162

MVS update

March 2000

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,
Westminister, CO 80035-0100
USA
Telephone: 303 410 9344

Contributions

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

Editor

Jaime Kaminski

Disclaimer

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

MVS Update on-line

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

Subscriptions and back-issues

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

Printed in England.

Verifying internal/external VOLSERs in a tape silo

THE PROBLEM

It is common for large sites to work with massive numbers of tape cartridges. Volumes flow between sites and new tapes are constantly being added to the pools. The normal procedure for operators would be to initialize tapes in groups, define them into the tape control system and then enter them into one of the existing pools of tapes. This often occurs in a silo environment and tapes are imported in bulk. At times it may also be necessary to re-label because of damage done to the bar-coded external labels.

There is one problem that might require a manual intervention at some later stage – this will occur if for some reason the internal and external tape volume labels (VOLSERs) are different. This can be a major problem because it will also impact the production run – a job asks for one tape and the wrong one is then mounted because of a mismatch between the external label and the internal label or VOLSER. One way to overcome this is to run a job that dittos the first file on each tape to make sure that there is no such discrepancy. This is a tedious task and is not something most operators would look forward to.

A SOLUTION

The following program was developed to overcome this problem. It is ideal in a tape silo environment because it can scan large ranges of tapes and cross-check the internal VOLSERs with the external ones. The program can obviously not ‘see’ the external label – that is taken care of by the silo (or operator in the case of a manual mount) by mounting the tape with the external bar-coded label matching that of the VOLSER in the MOUNT message.

This program is given a range of tapes (eg ABC001-ABC900); it will then start working through the 900 tapes (in this case) by asking for them to be mounted one at a time. Once the tape has been mounted, the label is treated as a file and read. The VOLSER appears in the first label, a check can thus be done to make sure that the tape has the correct VOLSER. The JCL used to run the tape has to use the NL (No-Label) option. If SL (Standard Label) is used, MVS will reject the tape as soon as the file is opened and another will be issued. Even if the correct tape were mounted, it would clash with the logic of this program because it would have been written specifically to read the label as the first file on the tape. Note that it may be necessary to get special authority to run the job in an environment where a tape control system is in place – not all jobs would normally be permitted to use the NL option. You can put your tape administrator at ease: this job will read only the tape. If the internal/ external VOLSERs match, the next tape is asked for. If they do not match, a WTO is issued (which goes to the job log only) and the next tape is asked for. At the end of the run, a list is thus available that will point out any volumes for which there are discrepancies. These can then be relabelled or re-initialized and verified. The program can also do verification for single tapes, it does not have to be ranges.

Note that, in the case where we have a large number of volumes to verify, it may be a good idea to check smaller ranges at a time by running multiple jobs with different ranges. For the keen ones, it would be easy to modify this program to subtask so that each subtask can verify a subset of the specified range. Just keep in mind that each such job would keep one tape drive busy for 100% of the duration of the job – do not run too many because you may impact production. The program was written to deal with a tape range of up to 20,000 tapes (should be plenty!), but if that is not enough, it can easily be upgraded by upping the value of field NUMBYTES and re-assembling it.

SOURCE

```
*********************************************************************
*
*  This program checks the internal labels against external labels for
*  tapes, the VOLSERs of which are supplied from //SYSIN.
*  A list of VOLSERs can be supplied, one per SYSIN card. A card can
*  also contain a range of VOLSERs, eg TAPØØ1-TAPØØ2. In this case
*  the length must be 13 bytes with a "-" in col. 7. The following are
*  rules for ranges:
*  1) VOLSERs in a range must be six bytes long
*  2) VOLSERs in a range can only contain A-Z and Ø-9
*  3) If the "FROM" VOLSER has a digit in a certain byte, the "TO"
*     VOLSER must also have a digit in that byte, the same applies for
*     chars.
*     eg  TAPØØ1-TAP453 (valid)
*         TAPAØ1-TAP453 (invalid as "A" and "4" are different types)
*
*     Sample JCL
*
*     //CHECK  EXEC PGM=VOLSERCK
*     //TAPEVOL  DD UNIT=(CART,,DEFER),LABEL=(1,BLP),DISP=(OLD,PASS), ),
*     //         DSN=A.B,VOL=SER=ANYVOL
*     //SYSIN    DD *
*     99Ø443
*     VOL13Q-VOL13Z
*     //
*
*          MODE:     Program runs unauthorized, is non-reentrant
*         INPUT:     SYSIN DD-card to contain VOLSERS
*        OUTPUT:     WTO messages to indicate if internal/ external
*                    VOLSERS are the same
*         AMODE:     24
*         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 to contain VOLSERS/VOLSER ranges
*  Special regs:     None
*                    R12= Base register
*                    R13= Pointer to getmained area and saveares
*
***********************************************************************
VOLSERCK CSECT
VOLSERCK AMODE 24
VOLSERCK RMODE 24
         BAKR  R14,Ø               Save Caller's Status
         BALR  R12,Ø
         USING Load,R12
**********************************************************************
*        Main driver routine
**********************************************************************
Load     L     R4,Ø(R1)            Ptr to parm
Storage  LA    R3,GetMSize
         A     R3,BufferSz         Add the buffer size
                                   STORAGE OBTAIN,LENGTH=(3),LOC=ANY
         LA    r3,getmsize         PART OF STORAGE WE WISH TO CLEAR
         LR    R2,R1               Point to getmained area
         XR    R9,R9
         MVCL  R2,R8               Propagate binary zeros
         USING GetMArea,R1
         ST    R13,SAVEAREA+4      Backchain
         DROP  R1
         lr    r13,r1
         USING GetMArea,R13        Addressability to getmained area
         OPEN  SYSIN
         TM    SYSIN+48,X'1Ø'      Did the file open?
         BO    GetInCrd            Yes
         WTO   'VOLSERCK(E): -Could not open SYSIN DD-card',ROUTCDE=11
         ABEND ØØØ1
GetInCrd LA    R2,InCard           Where input card should go
         GET   SYSIN,(2)
         BAS   R14,ScanCard        Analyse input card
         LTR   R15,R15             Card OK?
         BNZ   GetInCrd            No, ignore
NextOne  BAS   R14,MountVol        Go mount the volume
         TM    LastVol,Yes         Last volume in the range?
         BO    LoopEnd             Yes, get next SYSIN card
         BAS   R14,DetmNVol        Go determine the next vol in range
         B     NextOne             Redo the loop for each vol in range
LoopEnd  B     GetInCrd            Go get the next card
EndSysin CLOSE SYSIN
FreeTabl L     R2,Table@           Address of range table
         LTR   R2,R2               Do we have a table?
         BZ    FreeWrkA            No
         L     R3,TabSize          Size of area to release
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
FreeWrkA L     R4,RetCode          Pick up return code
         LA    R3,GetMSize         Size of area to free
         A     R3,BufferSz         Add the buffer size
         LR    R2,R13              Address of area to free
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
         LR    R15,R4              Copy return code
ToCaller PR    ,                   To caller
         DS    ØD                  Align
**********************************************************************
*        This routine scans the input card
**********************************************************************
ScanCard BAKR  R14,Ø
         NI    LastVol,No         Turn "last volume" flag off
         CLC   InCard+13(59),=59C' '
         BE    ChkDash
         WTO   'VOLSERCK(E): -Col 13 - 72 of input card must be blank',X
               ROUTCDE=11
         ABEND ØØØ1ChkDash  CLI   InCard+6,C'-'      Continuation?
         BE    Range              No
         OI    LastVol,Yes        Mark as last volume on this card
         B     MoveVol            Pick up volume
Range    MVC   FromVol,InCard     Start of range
         MVC   ToVol,InCard+7     End of range
         CLC   FromVol,ToVol      Starting volume < ending volume?
         BNH   ChkType            Yes
         MVC   InVlRnge+66(13),InCard
InVlRnge WTO   'VOLSERCK(E): "TO" range not greater than "FROM" range fX
               or xxxxxxxxxxxxx',ROUTCDE=11
         ABEND ØØØ3
ChkType  TR    FromVol,CharTab    Set "type" byte for each character
         TR    ToVol,CharTab      Set "type" byte for each character
         CLC   FromVol,ToVol      Are they of the same type?
         BE    ChkChar            Yes, go check for valid characters
         MVC   CorrErr+81(13),InCard
CorrErr  WTO   'VOLSERCK(E): -Corresponding chars in range must both beX
                numeric or char, xxxxxxxxxxxxx not processed',         X
               ROUTCDE=11
         LA    R15,4              Set return code to 4
         ST    R15,RetCode
         B     ScanCarX           Get out
ChkChar  EQU   *                  Make sure all characters valid
         LA    R3,6
         LA    R1,FromVol         From volume (containing char type)
         LA    R2,ToVol           To volume (containing char type)
         LA    R3,6
TypeLoop EQU   *
         CLI   Ø(R1),X'ØØ'        Invalid character?
         BE    InvlChar           Yes
         CLI   Ø(R2),X'ØØ'        Invalid character?
         BE    InvlChar           Yes
         LA    R1,1(R1)           Point to next character in from vol
         LA    R2,1(R2)           Point to next character in to vol
         BCT   R3,TypeLoop        Do for each character
         B     RangeOK            Go give a message
InvlChar MVC   InvlWTO+71(13),InCard
InvlWTO  WTO   'VOLSERCK(E): -Only A-Z and Ø-9 allowed in range specifiX
               cation, xxxxxxxxxxxxx not processed',ROUTCDE=11
         LA    R15,4              Set return code to 4
         ST    R15,RetCode
         B     ScanCarX           Get out
RangeOK  MVC   RangeWTO+28(13),InCard
RangeWTO WTO   'VOLSERCK(I): -Range xxxxxxxxxxxxx expanded',           X
               ROUTCDE=11
         BAS   R14,BldTable
MoveVol  MVC   CurrVol,InCard     Move VOLSER into current volume
         XR    R15,R15            Card OK
ScanCarX PR
**********************************************************************
*        This routine updates the JFCB, and opens the dataset
**********************************************************************
MountVol BAKR  R14,Ø
         RDJFCB TAPEVOL
         LA    4,JFCB
         USING INFMJFCB,4
         MVC   JFCBVOLS(6),CurrVol Move VOLSER into JFCB
OPENTAPE NI    IOError,NO          Turn open error flag off
         OPEN  (TAPEVOL,INPUT),TYPE=J
         TM    IOError,Yes        Did we get an I/O error?
         BO    MountVoX           Yes, skip this volume
```

```
VolOK    BAS   R14,ChkVol         Go compare external/internal
*        TM    IOError,Yes        Did we get an I/O error?
*        BO    MountVoX           Yes, skip this volume
CloseIt  CLOSE (TAPEVOL,REWIND)
MountVoX PR
*********************************************************************
*        This routine expands the range into a table
*********************************************************************
BldTable BAKR  R14,Ø
         CLC   Table@,=F'Ø'       Do we already have a table?
         BE    GetTable           No
         L     R1,Table@          Yes, no need to do getmain
         B     SetStart           Jump over OBTAIN
Gettable L     R3,TabSize         Size of table to get
         STORAGE OBTAIN,LENGTH=(3),LOC=BELOW
         LTR   R15,R15            Did we get it?
         BZ    Bldrange           Yes
         WTO   'VOLSERCK(E): Not enough REGION for table',ROUTCDE=11
         ABEND ØØØ2
BldRange ST    R1,Table@          Keep this address
SetStart LR    R4,R1              Start of table
         LA    R4,8(R4)           First word used as counter
         XR    R5,R5              Volume counter
         MVC   NextVol,InCard
         MVC   ToVol,InCard+7     Where the range should stop
VolLoop  EQU   *                  Build the range of volumes
         LA    R2,6               Number of characters in VOLSER
         LA    R1,NextVol+5       Character that was bumped up
TrLoop   EQU   *
         TR    Ø(1,R1),NextTab    Bump up a character
         CLI   Ø(R1),C''          Did we get a '' (after "Z")?
         BE    DoZ
         CLI   Ø(R1),C'Ø'         Did we get a "Ø" (after 9)?
         BE    PrevChar           Have to move up the previous char
         B     BumpUpOK           Successfully bumped up
DoZ      MVI   Ø(R1),C'A'
         B     PrevChar
PrevChar BCTR  R1,Ø               Point 1 character back
         BCT   R2,TrLoop          Redo the loop
BumpUpOK MVC   Ø(6,R4),NextVol    Move the volume into the table
         LA    R4,6(R4)
         LA    R5,1(R5)           Bump up volume counter
         C     R5,=F'19999'       Maximum # entries allowed in table
         BL    ChkEnd
         CLC   NextVol,ToVol      Did we reach the last one?
         BE    RangeEnd           Yes, just enough space
         MVC   MaxWTO+54(13),InCard
         MVC   MaxWTO+84(6),NextVol
MaxWTO   WTO   'VOLSERCK(W): -Max range of 2Ø ØØØ allowed for xxxxxxxxxX
               xxxx, last volume is xxxxxx',ROUTCDE=11
         LA    R15,4
         ST    R15,RetCode        Plug the return code
         B     RangeEnd
ChkEnd   CLC   NextVol,ToVol      Have we reached the last one?
         BL    VolLoop            No, not yet
RangeEnd L     R1,Table@
         ST    R5,Ø(R1)           Number of entries in table
         MVC   4(4,R1),=F'Ø'      Reset offset into table
BldTablX PR
*********************************************************************
*        This routine compares the internal and external VOLSERS
*********************************************************************
ChkVol   BAKR  R14,Ø
READTAPE LA    9,TapeRec
         READ  CHK,SF,TAPEVOL,(9),'S'
         CHECK CHK,DSORG=ALL
         TM    IOError,Yes          Did we get an I/O error?
         BO    ChkVolX              Yes, get out
         CLC   CurrVol,4(R9)        Are the VOLSERS the same?
         BE    SameVol              Yes
NotSame  MVC   NSameWTO+37(6),4(R9)
         MVC   NSameWTO+54(6),CurrVol
NSameWTO WTO   'VOLSERCK(W): - ===> Internal=xxxxxx, External=xxxxxx <=X
               == NOT THE SAME',ROUTCDE=11
         LA    R15,4
         ST    R15,RetCode          Set return code to 4
         B     ChkVolX              Get out
SameVol  MVC   SameWTO+31(6),4(R9)
         MVC   SameWTO+48(6),CurrVol
SameWTO  WTO   'VOLSERCK(W): -Internal=xxxxxx, External=xxxxxx same',  X
               ROUTCDE=11
ChkVolX  PR
*********************************************************************
*        This routine determines the next volume in the range
*********************************************************************
DetMNvol BAKR  R14,Ø
         L     R1,Table@
         L     R2,Ø(R1)            Number of entries available
         BCTR  R2,Ø                Reduce by 1
         ST    R2,Ø(R1)            Store back
         LTR   R2,R2               All done?
         BNZ   MoveNVol            No
         OI    LastVol,Yes         Set last-in-range flag on
MoveNVol L     R2,4(R1)            Pick up the last offset
         AR    R2,R1               Add to start of table
         LA    R2,8(R2)            Length of 2 control fields at start
         MVC   CurrVol,Ø(R2)       Make the next vol the current one
         L     R2,4(R1)            Pick up the last offset
         LA    R2,6(R2)            Bump up offset pointer
         ST    R2,4(R1)            Store it back
DetMNvoX PR
```

```
*********************************************************************
*        Tape error routine
*********************************************************************
         DS    ØD
TapeErr  BAKR  R14,Ø
         LR    R12,R15             Pick up our current address
         DROP  R12
         USING TapeErr,R12
         OI    IOError,Yes         Set error flag on
         MVC   ErrWTO+29(6),CurrVol
ErrWTO   WTO   'VOLSERCK(W): -Volume xxxxxx damaged, uninitialised or Nx
               CA',ROUTCDE=11
         LA    R15,8
         ST    R15,RetCode
         PR         DROP  R12      USING Identify,12
*********************************************************************
*        Constants follow
*********************************************************************
SYSIN    DCB   RECFM=FB,DSORG=PS,MACRF=GM,DDNAME=SYSIN,               *
               LRECL=8Ø,EODAD=EndSysin
TAPEVOL  DCB   RECFM=U,LRECL=3276Ø,DSORG=PS,MACRF=R,DDNAME=TAPEVOL,    X
               BLKSIZE=3276Ø,EXLST=EXLST,SYNAD=TapeErr
EXLST    DS    ØF
         DC    X'87'
         DC    AL3(JFCB)
JFCB     DS    CL176TabSize
         DS    ØF
         DC    AL4(NumBytes)NumBytes EQU   (19999*6)+4+4      Counter +
position + 19999 volsers
NumRange DC    C'Ø123456789'
ChrRange DC    C'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
IOError  DS    C                  Open error flag
CharTab  DC    193X'ØØ'           Characters < A
         DC    9X'Ø1'             Range A - I
         DC    7X'ØØ'             I < character < J
         DC    9X'Ø1'             Range J - R
         DC    8X'ØØ'             R < character < S
         DC    8X'Ø1'             Range S - Z
         DC    6X'ØØ'             Z < character < 1
         DC    1ØX'Ø2'            Range Ø - 9
         DC    6X'ØØ'             Up to X'FF'
NextTab  DC    193X'ØØ'           Characters < A
         DC    C'BCDEFGHIJ'       Range A - I
         DC    7X'ØØ'             I < character < J
         DC    C'KLMNOPQRS'       Range J - R
         DC    8X'ØØ'             R < character < S
         DC    C'TUVWXYZ'         Range S - Z
         DC    6X'ØØ'             Z < character < 1
         DC    C'123456789Ø'      Range Ø - 9
         DC    5X'ØØ'             Up to X'FF'
```

```
BufferSz DS    ØF
         DC    AL4(BuffSize)
BuffSize EQU   3276Ø
         LTORG
*********************************************************************
*        DSECTs follow
*********************************************************************
GetMArea DSECT
SaveArea DS    18F                 General savearea
RetCode  DS    F                   Return code
InCard   DS    CL8Ø                Input card from SYSIN
FromVol  DS    CL6                 Start volume in range
NextVol  DS    CL6                 Next volume name in range
CurrVol  DS    CL6                 Volume we are currently looking at
ToVol    DS    CL6                 Last volume in range
LastVol  DS    C                   Flag to indicate last vol in range
RangePtr DS    F                   Pointer to position in ChrRange
NumVols  DS    F                   Number of volumes in the range
Table@   DS    F                   Address of range table
TapeRec  DS    CL33ØØØ             Buffer area to read tape record
GetMSize EQU   TapeRec-GetMArea    Size of area excluding buffer
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
No       EQU   X'ØØ'
Yes      EQU   X'8Ø'
         IEFJFCBN
         END
```

