="#ØØ8ØØØ">'||,
 '<H1 ALIGN="CENTER">Result of' lparname 'Diskspace Command</H1>'||,
 '<TABLE BORDER CELLSPACING=1><TR>'
DO x=1 TO WORDS(titleline)
  msg.wsock.y=msg.wsock.y||'<TD VALIGN="MIDDLE"><P><FONT FACE="Arial">',
   ||WORD(titleline,x)||'</FONT></TD>'
END
msg.wsock.y=msg.wsock.y||'</TR>';y=y-1
msg.wsock.y=''
DO x=1 TO volser.Ø*1
   msg.wsock.y=msg.wsock.y||,
               '<TR><TD VALIGN="MIDDLE"><P>'||address.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||volser.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||1*free_extents.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||1*free_cylinders.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||,
               (1*free_tracks.x)+(15*free_cylinders.x)||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||,
               1*largest_cylinder_extent.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||,
               1*largest_track_extent.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||index_status.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||,
               1*fragmentation_index.x||'</TD></TR>'
```

```
END
msg.wsock.y=msg.wsock.y||'</TABLE></HTML>'
/* */
RETURN 2
/* */
tapeinfo_process:
/* */
CALL UCBTAPE
/* */
/* first pass back the title line. */
/* */
titleline='Address Device Volume Jobname'
y=2
/* */
/* Insert <META HTTP-EQUIV="REFRESH" CONTENT=30> after the HEAD */
/* and before title if you want to get this screen to refresh   */
/* itself every 30 seconds.                                     */
/* */
msg.wsock.y='<HTML><HEAD><TITLE>TAPE information</TITLE><HEAD>'||,
 '<BODY BGCOLOR="#00ffff" LINK="#0000ff" VLINK="#800080">'||,
 '<FONT COLOR="#008000">'||,
 '<H1 ALIGN="CENTER">Result of' lparname 'Tapeinfo Command</H1>'||,
 '<TABLE BORDER CELLSPACING=1><TR>'
DO x=1 TO WORDS(titleline)
  msg.wsock.y=msg.wsock.y||'<TD VALIGN="MIDDLE"><P><FONT FACE="Arial">',
   ||WORD(titleline,x)||'</FONT></TD>'
END
msg.wsock.y=msg.wsock.y||'</TR>';y=y-1
msg.wsock.y=''
DO x=1 TO volume.0*1
   msg.wsock.y=msg.wsock.y||,
               '<TR><TD VALIGN="MIDDLE"><P>'||address.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||STRIP(unit_type.x)||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||volume.x||'</TD>',
               '<TD VALIGN="MIDDLE"><P>'||job_name.x||'</TD></TR>'
END
msg.wsock.y=msg.wsock.y||'</TABLE></HTML>'
/* */
RETURN 2
/* */
tsocmds_process:
ADDRESS TSO
CALL OUTTRAP('LINE.')