Tommy Afbeen
Senior Consultant
Brandenberg Consulting (South Africa)

An implementation of the DES algorithm on MVS

INTRODUCTION

The DES algorithm, a creation of IBM in the 1970s, is renowned and its use is pervasive in the IT world. Its strength (which is sometimes questioned) can be reinforced by using triple-DES, which widens the key-length to an equivalent of 112 bits (instead of the 56 bits that are in the standard DES). Triple-DES is simply the DES algorithm used three times successively, with two or three different DES keys.

RACF uses the 56-bit standard DES (with a slight transformation of the key) for password encryption. Because as it is a one-way function, it is unusable for any other cryptographic requirement (moreover, supervisor mode is demanded). For those interested in RACF encryption see the article on *Displaying user-ids with weak DES passwords* in *RACF Update*, Issue 2, November 1995.

DES proceeds by permutation and substitution on 64-bit blocks, using 56 bits of a 64-bit key (in fact 16 subkeys are extracted from the main DES key). Encryption and decryption require a 16-round process where a non-linear F function is used.

These programs do not provide you with an optimized implementation of DES on MVS, because no high-level language (even Assembler) achieves bit permutation or substitution in a quick and straightforward way. Should your cryptographic needs become important, I would recommend that you review IBM's ICSF product (which uses the cryptographic processor), or a much lighter software-only product, Megacrypt/MVS. The code shown below contains the following elements:

- Macros – for bit permutation or substitution:
  - DESPRM: bit permutation
  - DESF: F function of DES algorithm
  - DESPIP1: initial permutation IP1 of DES algorithm
  - DESPIF1: final permutation IP-1 of DES algorithm

  - DESFFP: F function (final 32 bits permutation)
  - DESFSB: F function (substitutions by S-BOX)
  - DESFEXPI: F function (initial expansion 32 bits to 48 bits)
  - DESSHIFT: circular shift 1 bit toward the left
  - DESPIPK2: final permutation on the 56-bit key (permutation PC-2)
  - DESPIPK1: initial permutation on the 64-bit key (permutation PC-1)
  - DESNXTK: DES subkey generation

- The subroutine DESKEYS – called by DESTEST to generate the 16 DES subkeys.
- The main program, DESTEST – encryption and decryption of a 8-byte field.