''data
/* */
/* now build the html */
/* */
msg.wsock.1='<HTML><HEAD><TITLE>TSO Command</TITLE></HEAD>'||,
 '<BODY BGCOLOR="#00ffff" LINK="#0000ff" VLINK="#800080">'||,
 '<FONT COLOR="#008000">'||,
'<B><FONT FACE="Arial" SIZE=4 COLOR="#ff0000"><P ALIGN="CENTER">'||,
lparname 'TSO Command:' data '</P></B><HR></FONT><FONT SIZE=2>'
```

```
DO x=1 TO line.0
msg.wsock.1=msg.wsock.1||line.x '<BR>'
DROP line.x
END
msg.wsock.1=msg.wsock.1||'</FONT></BODY></HTML>'
CALL OUTTRAP('OFF')
RETURN 1
/* */
/* The front screen routine sends back a basic data entry screen */
/* to enable the user to make requests of this server            */
/* */
front_screen:
msg.wsock.1='<HTML><HEAD><TITLE>TITLE LINE</TITLE></HEAD>'||,
 '<BODY BGCOLOR="#00ffff" LINK="#0000ff" VLINK="#800080">'||,
 '<FONT COLOR="#008000">'||,
'<B><FONT FACE="Arial" SIZE=4 COLOR="#ff0000"><P ALIGN="CENTER">'||,
'ENTER COMMAND FOR' lparname 'SERVER </P></B></FONT><FONT SIZE=2>'||,
'<form name="topicForm">'||,
 'Please specify required command <INPUT NAME="Command" SIZE=40>'||,
 '</select></form></font>'||,
 '<H2>Available Commands</H2><HR>'||,
 '<FONT FACE="Courier New"><OL>'||,
 '<LI>DISKSPACE ........ Requests disk space information'||,
 '<LI>TAPEINFO ......... Requests Tape Drive Usage'||,
 '<LI>TSO .............. Issues a TSO command'||,
 '<LI>SHUTDOWN ......... Closes down the server'||,
 '</OL></BODY></HTML>'
RETURN 1
/* */
/* data translation routine to clear up the CGI stuff */
/* */
data_fix:
IF INDEX(data,'+')/=0 THEN data=TRANSLATE(data,' ','+')
/* Now do left brackets */
IF INDEX(data,'%28')/=0 THEN DO UNTIL INDEX(data,'%28')=0
   PARSE VAR data data '%28' rem
   data=data'('rem
   END
IF INDEX(data,'%29')/=0 THEN DO UNTIL INDEX(data,'%29')=0
   PARSE VAR data data '%29' rem
   data=data')'rem
   END
IF INDEX(data,'%27')/=0 THEN DO UNTIL INDEX(data,'%27')=0
   PARSE VAR data data '%27' rem
   data=data"'"rem
   END
IF INDEX(data,'%23')/=0 THEN DO UNTIL INDEX(data,'%23')=0
   PARSE VAR data data '%23' rem
   data=data"#"rem
   END
IF INDEX(data,'%3B')/=0 THEN DO UNTIL INDEX(data,'%3B')=0
   PARSE VAR data data '%3B' rem
   data=data';'rem
   END
IF INDEX(data,'%22')/=Ø THEN DO UNTIL INDEX(data,'%22')=Ø
   PARSE VAR data data '%22' rem
   data=data'"'rem
   END
IF INDEX(data,'%A3')/=Ø THEN DO UNTIL INDEX(data,'%A3')=Ø
   PARSE VAR data data '%A3' rem
   data=data'$'rem
   END
IF INDEX(data,'%25')/=Ø THEN DO UNTIL INDEX(data,'%25')=Ø
   PARSE VAR data data '%25' rem
   data=data'%'rem
   END
IF INDEX(data,'%2B')/=Ø THEN DO UNTIL INDEX(data,'%2B')=Ø
   PARSE VAR data data '%2B' rem
   data=data'+'rem
   END
IF INDEX(data,'%2C')/=Ø THEN DO UNTIL INDEX(data,'%2C')=Ø
   PARSE VAR data data '%2B' rem
   data=data','rem
   END
IF INDEX(data,'%21')/=Ø THEN DO UNTIL INDEX(data,'%21')=Ø
   PARSE VAR data data '%21' rem
   data=data'!'rem
   END
IF INDEX(data,'%3F')/=Ø THEN DO UNTIL INDEX(data,'%3F')=Ø
   PARSE VAR data data '%3F' rem
   data=data'?'rem
   END
IF INDEX(data,'%3D')/=Ø THEN DO UNTIL INDEX(data,'%3D')=Ø
   PARSE VAR data data '%3D' rem
   data=data'='rem
   END
RETURN
```

SAMPLE START-UP JOB

```
//your job card
//A EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=6M
//STEPLIB DD DSN=library.with.ucbtape.and.rvoldata,DISP=SHR
//SYSPROC  DD DSN=library.with.htmlrexx.in,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSTSOUT DD  SYSOUT=*
//SYSOUT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
PROFILE NOPREFIX
HTMLREXX
```

UCBTAPE

Note that this routine can be linked into any load library, as it does not require APF authorization. Furthermore this routine is not limited to use in just this server. It can be exploited by any REXX routine wishing to access tape details. See the list of variables that are created in the comments for ideas.