DESMACRO

```
         MACRO
&NLA     DESPRM     &N,&P,&O1,&O2,&R1,&R2,&W1,&OPTION
&NLA     DS    ØH
.*
.* DESPRM MACRO    : BIT PERMUTATION
.*
.* N=TO-POSITION    P=FROM-POSITION
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 64 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 64 BITS (MUST BE = Ø)
.* W1 WORK REGISTER
.* OPTION=NOINIT IF W1 NEEDS NOT BE INITIALISED
.*
         LCLA  &I,&J,&K,&Z
         LCLC  &INST
         GBLA  &ITER
&ITER    SETA  &ITER+1
DESPRM_ITER&ITER._TO&N._FROM&P    DS   ØH
&I       SETA  &P
&K       SETA  &N
         AIF   ('&OPTION' EQ 'NOINIT').CAPTUR
.*------------------------------------------------------------
.* LOAD AND SHIFT "1" TO ALIGN IT AT POSITION P
.*------------------------------------------------------------
         AIF   (&I EQ 32).NOSFT2
         AIF   (&I EQ 64).NOSFT2
         AIF   (&I LE 32).DEC1
&J       SETA  64-&I
         AGO   .SHFT$1
.DEC1    ANOP
&J       SETA  32-&I
.SHFT$1  ANOP
.*
.* IF LESS THAN 11 SHIFTS, LOAD IMMEDIATELY (2**&J)
.*
         AIF   (&J GT 11).SHFT$2
.*  2 POWER &J, RESULT IN &Z
&Z       SETA  2
.LOOPW   ANOP
         AIF   (&J EQ 1).FINW
&Z       SETA  &Z*2
&J       SETA  &J-1
         AGO   .LOOPW
.FINW    ANOP
         LA    &W1,&Z
         AGO   .NOSFT1
.*
.* IF MORE THAN 11 SHIFTS, LOAD A WORD
.*
.SHFT$2  ANOP
.* 2 POWER &J
         AIF   (&J NE 12).NOT12
         L     &W1,=F'4Ø96'      2  POWER      12
         AGO   .NOSFT1
.NOT12   ANOP
         AIF   (&J NE 13).NOT13
         L     &W1,=F'8192'      2  POWER      13
         AGO   .NOSFT1
.NOT13   ANOP
         AIF   (&J NE 14).NOT14
         L     &W1,=F'16384'     2  POWER      14
         AGO   .NOSFT1
.NOT14   ANOP
         AIF   (&J NE 15).NOT15
         L     &W1,=F'32768'     2  POWER      15
         AGO   .NOSFT1
.NOT15   ANOP
         AIF   (&J NE 16).NOT16
         L     &W1,=F'65536'     2  POWER      16
         AGO   .NOSFT1
.NOT16   ANOP
         AIF   (&J NE 17).NOT17
         L     &W1,=F'131Ø72'    2  POWER      17
         AGO   .NOSFT1
.NOT17   ANOP
         AIF   (&J NE 18).NOT18
         L     &W1,=F'262144'    2  POWER      18
         AGO   .NOSFT1
.NOT18   ANOP
         AIF   (&J NE 19).NOT19
         L     &W1,=F'524288'    2  POWER      19
         AGO   .NOSFT1
.NOT19   ANOP
         AIF   (&J NE 2Ø).NOT2Ø
         L     &W1,=F'1Ø48576'   2  POWER      2Ø
         AGO   .NOSFT1
.NOT2Ø   ANOP
         AIF   (&J NE 21).NOT21
         L     &W1,=F'2Ø97152'   2  POWER      21
         AGO   .NOSFT1
.NOT21   ANOP
         AIF   (&J NE 22).NOT22
         L     &W1,=F'41943Ø4'   2  POWER      22
         AGO   .NOSFT1
.NOT22   ANOP
         AIF   (&J NE 23).NOT23
         L     &W1,=F'83886Ø8'   2  POWER      23
         AGO   .NOSFT1
.NOT23   ANOP
         AIF   (&J NE 24).NOT24
         L     &W1,=F'16777216'  2  POWER      24
         AGO   .NOSFT1
.NOT24   ANOP
         AIF   (&J NE 25).NOT25
         L     &W1,=F'33554432'  2  POWER      25
         AGO   .NOSFT1
.NOT25   ANOP
         AIF   (&J NE 26).NOT26
         L     &W1,=F'671Ø8864'  2  POWER      26
         AGO   .NOSFT1
.NOT26   ANOP
         AIF   (&J NE 27).NOT27
         L     &W1,=F'134217728' 2  POWER      27
         AGO   .NOSFT1
.NOT27   ANOP
         AIF   (&J NE 28).NOT28
         L     &W1,=F'268435456' 2  POWER      28
         AGO   .NOSFT1
.NOT28   ANOP
         AIF   (&J NE 29).NOT29
         L     &W1,=F'53687Ø912' 2  POWER      29
         AGO   .NOSFT1
.NOT29   ANOP
         AIF   (&J NE 3Ø).NOT3Ø
         L     &W1,=F'1Ø73741824' 2  POWER      3Ø
         AGO   .NOSFT1
.NOT3Ø   ANOP
         AIF   (&J NE 31).NOT31
         L     &W1,=F'-2147483648'       2  POWER      31
         AGO   .NOSFT1
.NOT31   ANOP
         AGO   .NOSFT1
.*
.* IF POSITION 32 OR 64, LOAD 1  (NO SHIFTING)
.*
.NOSFT2  ANOP
         LA    &W1,1            LAST BIT = 1
.*
.NOSFT1  ANOP
.CAPTUR  ANOP
.*------------------------------------------------------------
.* CAPTURE ORIGINAL BIT IN &W1
.*------------------------------------------------------------
         AIF   (&I GT 32).CAPT1
         NR    &W1,&O1          TAKE BIT IN FROM-REGISTER
         AGO   .CAPTF
.CAPT1   ANOP
         NR    &W1,&O2          TAKE BIT IN FROM-REGISTER
.CAPTF   ANOP
.*------------------------------------------------------------
.* SET THE BIT AT TO-POSITION
.*------------------------------------------------------------
         AIF   (&I LE 32).DEC2
&I       SETA  &I-32
.DEC2    ANOP
         AIF   (&K LE 32).DEC3
&K       SETA  &K-32
.DEC3    ANOP
         AIF   (&I EQ &K).NOSFT3
         AIF   (&I LT &K).DECA2
&Z       SETA  &I-&K
&INST    SETC  'SLL'
         AGO   .DODEC2
.DECA2   ANOP
&Z       SETA  &K-&I
&INST    SETC  'SRL'
.DODEC2  ANOP
         &INST &W1,&Z       SHIFT LEFT OR RIGHT TOWARD TO-POSITION
.NOSFT3  ANOP
.*------------------------------------------------------------
.* STORE RESULT
.*------------------------------------------------------------
&K       SETA  &N
         AIF   (&K GT 32).STO1
         OR    &R1,&W1
         MEXIT
.STO1    ANOP
         OR    &R2,&W1
         MEXIT
```

```
.* END OF MACRO DESPRM
         MEND
         MACRO
&NLA     DESF  &O,&W1,&W2,&W3,&W9,&KEY1,&KEY2
&NLA     DS    ØH
.*
.* F FUNCTION           DES ALGORITHM
.*
.* O     REGISTER CONTAINING THE ORIGINAL 32 BITS OR PREVIOUS RI
.*       WILL CONTAIN THE 32 BITS RESULTING FOR THE ROUND
.* W1    WORK REGISTER
.* W23   WORK REGISTER PAIR
.* W9    WORK REGISTER
.* KEY1  4 FIRST BYTES OF KI KEY
.* KEY2  4 NEXT BYTES OF KI KEY
.*
.* INITIAL EXPANSION 32 BITS -> 48 BITS
.*
         DESFEXPI &O,&W2,&W3,&W1
.*
.* XOR WITH 48 BITS OF KEY
.*
         ICM   &W1,15,&KEY1
         XR    &W2,&W1
         ICM   &W1,12,&KEY2
         XR    &W3,&W1
.*
.* 8 SUBSTITUTION BOXES : 8X6 BITS -> 8X4 BITS  (48->32 BITS)
.*
         XR    &W9,&W9
         DESFSB   1,&W2,&W3,&W9,&W1
         DESFSB   2,&W2,&W3,&W9,&W1
         DESFSB   3,&W2,&W3,&W9,&W1
         DESFSB   4,&W2,&W3,&W9,&W1
         DESFSB   5,&W2,&W3,&W9,&W1
         DESFSB   6,&W2,&W3,&W9,&W1
         DESFSB   7,&W2,&W3,&W9,&W1
         DESFSB   8,&W2,&W3,&W9,&W1
.*
.* FINAL PERMUTATION
.*
         DESFFP  &W9,&O,&W1
.* END OF MACRO DESF
         MEND
         MACRO
&NLA     DESPIP1          &O1,&O2,&R1,&R2,&W1
&NLA     DS    ØH
.*
.* INITIAL PERMUTATION IP1 OF DES ALGORITHM
.*
```

```
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 64 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 64 BITS
.* W1    WORK REGISTER
.*
         XR    &R1,&R1   RESULT=0
         XR    &R2,&R2   RESULT=0
         DESPRM     01,58,&O1,&O2,&R1,&R2,&W1
         DESPRM     02,50,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'4194305'      X'00400001'   BITS 42 64
         DESPRM     03,42,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'1073742080'   X'40000100'   BITS 34 56
         DESPRM     04,34,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     05,26,&O1,&O2,&R1,&R2,&W1
         DESPRM     06,18,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'4194305'      X'00400001'   BITS 10 32
         DESPRM     07,10,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'1073742080'   X'40000100'   BITS 2 24
         DESPRM     08,02,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     09,60,&O1,&O2,&R1,&R2,&W1
         DESPRM     10,52,&O1,&O2,&R1,&R2,&W1
         DESPRM     11,44,&O1,&O2,&R1,&R2,&W1
         DESPRM     12,36,&O1,&O2,&R1,&R2,&W1
         DESPRM     13,28,&O1,&O2,&R1,&R2,&W1
         DESPRM     14,20,&O1,&O2,&R1,&R2,&W1
         DESPRM     15,12,&O1,&O2,&R1,&R2,&W1
         DESPRM     16,04,&O1,&O2,&R1,&R2,&W1
         DESPRM     17,62,&O1,&O2,&R1,&R2,&W1
         DESPRM     18,54,&O1,&O2,&R1,&R2,&W1
         DESPRM     19,46,&O1,&O2,&R1,&R2,&W1
         DESPRM     20,38,&O1,&O2,&R1,&R2,&W1
         DESPRM     21,30,&O1,&O2,&R1,&R2,&W1
         DESPRM     22,22,&O1,&O2,&R1,&R2,&W1
         DESPRM     23,14,&O1,&O2,&R1,&R2,&W1
         DESPRM     24,06,&O1,&O2,&R1,&R2,&W1
         DESPRM     27,48,&O1,&O2,&R1,&R2,&W1
         DESPRM     28,40,&O1,&O2,&R1,&R2,&W1
         DESPRM     31,16,&O1,&O2,&R1,&R2,&W1
         DESPRM     32,08,&O1,&O2,&R1,&R2,&W1
         DESPRM     33,57,&O1,&O2,&R1,&R2,&W1
         DESPRM     34,49,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'8388610'           X'00800002'  BITS 41 63
         DESPRM     35,41,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'-2147483136'       X'80000200'  BITS 33 55
         DESPRM     36,33,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     37,25,&O1,&O2,&R1,&R2,&W1
         DESPRM     38,17,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'8388610'           X'00800002'  BITS 9 31
         DESPRM     39,09,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'-2147483136'       X'80000200'  BITS 1 23
         DESPRM     40,01,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     41,59,&O1,&O2,&R1,&R2,&W1
         DESPRM     42,51,&O1,&O2,&R1,&R2,&W1
         DESPRM     43,43,&O1,&O2,&R1,&R2,&W1
         DESPRM     44,35,&O1,&O2,&R1,&R2,&W1
         DESPRM     45,27,&O1,&O2,&R1,&R2,&W1
         DESPRM     46,19,&O1,&O2,&R1,&R2,&W1
         DESPRM     47,11,&O1,&O2,&R1,&R2,&W1
         DESPRM     48,Ø3,&O1,&O2,&R1,&R2,&W1
         DESPRM     49,61,&O1,&O2,&R1,&R2,&W1
         DESPRM     5Ø,53,&O1,&O2,&R1,&R2,&W1
         DESPRM     51,45,&O1,&O2,&R1,&R2,&W1
         DESPRM     52,37,&O1,&O2,&R1,&R2,&W1
         DESPRM     53,29,&O1,&O2,&R1,&R2,&W1
         DESPRM     54,21,&O1,&O2,&R1,&R2,&W1
         DESPRM     55,13,&O1,&O2,&R1,&R2,&W1
         DESPRM     56,Ø5,&O1,&O2,&R1,&R2,&W1
         DESPRM     59,47,&O1,&O2,&R1,&R2,&W1
         DESPRM     6Ø,39,&O1,&O2,&R1,&R2,&W1
         DESPRM     63,15,&O1,&O2,&R1,&R2,&W1
         DESPRM     64,Ø7,&O1,&O2,&R1,&R2,&W1
.* END OF MACRO DESPIP1
         MEND
         MACRO
&NLA     DESPIF1          &O1,&O2,&R1,&R2,&W1
&NLA     DS    ØH
.*
.* FINAL PERMUTATION IP-1 OF DES ALGORITHM
.*
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 64 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 64 BITS
.* W1    WORK REGISTER
.*
         XR    &R1,&R1                   RESULT=Ø
         XR    &R2,&R2                   RESULT=Ø
         L     &W1,=F'1677722Ø'         X'Ø1ØØØØØ4'  BITS 4Ø 62
         DESPRM     Ø1,4Ø,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'1677722Ø'         X'Ø1ØØØØØ4'  BITS 8 3Ø
         DESPRM     Ø2,Ø8,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     Ø3,48,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø4,16,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø5,56,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø6,24,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø7,64,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø8,32,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'3355444Ø'         X'Ø2ØØØØØ8'  BITS 39 61
         DESPRM     Ø9,39,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'3355444Ø'         X'Ø2ØØØØØ8'  BITS 7 29
         DESPRM     1Ø,Ø7,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     11,47,&O1,&O2,&R1,&R2,&W1
         DESPRM     12,15,&O1,&O2,&R1,&R2,&W1
         DESPRM     13,55,&O1,&O2,&R1,&R2,&W1
         DESPRM     14,23,&O1,&O2,&R1,&R2,&W1
         DESPRM     15,63,&O1,&O2,&R1,&R2,&W1
         DESPRM     16,31,&O1,&O2,&R1,&R2,&W1
         DESPRM     17,38,&O1,&O2,&R1,&R2,&W1
         DESPRM     18,Ø6,&O1,&O2,&R1,&R2,&W1
         DESPRM     19,46,&O1,&O2,&R1,&R2,&W1
         DESPRM     2Ø,14,&O1,&O2,&R1,&R2,&W1
         DESPRM     21,54,&O1,&O2,&R1,&R2,&W1
         DESPRM     22,22,&O1,&O2,&R1,&R2,&W1
         DESPRM     25,37,&O1,&O2,&R1,&R2,&W1
         DESPRM     26,Ø5,&O1,&O2,&R1,&R2,&W1
         DESPRM     27,45,&O1,&O2,&R1,&R2,&W1
         DESPRM     28,13,&O1,&O2,&R1,&R2,&W1
         DESPRM     29,53,&O1,&O2,&R1,&R2,&W1
         DESPRM     3Ø,21,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'26843552Ø'           X'1ØØØØØ4Ø'   BITS 36 58
         DESPRM     33,36,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'26843552Ø'           X'1ØØØØØ4Ø'   BITS 4 26
         DESPRM     34,Ø4,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     35,44,&O1,&O2,&R1,&R2,&W1
         DESPRM     36,12,&O1,&O2,&R1,&R2,&W1
         DESPRM     37,52,&O1,&O2,&R1,&R2,&W1
         DESPRM     38,2Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     39,6Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     4Ø,28,&O1,&O2,&R1,&R2,&W1
         L     &W1,=F'53687104Ø'           X'2ØØØØØ8Ø'   BITS 35 57
         DESPRM     41,35,&O1,&O2,&R1,&R2,&W1,NOINIT
         L     &W1,=F'53687104Ø'           X'2ØØØØØ8Ø'   BITS 3 25
         DESPRM     42,Ø3,&O1,&O2,&R1,&R2,&W1,NOINIT
         DESPRM     43,43,&O1,&O2,&R1,&R2,&W1
         DESPRM     44,11,&O1,&O2,&R1,&R2,&W1
         DESPRM     45,51,&O1,&O2,&R1,&R2,&W1
         DESPRM     46,19,&O1,&O2,&R1,&R2,&W1
         DESPRM     47,59,&O1,&O2,&R1,&R2,&W1
         DESPRM     48,27,&O1,&O2,&R1,&R2,&W1
         DESPRM     49,34,&O1,&O2,&R1,&R2,&W1
         DESPRM     5Ø,Ø2,&O1,&O2,&R1,&R2,&W1
         DESPRM     51,42,&O1,&O2,&R1,&R2,&W1
         DESPRM     52,1Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     53,5Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     54,18,&O1,&O2,&R1,&R2,&W1
         DESPRM     57,33,&O1,&O2,&R1,&R2,&W1
         DESPRM     58,Ø1,&O1,&O2,&R1,&R2,&W1
         DESPRM     59,41,&O1,&O2,&R1,&R2,&W1
         DESPRM     6Ø,Ø9,&O1,&O2,&R1,&R2,&W1
         DESPRM     61,49,&O1,&O2,&R1,&R2,&W1
         DESPRM     62,17,&O1,&O2,&R1,&R2,&W1
.* END OF MACRO DESPIF1
         MEND
         MACRO
&NLA     DESFFP           &O,&R,&W1
&NLA     DS    ØH
.*
.* FINAL 32 BITS PERMUTATION  (F FUNCTION OF DES ALGORITHM)
.*
.* O  REGISTER CONTAINING THE ORIGINAL 32 BITS
.* R  REGISTER CONTAINING THE RESULTING 32 BITS
.* W1 WORK REGISTER
.*
         XR    &R,&R
         DESPRM     Ø1,16,&O,,&R,,&W1
         L     &W1,=F'6144'      X'18ØØ'
         DESPRM     Ø3,2Ø,&O,,&R,,&W1,NOINIT       SHIFTING 17
         L     &W1,=F'33685792'   X'Ø2Ø2Ø12Ø'
         DESPRM     Ø2,Ø7,&O,,&R,,&W1,NOINIT       SHIFTING 5
         L     &W1,=F'446464Ø'     '442ØØØ'
         DESPRM     16,1Ø,&O,,&R,,&W1,NOINIT       SHIFTING 6 RIGHT
         L     &W1,=F'-2Ø1326592Ø'
         DESPRM     Ø9,Ø1,&O,,&R,,&W1,NOINIT       SHIFTING 8 RIGHT
         LA    &W1,1Ø24+128    B'Ø1ØØ1ØØØØØØØ'  POS 22 25
         DESPRM     29,22,&O,,&R,,&W1,NOINIT       SHIFTING 7 RIGHT
         L     &W1,=F'1Ø8213Ø432'     X'4Ø8ØØØØØ'
         DESPRM     17,Ø2,&O,,&R,,&W1,NOINIT       SHIFTING 15 RIGHT
         DESPRM     Ø5,29,&O,,&R,,&W1
         DESPRM     Ø6,12,&O,,&R,,&W1
         DESPRM     Ø7,28,&O,,&R,,&W1
         DESPRM     Ø8,17,&O,,&R,,&W1
         DESPRM     11,23,&O,,&R,,&W1
         DESPRM     12,26,&O,,&R,,&W1
         DESPRM     14,18,&O,,&R,,&W1
         DESPRM     15,31,&O,,&R,,&W1
         DESPRM     18,Ø8,&O,,&R,,&W1
         DESPRM     21,32,&O,,&R,,&W1
         DESPRM     23,Ø3,&O,,&R,,&W1
         DESPRM     26,13,&O,,&R,,&W1
         DESPRM     27,3Ø,&O,,&R,,&W1
         DESPRM     28,Ø6,&O,,&R,,&W1
         DESPRM     3Ø,11,&O,,&R,,&W1
         DESPRM     31,Ø4,&O,,&R,,&W1
.* END OF MACRO DESFFP
         MEND
         MACRO
&NLA     DESFSB        &N,&O1,&O2,&R,&W1
&NLA     DS    ØH
.*
.* F FUNCTION - SUBSTITUTION S-BOX, INDEX &N     DES  ALGORITHM
.*
.* N     1 - 8  IS BOX NUMBER  (INVOKE IN ASCENDING ORDER)
.* O1 O2 DOUBLE REGISTER CONTAINING THE 8X6=48 ORIGINAL BITS
.* R     REGISTER  CONTAINING THE 8X4=32 RESULTING BITS (=Ø INITIALY)
.* W1    WORK REGISTER
.*
.* THE BOXES SBOX1 -> SBOX8 MUST BE DEFINED
.*
         LCLA  &I
.*
DESFSB_SB&SYSNDX  DS  ØH
         L     &W1,=F'-671Ø8864'    X'FCØØØØØØ'
         NR    &W1,&O1    CATCH 6 BITS IN ORIGIN REGS
         SRL   &W1,26     LAST BYTE REGISTER B'ØØXXXXXX'
         AIF   ('&N' NE '1').NODEB
.NODEB   ANOP
         IC    &W1,SBOX&N.(&W1)     TRANSFORM BYTE 6 BITS / 4 BITS
         AIF   ('&N' EQ '8').NOSFT
&I       SETA  &N
&I       SETA  (8-&I)*4
         SLL   &W1,&I     SHIFTING FOR STORING IN RESULT
.NOSFT   ANOP
         OR    &R,&W1     STORE THE 4 BITS IN THE RESULT
         SLDL  &O1,6      PREPARE NEXT 6 BITS
.* END OF MACRO DESFSB
         MEND
         MACRO
&NLA     DESFEXPI         &O1,&R1,&R2,&W1
&NLA     DS    ØH
.*
.* F FUNCTION : INITIAL EXPANSION 32 BITS -> 48 BITS  DES ALGORITHM
.*
.* O1    REGISTER  CONTAINING THE ORIGINAL 32 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 48 BITS
.* W1    WORK REGISTER
.*
         XR    &R1,&R1                   RESULT=Ø
         XR    &R2,&R2                   RESULT=Ø
         DESPRM     Ø1,32,&O1,,&R1,&R2,&W1
         L     &W1,=F'-134217728'
         DESPRM     Ø2,Ø1,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'5284823Ø4'       X'1F8ØØØØØ'
         DESPRM     Ø7,Ø4,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'33Ø3Ø144'        X'Ø1F8ØØØØ'
         DESPRM     13,Ø8,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'2Ø64384'         X'ØØ1F8ØØØ'
         DESPRM     19,12,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'129Ø24'          X'ØØØ1F8ØØ'
         DESPRM     25,16,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'6144'            X'18ØØ'       POSITIONS 2Ø 21
         DESPRM     31,2Ø,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'192Ø'            X'Ø78Ø'       POSITIONS 22 - 25
         DESPRM     33,22,&O1,,&R1,&R2,&W1,NOINIT
         L     &W1,=F'5Ø4'             X'1F8'
         DESPRM     37,24,&O1,,&R1,&R2,&W1,NOINIT
         LA    &W1,31                   BITS 1 - 5 : POSITIONS 28 - 32
         DESPRM     43,28,&O1,,&R1,&R2,&W1,NOINIT
         DESPRM     48,Ø1,&O1,,&R1,&R2,&W1
.* END OF MACRO DESFEXPI
         MEND
         MACRO
&NLA     DESSHIFT  &O1,&W1
&NLA     DS    ØH
.*
.* CIRCULAR SHIFT 1 BIT TOWARD THE LEFT   DES ALGORITHM
.* OPERATING ON THE 28 BITS ON LEFT
.*
.* O1 REGISTER CONTAINING ORIGINAL 28 BITS AND RESULT
.* W1 WORK REGISTER
.*
DESSHIFT_LAB&SYSNDX     DS   ØH
.*
         L     &W1,=F'-2147483648'      B'1ØØØØØ...ØØ'
         NR    &W1,&O1                  CATCH 1ST BIT ON THE LEFT
         SRL   &W1,27                   PREPARE FUTURE POSITION 28
         SLL   &O1,1                    ** SHIFT REGISTER **
         OR    &O1,&W1                  BIT OF CIRCULAR SHIFT
         MEXIT
         MEND
         MACRO
&NLA     DESPIPK2         &O1,&O2,&R1,&R2,&W1
&NLA     DS    ØH
.*
.* FINAL PERMUTATION ON THE 56-BIT KEY - DES ALGORITHM
.* PERMUTATION PC-2  (PERMUTED CHOICE 2)
.*
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 56 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 48 BITS
.* W1 WORK REGISTER
.*
         XR    &R1,&R1   RESULT=Ø
         XR    &R2,&R2   RESULT=Ø
         DESPRM     Ø1,14,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø2,17,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø3,11,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø4,24,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø5,Ø1,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø6,Ø5,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø7,Ø3,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø8,28,&O1,&O2,&R1,&R2,&W1
         DESPRM     Ø9,15,&O1,&O2,&R1,&R2,&W1
         DESPRM     1Ø,Ø6,&O1,&O2,&R1,&R2,&W1
         DESPRM     11,21,&O1,&O2,&R1,&R2,&W1
         DESPRM     12,1Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     13,23,&O1,&O2,&R1,&R2,&W1
         DESPRM     14,19,&O1,&O2,&R1,&R2,&W1
         DESPRM     15,12,&O1,&O2,&R1,&R2,&W1
         DESPRM     16,Ø4,&O1,&O2,&R1,&R2,&W1
         DESPRM     17,26,&O1,&O2,&R1,&R2,&W1
         DESPRM     18,Ø8,&O1,&O2,&R1,&R2,&W1
         DESPRM     19,16,&O1,&O2,&R1,&R2,&W1
         DESPRM     2Ø,Ø7,&O1,&O2,&R1,&R2,&W1
         DESPRM     21,27,&O1,&O2,&R1,&R2,&W1
         DESPRM     22,2Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     23,13,&O1,&O2,&R1,&R2,&W1
         DESPRM     24,Ø2,&O1,&O2,&R1,&R2,&W1
         DESPRM     25,41,&O1,&O2,&R1,&R2,&W1
         DESPRM     26,52,&O1,&O2,&R1,&R2,&W1
         DESPRM     27,31,&O1,&O2,&R1,&R2,&W1
         DESPRM     28,37,&O1,&O2,&R1,&R2,&W1
         DESPRM     29,47,&O1,&O2,&R1,&R2,&W1
         DESPRM     3Ø,55,&O1,&O2,&R1,&R2,&W1
         DESPRM     31,3Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     32,4Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     33,51,&O1,&O2,&R1,&R2,&W1
         DESPRM     34,45,&O1,&O2,&R1,&R2,&W1
         DESPRM     35,33,&O1,&O2,&R1,&R2,&W1
         DESPRM     36,48,&O1,&O2,&R1,&R2,&W1
         DESPRM     37,44,&O1,&O2,&R1,&R2,&W1
         DESPRM     38,49,&O1,&O2,&R1,&R2,&W1
         DESPRM     39,39,&O1,&O2,&R1,&R2,&W1
         DESPRM     4Ø,56,&O1,&O2,&R1,&R2,&W1
         DESPRM     41,34,&O1,&O2,&R1,&R2,&W1
         DESPRM     42,53,&O1,&O2,&R1,&R2,&W1
         DESPRM     43,46,&O1,&O2,&R1,&R2,&W1
         DESPRM     44,42,&O1,&O2,&R1,&R2,&W1
         DESPRM     45,5Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     46,36,&O1,&O2,&R1,&R2,&W1
         DESPRM     47,29,&O1,&O2,&R1,&R2,&W1
         DESPRM     48,32,&O1,&O2,&R1,&R2,&W1
         MEND
         MACRO
&NLA     DESPIPK1         &O1,&O2,&R1,&R2,&W1
&NLA     DS    ØH
.*
.* INITIAL PERMUTATION ON THE 64-BIT KEY - DES ALGORITHM
.* PERMUTATION PC-1  (PERMUTED CHOICE 1)
.*
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 64 BITS
.* R1 R2 REGISTERS CONTAINING THE RESULTING 56 BITS
.*    R1 WILL CONTAIN THE FIRST 28 BITS, R2 THE FOLLOWING 28 BITS
.* W1 WORK REGISTER
.*
         XR    &R1,&R1   RESULT=Ø
         XR    &R2,&R2   RESULT=Ø
         DESPRM    Ø1,57,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø2,49,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø3,41,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø4,33,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø5,25,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø6,17,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø7,Ø9,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø8,Ø1,&O1,&O2,&R1,&R2,&W1
         DESPRM    Ø9,58,&O1,&O2,&R1,&R2,&W1
         DESPRM    1Ø,5Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM    11,42,&O1,&O2,&R1,&R2,&W1
         DESPRM    12,34,&O1,&O2,&R1,&R2,&W1
         DESPRM    13,26,&O1,&O2,&R1,&R2,&W1
         DESPRM    14,18,&O1,&O2,&R1,&R2,&W1
         DESPRM    15,1Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM    16,Ø2,&O1,&O2,&R1,&R2,&W1
         DESPRM    17,59,&O1,&O2,&R1,&R2,&W1
         DESPRM    18,51,&O1,&O2,&R1,&R2,&W1
         DESPRM    19,43,&O1,&O2,&R1,&R2,&W1
         DESPRM    2Ø,35,&O1,&O2,&R1,&R2,&W1
         DESPRM    21,27,&O1,&O2,&R1,&R2,&W1
         DESPRM    22,19,&O1,&O2,&R1,&R2,&W1
         DESPRM    23,11,&O1,&O2,&R1,&R2,&W1
         DESPRM    24,Ø3,&O1,&O2,&R1,&R2,&W1
         DESPRM    25,6Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM    26,52,&O1,&O2,&R1,&R2,&W1
         DESPRM    27,44,&O1,&O2,&R1,&R2,&W1
         DESPRM    28,36,&O1,&O2,&R1,&R2,&W1
         DESPRM    29,63,&O1,&O2,&R1,&R2,&W1
         DESPRM    3Ø,55,&O1,&O2,&R1,&R2,&W1
         DESPRM    31,47,&O1,&O2,&R1,&R2,&W1
         DESPRM    32,39,&O1,&O2,&R1,&R2,&W1
         DESPRM    33,31,&O1,&O2,&R1,&R2,&W1
         DESPRM    34,23,&O1,&O2,&R1,&R2,&W1
         DESPRM    35,15,&O1,&O2,&R1,&R2,&W1
         DESPRM    36,Ø7,&O1,&O2,&R1,&R2,&W1
         DESPRM    37,62,&O1,&O2,&R1,&R2,&W1
         DESPRM    38,54,&O1,&O2,&R1,&R2,&W1
         DESPRM    39,46,&O1,&O2,&R1,&R2,&W1
         DESPRM    4Ø,38,&O1,&O2,&R1,&R2,&W1
         DESPRM    41,3Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM    42,22,&O1,&O2,&R1,&R2,&W1
         DESPRM    43,14,&O1,&O2,&R1,&R2,&W1
         DESPRM    44,Ø6,&O1,&O2,&R1,&R2,&W1
         DESPRM    45,61,&O1,&O2,&R1,&R2,&W1
         DESPRM    46,53,&O1,&O2,&R1,&R2,&W1
         DESPRM    47,45,&O1,&O2,&R1,&R2,&W1
         DESPRM    48,37,&O1,&O2,&R1,&R2,&W1
         DESPRM    49,29,&O1,&O2,&R1,&R2,&W1
         DESPRM    5Ø,21,&O1,&O2,&R1,&R2,&W1
         DESPRM    51,13,&O1,&O2,&R1,&R2,&W1
         DESPRM    52,Ø5,&O1,&O2,&R1,&R2,&W1
         DESPRM     53,28,&O1,&O2,&R1,&R2,&W1
         DESPRM     54,2Ø,&O1,&O2,&R1,&R2,&W1
         DESPRM     55,12,&O1,&O2,&R1,&R2,&W1
         DESPRM     56,Ø4,&O1,&O2,&R1,&R2,&W1
.* DISPATCH 28 BITS IN R1 AND 28 IN R2
         SRDL       &R1,4      28 BITS IN &R2
         SLL        &R1,4      28 BITS IN &R1
         MEND
         MACRO
&NLA     DESNXTK       &N,&O1,&O2,&R1,&R2,&K1,&K2,&W1,&W2,&W3
&NLA     DS    ØH
.*
.* THIS MACRO GENERATE SUBKEY KI+1 FROM SUBKEY KI
.* N=1-16 : ROUND NUMBER
.* O1 O2 REGISTERS CONTAINING THE ORIGINAL 64 BITS (1ST ROUND) OR
.*       THE 56 BITS CI DI FROM PREVIOUS ROUND
.* R1 R2 REGISTER PAIR CONTAINING RESULTING 56 BITS (CI+1 DI+1)
.* K1 K2 REGISTERS CONTAINING 48 BITS OF SUBKEY KI+1
.* W1-3  WORK REGISTERS
.*
         AIF  ('&N' NE '1').NOTFIRS
.*
.* FIRST ROUND : PERMUTATION PC-1
.*
         DESPIPK1  &O1,&O2,&R1,&R2,&W1
         AGO .DOSHFT
.*
.NOTFIRS  ANOP
         LR     &R1,&O1
         LR     &R2,&O2
.*
.DOSHFT   ANOP
.*
.* CIRCULAR SHIFTS (1 OR 2 BITS)
.*
         DESSHIFT  &R1,&W1
         DESSHIFT  &R2,&W1
         AIF  ('&N' EQ '1' OR '&N' EQ '2').ONESHFT
         AIF  ('&N' EQ '9' OR '&N' EQ '16').ONESHFT
         DESSHIFT  &R1,&W1
         DESSHIFT  &R2,&W1
.ONESHFT   ANOP
.*
.* SAVE CI+1 DI+1 BEFORE PERMUTATION
.*
         LR     &W2,&R1
         LR     &W3,&R2
.*
.* CI+1 DI+1 COUPLED TO GET 56 CONTIGUOUS BITS
.*
         SRL    &R1,4
         SLDL   &R1,4
.*
.* PERMUTATION PC-2 TO CREATE FINAL KEY
.*
         DESPIPK2  &R1,&R2,&K1,&K2,&W1
.*
.* RESTORE CI+1 DI+1
.*
         LR     &O1,&W2
         LR     &O2,&W3
         MEND
```