```
//your job card
//STEPA EXEC ASMFCL,PARM.LKED='NORENT,NOREUS'
//ASM.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//           DD DSN=SYS1.MODGEN,DISP=SHR
//ASM.SYSIN DD *
***********************************************************************
* THIS ROUTINE ANALYSES THE UCBS OF THE DEVICES ON THE SYSTEM AND
* CREATES A VARIETY OF REXX ARRAY VARIABLES FOR ON-LINE TAPE UNITS
* THE VARIABLES CREATED ARE AS FOLLOWS:
*
* CHPS .............THE NUMBER OF CHANNEL PATHS ATTACHED TO A DEVICE
* RCODE ............RETURN CODE FORM THE UCBINFO MACRO
* UNIT_TYPE ........CONFIRMATION OF DEVICE TYPE USING THE EDTINFO
*                   MACRO.
* VOLUME ...........VOLUME SERIAL NUMBER.
* ADDRESS ..........DEVICE ADDRESS
* JOB_NAME .........JOB USING THIS TAPE (IF A TAPE ON DRIVE).
* PATH_ID.1 ........PATH NUMBER ATTACHED TO THE DEVICE. THIS IS AN
*                   ARRAY DEPTH 8 (MAX NUM OF ATTACHABLE CHANNELS.
* PATH_TYPE.1 ......CHARACTER DESCRIPTION OF THE TYPE OF PATH TO WHICH
*                   THE DEVICE IS ATTACHED (ESCON, COPPER ETC.)
*
* NOTE: THE BASE VARIABLE FOR ALL THESE ITEMS IS VOLUME.Ø
*       IF INDIVIDUAL, OR OTHER BASES ARE REQUIRED, SIMPLY ADD IN
*       THE APPROPRIATE SHOWBASE MACROS.
***********************************************************************
UCBTAPE TITLE 'REXX FUNCTION TO RETRIEVE UCB INFORMATION'
        MACRO
        REXREGS
        LCLA &CNT
&CNT    SETA Ø
.LOOP   ANOP
R&CNT   EQU &CNT
&CNT    SETA &CNT+1
        AIF (&CNT LT 16).LOOP
        MEND
        MACRO
        SHOWSET
        AIF (D'SHOW_START).NONEED
        B  BY_SHOW_START
SHOW_START DS ØH
        ST R1Ø,COMRET
        LA 6,COMSHVB
        USING SHVBLOCK,R6
        XC COMSHVB(SHVBLEN),COMSHVB
        XC SHVNEXT,SHVNEXT
        MVI SHVCODE,C'S'
        BR 14
ABENDØØ1 DS ØH
        ABEND 1   * REQUIRED FOR THE OTHER MACROS. SAVES SOME CODING.
BY_SHOW_START DS ØH
LITLOC  LOCTR
@_UNPACK DC CL16' '
         DC CL8' '  * FILL FIELD
         ORG @_UNPACK+8
@_UNPACKER DC CL8' '
         ORG
@_DWORD  DS CL8     * USED FOR THE DEBIN FUNCTION
&SYSECT LOCTR
.NONEED ANOP
        BAL 14,SHOW_START
        MEND
        MACRO
  SHOWARAY &LABEL,&ASNAME,&ERR=ABENDØØ1,&LEN=,&SUBARRAY=,&DEBIN=,&LINK=
        PRINT NOGEN
***********************************************************************
* MACRO TO CREATE REXX ARRAY VARIABLES
*
* NOTE RESTRICTION: THIS MACRO IS LIMITED TO CREATING UP TO 9,999,999
*                   ENTRIES FOR EACH ARRAY.
* MACRO FORMAT:
*       SHOWARAY &LABEL,&ASNAME,&ERR=,&LEN=,&SUBARRAY=,&DEBIN=
* WHERE:
*       &LABEL IS THE NAME OF THE LABEL WHICH ADDRESS THE FIELD FROM
*              WHERE THE DATA TO BE DEFINED IN A REXX VARIABLE IS
*              LOCATED
*       &ASNAME IS THE NAME TO BE ASSIGNED TO THE DATA FOR USE IN REXX
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*       &LEN= IF THE DATA AT &LABEL IS NOT DEFINED SUCH THAT THE LENGTH
*             OF THE DATA IS WHAT YOU WANT, SIMPLY ENTER A NUMBER HERE
*             THAT DEFINES THE LENGTH REQUIRED. CAN ALSO BE USEFUL IF
*             NECESSARY TO DUMP OUT A LARGE AREA.
*       &SUBARRAY= IF A MULTI LEVEL ARRAY IS REQUIRED EG A.1.1 THEN
*                  SET THIS VALUE ACCORDINGLY.
*       &DEBIN= IF THE DATA TO BE CREATED IS BINARY, SETTING THIS TO A
*               VALUE WILL CONVERT THE SPECIFIED NUMBER OF BYTES FROM
*               BINARY TO CHARACTER. THE DEFAULT LENGTH FOR THE
*               OUTPUT DATA IS 4 BYTES. IF THIS IS INSUFFICIENT, THEN
*               SPECIFY A SUITABLE &LEN VALUE TO OVERRIDE IT.
*       &LINK= THIS IS A REXX NAME LABLE TO WHICH THE ARRAY COUNT IS
*              LINKED. THE PURPOSE OF THIS IS TO ALLOW A BRANCH OUT
*              OF ARRAY LOOPS WHILE STILL MAINTAINING NUMERIC
*              CONSISTENCY.
**********************************************************************
```

```
        PRINT GEN
        LCLA &DEFLEN
&DEFLEN SETA 16
        SHOWSET
LITLOC  LOCTR
&LABCHECK SETC '@_&ASNAME&SUBARRAY'
&LINKNAME SETC '@_&LINK'
         AIF   (D'&LABCHECK).BYPASS
         AIF   (T'&SUBARRAY EQ 'O').NORMNAME
&LABCHECK DC C'&ASNAME..&SUBARRAY'
         AGO   .EOFARRAY
.NORMNAME ANOP
&LABCHECK DC C'&ASNAME'
.EOFARRAY ANOP
&LABCHECK._ARRAY DC C'.       '
&LABCHECK._COUNTER DC PL4'Ø' * COUNTER FIELD FOR THIS ITEM
.BYPASS  ANOP
&SYSECT LOCTR
        AIF (T'&LINK EQ 'O').DOADD
        MVC &LABCHECK._COUNTER,&LINKNAME._COUNTER
        AGO .DOUNPK
.DOADD  ANOP
        AP &LABCHECK._COUNTER,=P'1' * INCREMENT THE COUNTER THIS PASS
.DOUNPK ANOP
        UNPK @_UNPACKER,&LABCHECK._COUNTER * UNPACK THE VALUE
        OI   @_UNPACKER+7,X'FØ'       * REMOVE THE SIGN
* NOW NEED TO WORK OUT THE LENGTH OF THE COUNTER BIT TO ADD TO ARRAY
        L    R15,&LABCHECK._COUNTER * LOAD THE COUNTER VALUE TO WORK
*                             OUT THE LENGTH
        SRL  R15,4                  * REMOVE THE SIGN
        XR   R14,R14                * CLEAR R14 FOR A COUNTER
LOOP&SYSNDX DS ØH
        SRA  R15,4                  * MOVE DIGIT BY DIGIT
        LTR  R15,R15
        BZ   COUNT&SYSNDX
        LA   R14,1(,R14)
        B    LOOP&SYSNDX
COUNT&SYSNDX DS ØH
* NOW ADD COUNT FIELD TO NAME
        LA  R15,@_UNPACKER+7         * POINT TO END OF FIELD
        SR  R15,R14                 * AND COME BACK TO FIRST DIGIT.