DESTEST

```
*
*---------------------------------------------------------------------*
* EXAMPLE OF A DES ENCRYPTION/DECRYPTION PROGRAM                      *
*                                                                     *
* THE 8-BYTE TEXT "ITSCLEAR" IS ENCIPHERED AND DECIPHERED             *
*    USING A DES KEY "WATERLOO".                                      *
*---------------------------------------------------------------------*
*
DESTEST  CSECT
DESTEST  AMODE 31
DESTEST  RMODE ANY
         USING *,R15
         SAVE  (14,12),,DESTEST-&SYSDATC-&SYSTIME
         DROP  R15
         LR    R12,R15                 1ST BASE REGISTER
         LR    R11,R12                 2ND BASE REGISTER
         LA    R11,4Ø95(Ø,R11)         INIT 2ND BASE REGISTER
         LA    R11,1(Ø,R11)            INIT 2ND BASE REGISTER
         USING DESTEST,R12,R11
         ST    R13,SAVE+4
         LA    R13,SAVE
*---------------------------------------------------------------------*
* 1. GENERATING THE 16 DES SUBKEYS EXTRACTED FROM THE DES 56-BIT KEY  *
*---------------------------------------------------------------------*
         LA    R1,WKEYS          POINT TO SUBKEYS + DESKEY
         CALL  DESKEYS
*---------------------------------------------------------------------*
* 2. ENCRYPTION ROUTINE - INPUT=R1-R2   OUTPUT=R1-R2                  *
*---------------------------------------------------------------------*
ENCRYPT  DS    ØH
         ICM   R1,15,TEXT            LOAD CLEAR TEXT
         ICM   R2,15,TEXT+4          LOAD CLEAR TEXT
         LA    R8,16                 NUMBER OF DES ROUNDS
* SET ADDRESS OF 1ST SUB-KEY
         XR    R3,R3
         XR    R4,R4
         LA    R9,WKEYS              ACCESS 1ST SUB-KEY
```

```
* INITIAL PERMUTATION, RESULT IN R3-R4
         DESPIP1   R1,R2,R3,R4,R5
* DES F FUNCTION F, ROUND 1 - 16
DESROUND_ENCRYPT  DS   ØH
         STCM  Name: Deskeys.txt

*
*-----------------------------------------------*
* SUBROUTINE FOR CREATING THE 16 DES SUBKEYS    *
*-----------------------------------------------*
*
* PARAMETERS : R1  ->  16X6 BYTES FOR THE 16 SUBKEYS + 8 BYTES DES KEY
*
DESKEYS  CSECT
DESKEYS  AMODE 31
DESKEYS  RMODE ANY
         USING *,R15
         SAVE  (14,12),,DESKEYS-&SYSDATC-&SYSTIME
         DROP  R15
         LR    R12,R15
         USING DESKEYS,R12,R4,R5     3 BASE REGISTERS
         LR    R4,R12
         LA    R4,4Ø95(Ø,R4)
         LA    R4,1(Ø,R4)
         LR    R5,R4
         LA    R5,4Ø95(Ø,R5)
         LA    R5,1(Ø,R5)
         LR    R3,R1                 ADDRESSING PARAMETER
         USING PARAM,R3
         ICM   R6,15,CLE             LOAD THE KEY
         ICM   R7,15,CLE+4 LOAD THE KEY
* COMPUTING THE 16 DES SUBKEYS
         DESNXTK  1,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY1
         STCM  R14,12,KEY1+4
         DESNXTK  2,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY2
         STCM  R14,12,KEY2+4
         DESNXTK  3,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY3
         STCM  R14,12,KEY3+4
         DESNXTK  4,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY4
         STCM  R14,12,KEY4+4
         DESNXTK  5,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY5
         STCM  R14,12,KEY5+4
         DESNXTK  6,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY6
         STCM  R14,12,KEY6+4
         DESNXTK  7,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY7
         STCM  R14,12,KEY7+4
         DESNXTK  8,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY8
         STCM  R14,12,KEY8+4
         DESNXTK  9,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY9
         STCM  R14,12,KEY9+4
         DESNXTK  1Ø,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY1Ø
         STCM  R14,12,KEY1Ø+4
         DESNXTK  11,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY11
         STCM  R14,12,KEY11+4
         DESNXTK  12,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY12
         STCM  R14,12,KEY12+4
         DESNXTK  13,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY13
         STCM  R14,12,KEY13+4
         DESNXTK  14,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY14
         STCM  R14,12,KEY14+4
         DESNXTK  15,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY15
         STCM  R14,12,KEY15+4
         DESNXTK  16,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY16
         STCM  R14,12,KEY16+4
         LM    R14,R12,12(R13)      RESTORE REGISTERS
         BR    R14
         LTORG
PARAM    DSECT
KEY1     DS    CL6
KEY2     DS    CL6
KEY3     DS    CL6
KEY4     DS    CL6
KEY5     DS    CL6
KEY6     DS    CL6
KEY7     DS    CL6
KEY8     DS    CL6
KEY9     DS    CL6
KEY1Ø    DS    CL6
KEY11    DS    CL6
KEY12    DS    CL6
KEY13    DS    CL6
KEY14    DS    CL6
KEY15    DS    CL6
KEY16    DS    CL6
CLE      DS    CL8
         END
```