        MVC &LABCHECK._ARRAY+1(7),Ø(R15)
        LA 1,&LABCHECK
        ST 1,SHVNAMA
* NOW CALCULATE NEW LENGTH
        LA 1,L'&LABCHECK
        LA 1,2(R14,R1)
        ST 1,SHVNAML
        AIF (T'&DEBIN EQ 'O').NORMLAB
*** NOW ALLOW FOR A BINARY CONVERSION
*** FIST CALCULATE THE ICM VALUE
*
```

```
&ICM    SETA (1 SLL &DEBIN)-1
        XR R15,R15
        ICM R15,&ICM,&LABEL * LOAD THE BINARY VALUE
        CVD R15,@_DWORD               * CONVERT TO PACKED
        OI  @_DWORD+7,X'ØF'
        UNPK @_UNPACK,@_DWORD
*
*** IF THE LEN VALUE IS SUPPLIED THIS OVERRIDES THE DEFAULT OF 16
*
        AIF (T'&LEN EQ 'O').SETDEF  * LENGTH NOT SUPPLIED USE DEFLEN
&DEFLEN SETA &LEN                   * RESET DEFLEN TO SUPPLIED LEN
.SETDEF ANOP
        LA R1,@_UNPACK+(16-&DEFLEN)
        ST R1,SHVVALA
        LA R1,&DEFLEN
        AGO .OK
.NORMLAB ANOP
        LA 1,&LABEL
        ST 1,SHVVALA
        AIF (T'&LEN NE 'O').DOLEN
        LA 1,L'&LABEL
        AGO .OK
.DOLEN  ANOP
        LA 1,&LEN
.OK     ANOP
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
        MACRO
        SHOWBASE &LABEL,&ERR=ABENDØØ1,&SUBARRAY=
***********************************************************************
* MACRO TO CREATE REXX BASE VARIABLES
* SHOULD BE USED IN ASSOCIATION WITH A SHOWARAY MACRO. NOTE THAT A
* SHOWBASE MACRO IS OPTIONAL IF YOU ALREADY KNOW THE NUMBER OF
* VARAIABLES BEING SET. THIS WILL CREATE THE A.Ø ENTRY
*
* MACRO FORMAT:
*       SHOWBASE &LABEL,&ERR=,&SUBARRAY=
* WHERE:
*       &LABEL IS THE NAME OF THE REXX ARRAY LABEL WHICH HAS BEEN
*              CREATED. THIS WILL CREATE THAT LABEL.Ø ENTRY
*       &ERR= IS THE LABEL TO BRANCH TO SHOULD AN ERROR OCCUR WHILE
*             CREATING THE REXX VARIABLE. BY DEFAULT IT IS ABENDØØ1
*       &SUBARRAY= IF SUBARRAYS HAVE BEEN USED THIS WILL INSERT THE
*                  APPROPRIATE VALUE EG A.1.Ø
**********************************************************************
        SHOWSET
        AIF (T'&SUBARRAY EQ 'O').NORMNAME
&ASNAME   SETC '&LABEL..&SUBARRAY..Ø'
        AGO .CHECKER
.NORMNAME ANOP
&ASNAME   SETC '&LABEL..Ø'
.CHECKER  ANOP
&LABCHECK SETC '@_&LABEL&SUBARRAY._COUNTER'
        AIF (D'&LABCHECK).ITSOK
MNOTE   NO ARRAY ELEMENTS DEFINED.
        MEXIT
.ITSOK  ANOP
LITLOC  LOCTR
@_A&SYSNDX DC C'&ASNAME'
&SYSECT LOCTR
        LA 1,@_A&SYSNDX
        ST 1,SHVNAMA
        LA 1,L'@_A&SYSNDX
        ST 1,SHVNAML
        UNPK @_UNPACKER,&LABCHECK
        OI  @_UNPACKER+L'@_UNPACKER-1,C'Ø'
        LA 1,@_UNPACKER
        ST 1,SHVVALA
        LA 1,L'@_UNPACKER
        ST 1,SHVVALL
        LR Ø,1Ø
        LA 1,COMS
        L 15,IRXEXCOM
        BALR 14,15
        LTR 15,15
        BNZ &ERR
        MEND
UCBTAPE AMODE 31
UCBTAPE RMODE ANY
UCBTAPE CSECT
         REXREGS
         BAKR  R14,RØ
         LR    R12,R15
         LA    R11,2Ø48(,R12)            * ESTABLISH ADDRESSABILITY FOR
         LA    R11,2Ø48(,R11)            * UP TO 8K
         USING UCBTAPE,R12,R11
         LR    R1Ø,RØ                  * R1Ø -> A(ENVIRONMENT BLOCK)
         USING ENVBLOCK,R1Ø
         L     R9,ENVBLOCK_IRXEXTE     * R9 -> A(EXTERNAL EP TABLE)
         USING IRXEXTE,R9
         STORAGE OBTAIN,LENGTH=GETLEN,ADDR=(8)
         USING COMSDS,R8
* PREPARE THE REXX AREA FOR USE
         XC    COMS(COMSLEN),COMS      * SET TO LOW VALUES
         LA    R15,COMID
         ST    R15,COMS
         LA    R15,COMDUMMY
         ST    R15,COMS+4
         ST    R15,COMS+8
         LA    R15,COMSHVB
         ST    R15,COMS+12
         LA    R15,COMRET
         ST    R15,COMS+16
         OI    COMS+16,X'80'            * INDICATE END OF PARMS
         MVC   COMID,=C'IRXEXCOM'
* COMMENCE THE LOOP OF UCBS LOOKING FOR A RELEVANT DISKS.
* HAVING FIRST SET ADDRESSABILITY TO THE UCB AREA TO R3
          LA R3,UCBWORK
          USING UCBOB,R3
          LA R4,UCBEXTN
          USING UCBCMEXT,R4
          LA R5,PATHINFO
          USING PATH,R5
          XC HUNDRED,HUNDRED
UCBLOOP DS 0H
          UCBSCAN COPY,WORKAREA=HUNDRED,RANGE=ALL,UCBAREA=UCBWORK,     +
               DYNAMIC=YES,DEVCLASS=TAPE,DEVNCHAR=MYDEV,DEVN=0,        +
               PLISTVER=MAX,CMXTAREA=UCBEXTN
* IF R15 CONTAINS 04, THEN LAST UCB RETRIEVED SO SET END FLAG
* FOR DEFENSIVE CODE, ASSUME ALL NON-ZERO RETURN CODES ARE THE
* EQUIVALENT OF END OF UCBS
          LTR R15,R15          * END OF UCBS?
          BNZ RETURNV          * YES SO GO SET THE BASE ARRAY VALUES
*** IF THE DEVICE IS OFFLINE, THEN WE ARE NOT INTERESTED
          TM UCBSTAT,X'80'     * IS THS VOLUME ONLINE
          BNO UCBLOOP          * NO, SO GO BACK AROUND
*** THE FOLLOWING SECTION RELATES TO CREATING THE VARIABLES ASSOCIATED
*** WITH THE UCB
          MVI ATTRIBS,X'0A' * NEEDS TO BE DONE PRIOR TO EDTINFO CALL
          EDTINFO RTNUNIT,DEVTYPE=UCBTYP,OUTUNIT=UNIT,EXTENDED=YES
          SHOWARAY UNIT,UNIT_TYPE
          SHOWARAY UCBVOLI,VOLUME
          CLC      UCBVOLI,=XL6'00'      * IS THE VOLSER NULL
          BE       NULL_JOB              * YES SO SKIP THIS BIT
          LOCASCB  ASID=UCBASID          * TRY TO GET JOBNAME
          LTR 15,15
          BNZ      NULL_JOB
          LR  2,1   * MOVE ADDRESS TO USABLE REGISTER
          IAZXJSAB READ,ASCB=(2),JOBNAME=THISJOB
          LTR 15,15      * NO DETAILS FOUND?