DESKEYS

```
*
*----------------------------------------------*
* SUBROUTINE FOR CREATING THE 16 DES SUBKEYS    *
*----------------------------------------------*
*
* PARAMETERS : R1  ->  16X6 BYTES FOR THE 16 SUBKEYS + 8 BYTES DES KEY
*
DESKEYS  CSECT
DESKEYS  AMODE 31
DESKEYS  RMODE ANY
         USING *,R15
         SAVE  (14,12),,DESKEYS-&SYSDATC-&SYSTIME
         DROP  R15
         LR    R12,R15
         USING DESKEYS,R12,R4,R5     3 BASE REGISTERS
         LR    R4,R12
         LA    R4,4Ø95(Ø,R4)
         LA    R4,1(Ø,R4)
         LR    R5,R4
         LA    R5,4Ø95(Ø,R5)
         LA    R5,1(Ø,R5)
         LR    R3,R1                 ADDRESSING PARAMETER
         USING PARAM,R3
         ICM   R6,15,CLE             LOAD THE KEY
         ICM   R7,15,CLE+4 LOAD THE KEY
* COMPUTING THE 16 DES SUBKEYS
         DESNXTK  1,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY1
         STCM  R14,12,KEY1+4
         DESNXTK  2,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY2
         STCM  R14,12,KEY2+4
         DESNXTK  3,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY3
         STCM  R14,12,KEY3+4
         DESNXTK  4,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY4
         STCM  R14,12,KEY4+4
         DESNXTK  5,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY5
         STCM  R14,12,KEY5+4
         DESNXTK  6,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY6
         STCM  R14,12,KEY6+4
         DESNXTK  7,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY7
         STCM  R14,12,KEY7+4
         DESNXTK  8,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY8
         STCM  R14,12,KEY8+4
         DESNXTK  9,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY9
         STCM  R14,12,KEY9+4
         DESNXTK  1Ø,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY1Ø
         STCM  R14,12,KEY1Ø+4
         DESNXTK  11,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY11
         STCM  R14,12,KEY11+4
         DESNXTK  12,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY12
         STCM  R14,12,KEY12+4
         DESNXTK  13,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY13
         STCM  R14,12,KEY13+4
         DESNXTK  14,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY14
         STCM  R14,12,KEY14+4
         DESNXTK  15,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY15
         STCM  R14,12,KEY15+4
         DESNXTK  16,R6,R7,R8,R9,R11,R14,R1Ø,R15,R2
         STCM  R11,15,KEY16
         STCM  R14,12,KEY16+4
         LM    R14,R12,12(R13)      RESTORE REGISTERS
         BR    R14
         LTORG
PARAM    DSECT
KEY1     DS    CL6
KEY2     DS    CL6
KEY3     DS    CL6
KEY4     DS    CL6
KEY5     DS    CL6
KEY6     DS    CL6
KEY7     DS    CL6
KEY8     DS    CL6
KEY9     DS    CL6
KEY1Ø    DS    CL6
KEY11    DS    CL6
KEY12    DS    CL6
KEY13    DS    CL6
KEY14    DS    CL6
KEY15    DS    CL6
KEY16    DS    CL6
CLE      DS    CL8
         END
```

Thierry Falissard
Senior Systems Programmer (France)

Retrieving SMS information using the SSI

INTRODUCTION

I have put together the following code as an example of how to extract active SMS configuration information using the MVS SubSystem Interface (SSI). SMS is nearly 99% Object Code Only (OCO) so the calls to SMS and the mapping of data returned was very much a trial and error effort.