          BNZ NULL_JOB
          SHOWARAY THISJOB,JOB_NAME
          B   SET_DEVICE
NULL_JOB  DS 0H
          SHOWARAY BLANK,JOB_NAME
SET_DEVICE DS 0H
          SHOWARAY MYDEV,ADDRESS
*** NOW GET THE PATH DETAILS FOR DISPLAY
*** NOTE THAT A NON-ZERO RETURN CODE FORM THIS MACRO INDICATES A
*** NON-ATTACHED DEVICE. PRIOR TO CALLING THIS MACRO IT IS NECESSARY
*** TO ZEROIzE THE WORKAREA, OTHERWISE IT MAY END UP CONTAINING OLD
*** INFORMATION FROM A PREVIOUS CALL.
          XC PATHINFO,PATHINFO * ZEROISE ALL THE AREA
  UCBINFO PATHINFO,PATHAREA=PATHINFO,DEVN=UCBCHAN,RETCODE=RC
*
      SHOWARAY RC,RCODE,DEBIN=4
          L 15,RC
          LTR 15,15
          BNZ UCBLOOP
      SHOWARAY PATH#CHPIDS,CHPS,LEN=2,DEBIN=4,LINK=RCODE
*                                       * SHOW NUMBER OF ATTACHED PATHS
***
*** SECTION PATHLOOP CREATES THE VARIABLES FOR EACH OF THE ATTACHED
*** PATHS. NOTE THAT UP TO 8 CAN BE ATTACHED.
***
PATHLOOP  DS ØH
          LA R7,PATHCHPIDARRAY
          USING PATH_ARRAY_MAP,R7
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=1,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=1,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=2,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=2,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=3,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=3,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=4,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=4,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=5,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=5,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=6,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=6,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=7,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=7,LINK=RCODE
          LA R7,28(,R7)  * SHIFT PATH MAP ALONG TO NEXT 28 BYTE SET
      SHOWARAY IDPATH,PATH_ID,SUBARRAY=8,DEBIN=2,LINK=RCODE
          BAL R2,PATHSET
      SHOWARAY DESCRIPTION,PATH_TYPE,SUBARRAY=8,LINK=RCODE
          B UCBLOOP       * NOW GO AROUND AGAIN
RETURNV DS ØH
          SHOWBASE VOLUME
```

```
ENDREXX  DS ØH
          STORAGE RELEASE,LENGTH=GETLEN,ADDR=(8)
          PR
PATHSET  DS ØH
         IOSCHPD CHP_TYPE=TYPEPATH,DESC=DESCRIPTION
         BR  R2                       * AND RETUNR FOR DISPLAY
          LTORG
BLANK    DC C' '
*
*
PATH_ARRAY_MAP DSECT
IDPATH       DS CL2
             DS C
MASKPATH     DS C
TYPEPATH     DS C
TDESCT DSECT
          IEFUCBOB DEVCLASS=DA
***********************************************************************
*
***********************************************************************
***      IRXEXCOM PARAMETER AREA                                    ***
***********************************************************************
*
COMSDS    DSECT
COMS      DS    5AL4
COMID     DS    CL8               * IRXEXCOM ID - C'IRXEXCOM'
COMDUMMY  DS    AL4               * NOT USED
COMSHVB   DS    (SHVBLEN)X       * IRXEXCOM SHVBLOCK (LENGTH FROM DSECT)
COMRET    DS    AL4               * IRXECOM RC
COMSLEN   EQU *-COMS
HUNDRED   DS CL1ØØ
UCBWORK   DS CL25Ø
UCBEXTN   DS CL5Ø
MYDEV     DS CL4
PATHINFO  DS CL256
UNIT      DS CL8
ATTRIBS   DS CL1Ø
RC        DS F
THISJOB   DS CL8
STATUS    DS    CL7
DESCRIPTION DS CL32
GETLEN   EQU *-COMS
          IOSDPATH
          IAZJSAB
          IHAASCB
          IHAASSB
          DS ØD
          IRXEFPL
          IRXARGTB
          IRXEVALB
          IRXENVB
          IRXEXTE
          IRXSHVB
          END
/*
//LKED.SYSLMOD DD DSN=T226.D.SVC,DISP=SHR,UNIT=
//LKED.SYSIN DD *
         ENTRY UCBTAPE
         NAME  UCBTAPE(R)
```

ADDITIONAL FUNCTIONALITY

While developing this system I was curious about the possibility of using the Web to draw together the information from the servers on each machine so that the data could be viewed on one screen. The following code uses frames to generate the above screen. In order to exploit this, you will need to modify the HTTP code to point to your IP:PORT address for each of the machines/LPARs where the respective servers reside. Note that if you simply specify the IP:PORT without the tapeinfo command, you will be presented with multiple data entry screens, each of which can be used independently to allow all the servers to be contacted for differing information.

```
<HTML><HEAD>
<TITLE>This Document Demonstrates Frames</TITLE></HEAD>
<FRAMESET COLS="3Ø%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes"
NAME="frame1"
[ccc] FRAMEBORDER="yes">
<FRAMESET COLS="3Ø%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes"
NAME="frame2"
[ccc] FRAMEBORDER="yes"><FRAMESET ROWS="3Ø%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes"
NAME="frame3"
[ccc] FRAMEBORDER="yes">
<FRAMESET ROWS="3Ø%,*">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes"
[ccc] NAME="frame4">
FRAMEBORDER="yes">
<FRAME SRC="http://yourip:port/?Command=tapeinfo" SCROLLING="yes"
[ccc] NAME="frame5">
FRAMEBORDER="yes">
</FRAMESET>
</FRAMESET>
```

Systems Programmer (UK) © Xephon 2001

MVS news

Sites running TCP/IP with MVS will be interested in Landmark Systems' TMON for TCP/IP. The product is used to monitor and manage the availability of network devices and overall network performance. It provides a centralized monitor for both OS/390 and non-OS/390 stacks. TMON for TCP/IP complements Landmarks' TMON for IMS and TMON for Unix Systems Services product family.

The product promises to provide a detailed insight into resources impacting TCP/IP network performance, extending beyond the OS/390 environment to monitor and manage non-OS/390 TCP/IP stacks and Cisco routers, while also providing visibility into OS/390 subsystems, applications, and resources that impact the OS/390 TCP/IP stack.

Specifically, TMON for TCP/IP pinpoints both the problems and root causes throughout the entire network using conditional monitoring and alarming across SNMP and non-SNMP entities. It identifies the performance problems, bottlenecks, and availability issues that can impact business operations.

For further information contact:

Landmark Systems Corporation, 12700 Sunrise Valley Drive, Reston, Virginia 20191-5804, USA.
Tel: (703) 464 1300
Fax: (703) 464 4918
Landmark Systems (UK),The Boardroom, Morton Gardens, Lower Hillmorton Road, Rugby, Warwickshire, CV21 3SX, UK.
Tel: (08700) 744 755
Fax: (08700) 755 766
http://www.landmark.com

* * *

Software AG has announced support for SuSE Linux AG's Linux Enterprise Server for System/390 by its Tamino XML database, which means XML can be used for existing mainframe applications and, therefore, extended to the Web more effectively. The System/390 can host up to 30,000 virtual Linux servers concurrently. Apparently, Software AG ported Tamino in just a few weeks, supported by SuSE and IBM. The Tamino XML Database is the core of Software AG's Tamino XML Platform, and is an XML-based architecture for developing, processing, and storing documents in XML format.

For further information contact:

Software AG, Uhlandstr. 12, 64297 Darmstadt, Germany.

Phone: +49 6151 92 0
Fax: +49 6151 92 1191
http://www.softwareag.com

* * *

IBM has extended its WebSphere Application Server V3.5 Standard Edition to OS/390 with a run time version. It includes support for host Web components such as servlets and Java Server Pages developed with the WebSphere Family V3.5 programming model and it supports servlets and Java Server Pages, developed and tested using WebSphere V3.5 run times and V3.5 tooling on distributed platforms, that can be redeployed unchanged into the new software's run time.

Contact your local IBM representative for further information.

http://www.ibm.com/software

* * *

xephon