I have created two examples of code used to get information from SMS:

- LSGSMS00 – gets a list of ALL volume serial numbers defined to SMS and retrieves the CUA from the UCB, if it has one.
- LSGSMS01 – retrieves mgmtclas information for a specified dataset.

The starting point for writing this code was to review the following manual *MVS Using the Subsystem Interface* SC28-1789. This manual gave me the skeleton code to issue a call to the SSI.

I found the relevant sub-system function code and associated DSECT name, for SMS services, from Gilbert Saint-Flour’s Web site (http://www.members.home.net/gsf/tools/ssicodes.html). I later found the sub-function code in an SDUMP of the SMS address space.

The DSECT to map the SSOB function dependent area is called IEFSSSA and I found the source in the MODGEN library. Some comments in this DSECT allowed me to ‘guess’ my way through setting the values in this area.

The following is a list of required DSECTs:

• IEFSSOBH	– Subsystems Option Block
• IEFJSSIB	– Subsystem Identification Block
• IEFSSSA	– SSOB Function Dependent Area
• CVT	– Communications Vector Table

- IEFJESCT – Job Entry Subsystem Communication Table
- IGDVLD – Volume Definition Mapping.

I have created a ‘cut down’ version of DSECT IGDMCD. This DSECT is not supplied with OS/390, and seems to have been dropped after DFP3.3.

A useful source of information is a dump of the SMS address space. IPCS provides a few facilities to display SMS configuration and SSI information.

Using ‘IPCS SSIDATA’ from the IPCS dialogue command line produces a summary of the subsystem interface. There should be an entry for SMS in this report. Associated function codes are displayed here.

Command ‘IPCS VERBX SMSDATA 'FORMAT(ALL)'’ produces a report detailing all the SMS control blocks, it also dumps formatted configuration information as well. I used the formatted information to build an IGDMCD DSECT.

RETRIEVING MANAGED VOLUME LIST

Program LSGSMS00 requests SMS to provide a list of volume serial numbers it manages. The program then writes the output to a flat file, RECFM=FB and LRECL=80.

The output file contains the following fields:

- VOLSER
- CUA (blank if off-line)
- Storage group.

SMS returns the address of the VOLSERSUCB to the MVS image you run the program from; if it’s off-line it will return a null address. LSGSMS00 checks for this and if the UCB address is binary zeroes it does not look for the CUA.

I have had to GETMAIN a piece of storage below the line to put the DCB in. This allows me to run the program in address mode 31.

I have not developed the code to check the system status fields for each

VOLSER. SMS and MVS status can be extracted from fields VLDSTSMS and VLDSTMVS for each system in the SMS ‘plex’. Add code after label ‘GET_SYSTEM_INFO_LOOP’ if you wish to look at these fields.

Program LSGSMS00 is link edited AC=0, AMODE=31, RMODE=ANY and REUS.

The program can be called from a non-APF authorized library as well. The following JCL can be used to call the program:

```
//SGCSRT  JOB (,IS),'CALUM',CLASS=A,MSGCLASS=X,
//        NOTIFY=&SYSUID
//STEP0001 EXEC PGM=LSGSMS00
//STEPLIB  DD DISP=SHR,DSN=SG.UPDATE.LOAD
//SYSUDUMP DD SYSOUT=X
//SYSPRINT DD SYSOUT=X
//ABNLIGNR DD DUMMY
```

RETRIEVING MANAGEMENT-CLASS INFORMATION

Program LSGSMS01 retrieves the associated management class settings for a specified dataset name. It then returns the results into storage provided by the calling routine. The example I give is REXX calling LSGSMS01. Here is an example of REXX code to call LSGSMS01:

```
/* REXX */
 parse upper arg dataset_name .
 if dataset_name = "" then signal exit_point

 dataset_name=strip(dataset_name,b,"'")

 mgmtclas = copies(' ',30)
 expire   = copies(' ',8)
 primdays = copies(' ',4)
 l1days   = copies(' ',4)

 address linkmvs "lsgsms01 dataset_name mgmtclas expire primdays l1days"
 if rc ¬= 0 then do
  say 'command failed rc='d2x(rc)
  signal exit_point
 end

 say 'Dataset name         :' dataset_name
 say 'SMS Management Class :' mgmtclas
 say 'Mgmtclas settings    :'
 say 'retain for' strip(expire,,Ø)'days since last used'
 say 'keep on primary DASD for' strip(primdays,,Ø)'days after last used'
 say 'keep on HSM Level 1 DASD for',
      strip(l1days,,Ø)'days after last used'

exit_point:
exit 0
```

THE results from this EXEC could be:

- Dataset name: SG.SL452A.ASM
- SMS Management Class: STANDARD
- Mgmtclas settings: retain for 400 days since last used
  - Keep on primary DASD for 8 days after last used
  - Keep on HSM Level 1 DASD for 35 days after last used.

Program LSGSMS01 calls IBM routine IGWASMS to work out the management class for the dataset name passed by the REXX code. If a mgmtclas is returned then control blocks for IEFSSREQ are built. SMS provides a return area prefixed with characters ‘IGDMCD’. I have mapped only a few fields returned in this area, but the IGDMCD DSECT could be expanded to include other fields.

Program LSGSMS01 is link edited AC=0, AMODE=31, RMODE=ANY and REUS.

The program can be called from a non-APF authorized library as well.

OTHER CONSIDERATIONS

Calls can be made to not only list specific class entries, as in LSGSMS01, but also to display all entries in the current configuration (ie all DATACLAS entries). Using the SSI means you obtain up-to-date information without having to run batch extraction jobs such as DCOLLECT to get specific pieces of information. The only hindrance is that there may not be a DSECT to map the returned area. One way round this, of course, is to use the IPCS formatted dump as a template to work out what information is stored at which location.

Caution: I have only considered getting SMS to return information. There are parameter settings that could damage your SMS configuration and even delete data.

LSGSMS00

```
LSGSMSØØ AMODE 31
LSGSMSØØ RMODE ANY
LSGSMSØØ CSECT
         DS    ØH
         B     BEGIN-LSGSMSØØ(,15)
         DC    C'LSGSMSØØ: '
         DC    C'&SYSDATE &SYSTIME '
         DS    ØH
BEGIN    EQU   *
         BAKR  14,Ø
         LR    12,15
         USING LSGSMSØØ,12
         USING WORKAREA,11
         USING IEFSSSA,7
*
START    EQU   *
         L     2,=A(WORK_AREA_LENGTH)
         STORAGE OBTAIN,LENGTH=(2)
         LR    11,1
         ST    11,GETMAIN_ADDRESS
         LA    13,SAVEAREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         MVC   EYECATCHER,=CL8'LSGSMSØØ'
         L     2,=A(SSSA_LENGTH)
         STORAGE OBTAIN,LENGTH=(2)
         LR    7,1
         ST    7,SSSA_ADDRESS
*
         L     2,=A(BELOW_DSECT_LENGTH)
         STORAGE OBTAIN,LENGTH=(2),LOC=BELOW
         LR    1Ø,1
         ST    1Ø,BELOW_STORAGE_SAVEAREA_ADDRESS
         USING BELOW_DSECT,1Ø
*
         MVC   OPEN_,OPEN#
         MVC   CLOSE_,CLOSE#
         MVC   OUTPUTDCB_,OUTPUTDCB#
*
         LA    Ø,INIT_START
         L     1,=A(INIT_END-INIT_START)
         LA    14,INIT_START
         SR    14,14
         SR    15,15
         MVCL  Ø,14
*
         LA    8,SSOB_
         USING SSOB,8
         LA    9,SSIB_
         USING SSIB,9
*
         MVC   SSOBID,=C'SSOB'
         LA    1,SSOBHSIZ
         STH   1,SSOBLEN
         ST    9,SSOBSSIB
*
         ST    7,SSOBINDV
         MVC   SSOBFUNC,=Y(SSOBSSMS)
         MVC   SSIBSSNM,=CL4'SMS'
*
         MVC   SSIBID,=C'SSIB'
         LA    1,SSIBSIZE
         STH   1,SSIBLEN
*
         MVC   SSSAID,=A(SSOBSSID)
         MVC   SSSAVER,=Y(SSOBSSVR)
         MVC   SSSASFN,=Y(SSSAACTV)
         L     1,=A(SSSA_LENGTH)
         STH   1,SSSALEN
         MVI   SSSAIFLG,SSSANAUT
         MVI   SSSA1TYP,SSSA1AVL
*
         LA    1,SSOB_
         ST    1,PARMLIST
         OI    PARMLIST,X'8Ø'
         LA    1,PARMLIST
*
         IEFSSREQ
*
         ST    15,RETURN_CODE
         LTR   15,15
         BZ    EXTRACT_INFO_START
         L     Ø,SSOBRETN
         B     ABEND_ØC1
*
EXTRACT_INFO_START EQU *
         DROP  8,9
         USING IGDVLD,8
         L     8,SSSA1PTR
         L     2,VLDPCNT
         L     3,VLDPLEN
         LA    6,VLDEF
         CLC   =C'IGDVLD',VLDPID
         BNE   ABEND_ØC1
         DROP  8
         USING VLDEF,6
```

```
*
OPEN_REPORT_FILE EQU *
         OPEN  (OUTPUTDCB_,(OUTPUT)),MODE=31,MF=(E,OPEN_)
         LA    9,OUTPUTDCB_
         LA    15,CHECK_OPEN_ROUTINE
         BALR  14,15
         LTR   15,15
         BNZ   RELEASE_INFO
*
EXTRACT_VOLUME_INFO EQU *
         MVI   PRINT_LINE,C' '
         MVC   PRINT_LINE+1(L'PRINT_LINE-1),PRINT_LINE
         LA    9,P_VOLSER
         LH    5,VLDVSLEN
         LA    8,VLDVSER
         BCTR  5,Ø
         EX    5,MOVE_ROUTINE
         LA    9,P_STORGRP
         LH    5,VLDSGLEN
         LA    8,VLDSTGRP
         BCTR  5,Ø
         EX    5,MOVE_ROUTINE
*
GET_UCB_INFO EQU *
         L     5,VLDNUCBA
         LTR   5,5
         BZ    GET_SYSTEM_INFO
         USING U_UCBOB,5
         MVC   WORKS+2(2),UCBCHAN
         LA    15,CONVERT1
         BALR  14,15
         MVC   P_UCB,WORK_VAR+4
         DROP  5
*
GET_SYSTEM_INFO EQU *
         SR    8,8
         L     9,VLDSYSLN
         D     8,SYSLENGTH
         L     5,VLDSYSOF
         AR    5,6
         USING VLDSYSDT,5
*
GET_SYSTEM_INFO_LOOP EQU *
*
*  CHECK FOLLOWING FIELDS : VLDSTSMS - SMS SYSTEM STATUS
*                           VLDSTMVS - MVS SYSTEM STATUS
*                           VLDCNSMS - CONFIRMED SMS STATUS
*
*  FIELDS REPEATED FOR EVERY SYSTEM
*
         BCT   9,GET_SYSTEM_INFO_LOOP
*
```

```
PRINT_VOLUME_INFO EQU *
         DROP  5
         PUT   OUTPUTDCB_
         MVC   Ø(L'PRINT_LINE,1),PRINT_LINE
*
GETNEXT_VOLUME_INFO EQU *
         AR    6,3
         BCT   2,EXTRACT_VOLUME_INFO
*
CLOSE_REPORT_FILE EQU *
         CLOSE OUTPUTDCB_,MODE=31,MF=(E,CLOSE_)
*
RELEASE_INFO EQU *
         SR    4,4
         ICM   4,8,SSSADAID
         L     3,SSSA1PTR
         L     2,SSSA1ALN
         STORAGE RELEASE,LENGTH=(2),ADDR=(3),SP=(4)
*
ENDIT    EQU *
         L     5,RETURN_CODE
         L     2,=A(BELOW_DSECT_LENGTH)
         L     3,BELOW_STORAGE_SAVEAREA_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         L     2,=A(SSSA_LENGTH)
         L     3,SSSA_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         L     2,=A(WORK_AREA_LENGTH)
         L     3,GETMAIN_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         LR    15,5
         PR    ,
*
CONVERT1 EQU   *
         UNPK  WORK_VAR(9),WORKS(5)
         MVZ   WORK_VAR,=XL8'ØØ'
         TR    WORK_VAR,TABLE
         XC    WORKS,WORKS
         BR    14
*
CHECK_OPEN_ROUTINE EQU *
         USING IHADCB,9
         SR    15,15
         TM    DCBOFLGS,DCBOFOPN
         BO    CHECK_OPEN_ROUTINE_END
         LA    15,8
CHECK_OPEN_ROUTINE_END EQU *
         DROP  9
         BR    14
*
TABLE    DC    C'Ø123456789ABCDEF'
```

```
*
MOVE_ROUTINE MVC Ø(Ø,9),Ø(8)
*
OUTPUTDCB# DCB DSORG=PS,LRECL=L'PRINT_LINE,                            X
               RECFM=FB,MACRF=PL,DDNAME=SYSPRINT

OUTPUTDCB#_LENGTH EQU *-OUTPUTDCB#
*
OPEN#    OPEN  (OUTPUTDCB#,(OUTPUT)),MODE=31,MF=L
OPEN#_LENGTH EQU *-OPEN#
*
CLOSE#   CLOSE OUTPUTDCB#,MODE=31,MF=L
CLOSE#_LENGTH EQU *-CLOSE#
*
ABEND_ØC1 DC D'Ø'
*
         LTORG
*
           DS  ØF
SYSLENGTH DC  A(L'VLDSSTAT)
*
SSSA_LENGTH EQU SSSALN+SSSA1LN
*
WORKAREA DSECT
EYECATCHER DS  CL8
SAVEAREA DS    18F
GETMAIN_ADDRESS DS    F
SSSA_ADDRESS DS    F
BELOW_STORAGE_SAVEAREA_ADDRESS DS F
RETURN_CODE DS F
*
OPEN_    DS    CL(OPEN#_LENGTH)
CLOSE_   DS    CL(CLOSE#_LENGTH)
PRINT_LINE DS  CL8Ø
         ORG   PRINT_LINE
P_VOLSER DS    CL6,C
P_UCB    DS    CL4,C
P_STORGRP DS   CL3Ø,C
         ORG   ,
*
INIT_START EQU *
*
PARMLIST DS    F
*
         DS    ØF
SSOB_    DS    XL(SSOBHSIZ)
         DS    ØF
SSIB_    DS    XL(SSIBSIZE)
*
INIT_END EQU   *
*
```

```
WORKS    DS    CL4,C
WORK_VAR DS    CL8,C
*
WORK_AREA_LENGTH EQU *-WORKAREA
*
BELOW_DSECT DSECT
OUTPUTDCB_ DS  CL(OUTPUTDCB#_LENGTH)
BELOW_DSECT_LENGTH EQU *-BELOW_DSECT
*
         PRINT OFF
         DCBD  DSORG=PS
U_UCBOB  DSECT
         IEFUCBOB DEVCLAS=NONE,LIST=NO
         IGDVLD
         IEFSSOBH
         IEFJSSIB
         IEFSSSA
         CVT   DSECT=YES
         IEFJESCT
         END
```

LSGSMS01

```
LSGSMSØ1 AMODE 31
LSGSMSØ1 RMODE ANY
LSGSMSØ1 CSECT
         DS    ØH
         B     BEGIN-LSGSMSØ1(,15)
         DC    C'LSGSMSØ1: '
         DC    C'&SYSDATE &SYSTIME '
         DS    ØH
BEGIN    EQU   *
         BAKR  14,Ø
         LR    12,15
         LR    1Ø,1
         USING LSGSMSØ1,12
         USING WORKAREA,11
         USING IEFSSSA,7
         L     2,=A(WORK_AREA_LENGTH)
         STORAGE OBTAIN,LENGTH=(2)
         LR    11,1
         ST    11,GETMAIN_ADDRESS
         LA    13,SAVEAREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         MVC   EYECATCHER,=CL8'LSGSMSØ1'
         L     2,=A(SSSA_LENGTH)
         STORAGE OBTAIN,LENGTH=(2)
         LR    7,1
         ST    7,SSSA_ADDRESS
*
```

```
         LM    1,4,4(1Ø)
         STM   1,4,SAVE_PARMS
         L     1Ø,Ø(,1Ø)
         LH    9,Ø(,1Ø)
         ICM   Ø,B'1111',=C'PARM'
         SR    1,1
         CH    9,=H'Ø'
         BNH   ABEND_ØC1
         CH    9,=H'45'
         BH    ABEND_ØC1
*
         MVI   DATASET_NAME,C' '
         MVC   DATASET_NAME+1(L'DATASET_NAME-1),DATASET_NAME
         XC    DATASET_NAME_LENGTH,DATASET_NAME_LENGTH
         STH   9,DATASET_NAME_LENGTH+2
         BCTR  9,Ø
         LA    4,DATASET_NAME
         LA    1Ø,2(,1Ø)
         EX    9,MOVE_CHAR
*
         LA    1,RETURN_CODE
         ST    1,ASMS_PARMS+Ø
         LA    1,REASON_CODE
         ST    1,ASMS_PARMS+4
         LA    1,PROBLEM
         ST    1,ASMS_PARMS+8
         LA    1,DATASET_NAME_LENGTH
         ST    1,ASMS_PARMS+12
         LA    1,DATASET_NAME
         ST    1,ASMS_PARMS+16
         LA    1,SMS_INFO
         ST    1,ASMS_PARMS+2Ø
         LA    1,DATASET_TYPE
         ST    1,ASMS_PARMS+24
*
         LA    1,ASMS_PARMS
         LINK  EP=IGWASMS,SF=(E,LINK_)
         SR    1,1
         ICM   1,B'ØØ1Ø',RETURN_CODE+3
         ICM   1,B'ØØØ1',REASON_CODE+3
         ICM   Ø,B'1111',=C'ASMS'
         ST    1,RETURN_CODE
         LTR   1,1
         BNZ   ENDIT
*
         ICM   1,B'ØØ11',=XL2'Ø82Ø'
         ST    1,RETURN_CODE
         CLC   =C' ',MGMTCLAS               APAR OY6Ø851
         BE    ENDIT
*
         LA    Ø,INIT_START
         L     1,=A(INIT_END-INIT_START)
         LA    14,INIT_START
         SR    14,14
         SR    15,15
         MVCL  Ø,14
*
         LA    8,SSOB_
         USING SSOB,8
         LA    9,SSIB_
         USING SSIB,9
*
         MVC   SSOBID,=C'SSOB'
         LA    1,SSOBHSIZ
         STH   1,SSOBLEN
         ST    9,SSOBSSIB
*
         ST    7,SSOBINDV
         MVC   SSOBFUNC,=Y(SSOBSSMS)
         MVC   SSIBSSNM,=CL4'SMS'
*
         MVC   SSIBID,=C'SSIB'
         LA    1,SSIBSIZE
         STH   1,SSIBLEN
*
         MVC   SSSAID,=A(SSOBSSID)
         MVC   SSSAVER,=Y(SSOBSSVR)
         MVC   SSSASFN,=Y(SSSAACTV)
         L     1,=A(SSSA_LENGTH)
         STH   1,SSSALEN
         MVI   SSSAIFLG,SSSANAUT
         MVI   SSSA1TYP,SSSA1MC
         MVC   SSSA1CNT,=F'1'
         MVC   SSSA1NAM,MGMTCLAS
*
SETUP_SCAN EQU *
         LA    Ø,C' '
         LA    1,MGMTCLAS
         LA    3,L'MGMTCLAS(,1)
*
SCAN_PARM EQU  *
         SRST  3,1
         BC    1,SCAN_PARM
         BC    4,CHECK_MGMTCLAS_LENGTH
         B     RESET_MGMTCLAS_LENGTH
*
CHECK_MGMTCLAS_LENGTH EQU *
         SR    3,1
         C     3,=F'1'
         BL    RESET_MGMTCLAS_LENGTH
         L     2,=A(L'MGMTCLAS)
         CR    3,2
         BNH   SETUP_SSI_CALL
*
RESET_MGMTCLAS_LENGTH EQU *
         L     3,=A(L'MGMTCLAS)
*
SETUP_SSI_CALL EQU *
         STH   3,SSSA1NML
         LA    1,SSOB_
         ST    1,PARMLIST
         OI    PARMLIST,X'8Ø'
         LA    1,PARMLIST
*
         IEFSSREQ
*
         ICM   Ø,B'1111',=C'SSSA'
         L     1,SSOBRETN
         LTR   15,15
         BNZ   ABEND_ØC1
         ST    15,RETURN_CODE
*
         DROP  8,9
*
RETRIEVE_INFO EQU *
         USING IGDMCD,8
         L     8,SSSA1PTR
         ICM   Ø,B'1111',=C'MCD '
         SR    1,1
         CLC   =C'IGDMCD',MCDPID
         BNE   ABEND_ØC1
*
         MVC   WORKS,MCDEXPDY
         LA    15,CONVERT
         BALR  14,15
         MVC   TEMP_EXPIRE,WORK_VAR
         XC    WORK_VAR,WORK_VAR
*
         MVC   WORKS+2(2),MCDPRDY
         LA    15,CONVERT
         BALR  14,15
         MVC   TEMP_PRIDAY,WORK_VAR+4
         XC    WORK_VAR,WORK_VAR
*
         MVC   WORKS+2(2),MCDL1DY
         LA    15,CONVERT
         BALR  14,15
         MVC   TEMP_ML1DAY,WORK_VAR+4
*
SAVE_DATA EQU  *
         L     1,SAVE_PARMS
         MVC   2(L'MGMTCLAS,1),MGMTCLAS
         L     1,SAVE_PARMS+4
         MVC   2(L'TEMP_EXPIRE,1),TEMP_EXPIRE
         L     1,SAVE_PARMS+8
         MVC   2(L'TEMP_PRIDAY,1),TEMP_PRIDAY
         L     1,SAVE_PARMS+12
         MVC   2(L'TEMP_ML1DAY,1),TEMP_ML1DAY
*
RELEASE_INFO EQU *
         SR    4,4
         ICM   4,8,SSSADAID
         L     3,SSSA1PTR
         L     2,SSSA1ALN
         STORAGE RELEASE,LENGTH=(2),ADDR=(3),SP=(4)
*
ENDIT    EQU *
         L     5,RETURN_CODE
         L     2,=A(SSSA_LENGTH)
         L     3,SSSA_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         L     2,=A(WORK_AREA_LENGTH)
         L     3,GETMAIN_ADDRESS
         STORAGE RELEASE,LENGTH=(2),ADDR=(3)
         LR    15,5
         PR    ,
*
CONVERT  EQU   *
         L     1,WORKS
         CVD   1,DOUBLE_WORD
         UNPK  WORK_VAR(9),WORD_1(5)
         OI    WORK_VAR+L'WORK_VAR-1,ZONEIT
         MVI   WORK_VAR+L'WORK_VAR-1,SPACE
         XC    WORKS,WORKS
         BR    14
*
CONVERT1 EQU   *
         UNPK  WORK_VAR(9),WORKS(5)
         MVZ   WORK_VAR,=XL8'ØØ'
         TR    WORK_VAR,TABLE
         XC    WORKS,WORKS
         BR    14
*
TABLE    DC    C'Ø123456789ABCDEF'
SPACE    EQU   C' '
ZONEIT   EQU   C'Ø'
*
MOVE_CHAR MVC Ø(Ø,4),Ø(1Ø)
*
ABEND_ØC1 DC D'Ø'
*
         LTORG
*
```

```
SSSA_LENGTH EQU SSSALN+SSSA1LN+L'SSSA1NML+L'SSSA1NAM
*
WORKAREA DSECT
SAVEAREA DS    18F
EYECATCHER DS  CL8
SAVE_PARMS DS  4F
GETMAIN_ADDRESS DS    F
SSSA_ADDRESS DS    F
RETURN_CODE DS F
REASON_CODE DS F
PROBLEM  DS   2F
DATASET_NAME_LENGTH DS F
DATASET_NAME DS CL44
SMS_INFO DS   3CL3Ø
         ORG  SMS_INFO
STORCLAS DS   CL3Ø
MGMTCLAS DS   CL3Ø
DATACLAS DS   CL3Ø
         ORG  ,
DATASET_TYPE DS F
*
LINK_    LINK  EP=,SF=L
*
ASMS_PARMS  DS 7F
*
TEMP_EXPIRE DS CL8
TEMP_ML1DAY DS CL4
TEMP_PRIDAY DS CL4
*
INIT_START EQU *
*
PARMLIST DS    F
*
         DS    ØF
SSOB_    DS    XL(SSOBHSIZ)
         DS    ØF
SSIB_    DS    XL(SSIBSIZE)
*
INIT_END EQU   *
*
         DS    ØF
WORKS    DS    CL4,C
WORK_VAR DS    CL8,C
DOUBLE_WORD DS ØD
WORD_2   DS    F
WORD_1   DS    F,F
*
WORK_AREA_LENGTH EQU *-WORKAREA
         PRINT OFF
         COPY  IGDMCD
         IEFSSOBH
         IEFJSSIB
         IEFSSSA
         CVT   DSECT=YES
         IEFJESCT
         END
```

IGDMCD

```
IGDMCD   DSECT
MCD      DS    ØC
MCDP     DS    CL24                      MANAGEMENT CLASS PREFIX
         ORG   MCD+Ø
MCDPID   DS    CL8                       MC ID = 'IGDMCD'
         DS    CL2                       UNUSED
MCDPSVER DS    H                         VERSION OF MACRO
MCDPCNT  DS    F                         NUMBER OF MC DEFS COUNT
MCDPTYP  DS    H                         TYPE OF ITEMS IN LIST
         DS    CL2                       RESERVED
MCDPLEN  DS    F                         LENGTH OF ONE DEF MCDEF
*
************************************************************************
*        MANAGEMENT CLASSES                                            *
************************************************************************
*
MCDEF    DS    ØC        DIMENSION=(*)  ARRAY OF MGMTCLAS DEFS
         ORG   MCD+24
MCDNM    DS    CL32                      SPACE FOR NAME AND LENGTH
         ORG   MCD+24
MCDNMLEN DS    H                         RESERVED (WOULD BE NAME LEN)
MCDFNAME DS    CL3Ø                      MANAGEMENT CLASS NAME
MCDFUSER DS    CL8                       USERID OF LAST UPDATER
MCDFDATE DS    CL1Ø                      DATE LAST UPDATED
         DS    CL6                       RESERVED
MCDFTIME DS    CL8                       TIME LAST UPDATED
MCDFDESC DS    CL12Ø                     DESCRIPTION OF MANAGEMENT CLASS
         ORG   MCDEF+19Ø
MCDPRDY  DS    AL2                       DAYS ON PRIMARY
         ORG   MCDEF+192
MCDL1DY  DS    AL2                       DAYS ON LEVEL 1
         ORG   MCDEF+2Ø8
MCDEXPDY DS    AL4                       EXPIRE AFTER DAYS NON-USAGE
         ORG   ,
*
```

Calum Reid
Senior Systems Technician

Comprehensive compression

INTRODUCTION

One of the impediments of maintaining partitioned datasets is the necessity for compression. The higher the number of updates to a PDS, the more space will be left unutilized. Speaking of compression, let us take a brief look at what is happening when we write into a PDS. Although this has been discussed in many articles and books, it is relevant to discuss this first before we step into our 'comprehensive compression' program.

Whenever you save a member in a partitioned dataset, the system writes the entire member at the end of dataset and leaves the old version of the member in between. The directory entry for the corresponding member would be pointing to the new version. Normally when you edit a dataset, you would be saving when you come out of the dataset, but think about a user who works without a UPS (uninterruptible power supply); he would enter the 'SAVE' command as frequently as possible so that he does not lose what he typed. This would be the case with Edit.

Let's look at scenario two. Think about a load module that is updated by several development users frequently. A very good example would be a CICS RPL load library, COBOL load libraries, and other load libraries where members would be updated each time that users compile and link-edit. So there has to be a utility to compress PDS regularly to stay away from space abends. The compression process moves up the fragmented members and consolidates the free space at the end of PDS.

There are many utilities available in the market to compress, but there is no such thing as a free lunch. You have two choices – pay IBM and get DFDSS, or pay a third-party vendor and get the software. There's one more alternative for compression called IEBCOPY. IEBCOPY can be used to compress datasets-in-place.

When there is a need to compress all the datasets in a volume, it is not an easy job to code JCL for IEBCOPY. DFDSS does not allow datasets to be compressed in shared access (compressing datasets in shared access is an unavoidable requirement, especially with system datasets). For example, if there is a need to compress a test version of SIGYCOMP or any link-listed datasets, DFDSS would not allow you to compress. DFDSS would fail since it couldn't allocate the dataset in exclusive mode, even though it is a test version and on a different volume. Remember that GRS does not differentiate between datasets on different volumes for its serialization mechanism. GRS just relies on SYSDSN, dataset name, and scope for serialization. (To be clearer, link-listed datasets would be allocated by LLA in shared mode, but DFDSS would wait indefinitely, even though it needs the same dataset name in a different volume.) Moreover, DFDSS abends with 913 if there is no update access for a dataset.

When selecting IEBCOPY as the key resource for compressing datasets in a volume, there are two choices. One is to construct JCL with two DDs for each dataset on a volume and IEBCOPY control cards to compress-in-place. For example, let us assume a volume has 300 datasets of which 250 are partitioned datasets. We need to construct a JCL with 500 DDs (250 in, 250 out). We get to construct IEBCOPY control cards for each dataset. Eventually the total number of lines in the JCL would mount to approximately 750 lines. As the number of datasets on the pack increases, the number of lines in the JCL multiplies dramatically, but a simple REXX and a JCL would accomplish this task. The REXX program and JCL to perform this are shown below. You could exclude a few datasets by quoting that in the exclude list. Compressing in this way has its limitations, because the number of DDs allowed in a job step is around 3,000.

COMPRESSION METHOD 2

Now let's consider the 'comprehensive compress' program. The heart of this process is an Assembler module which gets a dataset list from a VTOC list (IEHLIST) and calls IEBCOPY to compress the dataset. It checks for authorization before passing it on to IEBCOPY. It skips the dataset if the user who is running compress utility doesn't have update/control/alter access for the dataset in question. Since it allocates and de-allocates datasets by changing JFCB, it follows the MVS convention by serializing datasets using ENQ/DEQ macros. Furthermore, it allows the user to compress a dataset in shared or exclusive mode. This can be passed as a parameter to the program when called in JCL. For each and every dataset, it creates a complete report under PRGPRINT sysout DD statement. Unlike other utilities, it wouldn't abend because of security violation, since it checks authorization with a RACROUTE macro.

It uses RDJFCB to read JFCB information and alters it with different datasets supplied with a VTOC list. The Assembler source and JCL for compression method 2 are shown below.

The first method does not produce any report other than IEBCOPY sysprint messages. It would have a hard time understanding the messages it produces for 250 or more datasets, for example. It is vulnerable for 913 abends if the user has no access for the resource. Another road block would be the number of lines in the JCL when compared with the second method.

Here's a brief description of how to use both methods:

- Method 1 – change VOLSER in three places, indicate in SYSPROC where REXX EXEC will be stored, and submit the job. It submits another job through an internal reader for compression.
- Method 2 – change VOLSER in four places, indicate in STEPLIB where load module will be stored, and submit the job. Look out for PRGPRINT DD for a detailed report.

Note:

- If you're sure that no-one updates datasets in the volume on which you are going to do compression and your site doesn't allow you to ENQ with SYSDSN qname, I would suggest you get rid of both ENQ and DEQ because this subroutine has been commented out in the source.
- Assembler source has to be link-edited with AC=1 and should be placed in an APF authorized library (calling IEBCOPY requires this authorization).

As a word of warning, it's good practice to have a good back-up before doing any sort of compression.

JCL

```
//IEBCTS JOB (ACCT),'COMPRESS',
//    REGION=4M,MSGCLASS=X,CLASS=H,NOTIFY=&SYSUID
//*** VOLSER CHANGES IN ALL 4 PLACES SHOULD BE UNIQUE
//VTOCILST EXEC PGM=IEHLIST
//SYSPRINT DD DSN=&&VTOCLST,DISP=(,PASS),
//            UNIT=DISK,
//            SPACE=(TRK,(4,1)),
//            DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6Ø5Ø)
//CATPACK  DD UNIT=DISK,DISP=SHR,VOL=SER=XXXXXX         <=== VOLSER
//SYSIN    DD *
 LISTVTOC VOL=DISK=XXXXXX
/*                                                      <=== VOLSER
//COMASM EXEC PGM=IEBCPVOL
//STEPLIB DD DSN=LOAD.DSNAME,DISP=SHR
//PRGPRINT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//CPUT1    DD UNIT=339Ø,VOL=SER=XXXXXX,DISP=SHR         <=== VOLSER
//CPUT2    DD UNIT=339Ø,VOL=SER=XXXXXX,DISP=SHR         <=== VOLSER
//VTOCDD   DD DSN=&&VTOCLST,DISP=(SHR,DELETE)
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD DUMMY
/*
```

IEBCPVOL

```
IEBCPVOL CSECT
IEBCPVOL AMODE 24
IEBCPVOL RMODE 24
*******************************************************************
* COMPREHENSIVE COMPRESS !
* NEEDS VTOC LIST AS AN INPUT
* COMPRESS ALL THE PARTITIONED DATASETS FOUND IN THE LIST
* CHECKS FOR RACF AUTHORIZATION, SKIPS IF NOT AUTHORIZED
* (NO MORE 913!)
* ACCEPTS PARAMETER TO ENQUEUE DATASETS
*   'S' ALLOWS TO COMPRESS DATASETS IN SHARED ACCESS
*   'E' ALLOWS TO COMPRESS DATASETS IN EXCLUSIVE ACCESS
* PRODUCES A COMPLETE REPORT UNDER PRGPRINT SYSOUT DD
*******************************************************************
         SAVE (14,12)
         BALR  R12,Ø
         USING *,R12
         LA    R2,SAVEAREA
         ST    R2,8(,R13)
         ST    R13,SAVEAREA+4
         LR    R13,R2
*******************************************************************
```

```
USEFCB1  USING INFMJFCB,R11
         LA    R11,JFCB1
USEFCB2  USING INFMJFCB,R1Ø
         LA    R1Ø,JFCB2
*******************************************************************
** MAIN LOOP
*******************************************************************
         LR    R4,R1
         BAL   R6,OPENIN                      OPEN VTOC LIST FILE
         LR    R1,R4
         BAL   R6,PARMCHK
         BAL   R6,READFCB                     READ JFCB
LOOPIT   BAL   R6,NEXTDS                      READ VTOC FILE
         CLC   DSORGN(11),=C'PARTITIONED'
         BNZ   LOOPIT
         BAL   R6,CHKAUTH
         LTR   R4,R4                  CHECK IF DS IS AUTHORIZED
         BNZ   LOOPIT
*        BAL   R6,GRSENQ
         BAL   R6,UPDTJFCB
         BAL   R6,CALLIEBC
*        BAL   R6,GRSDEQ
         BAL   R6,FPOOL
         B     LOOPIT
*******************************************************************
** MAIN LOOP ENDS
*******************************************************************
CLOSALL  DS    ØH
         CLOSE (INDCB)
         CLOSE (OUTDCB)
         CLOSE (DS1DCB)
         CLOSE (DS2DCB)
         L     R13,SAVEAREA+4
         RETURN (14,12),RC=Ø
*********************************************************************
** MAIN PROG ENDS
*********************************************************************
*** SUBROUTINES
*********************************
*** OPEN INPUT/OUTPUT FILES******
*********************************
OPENIN   DS    ØH
         OPEN (INDCB,INPUT)               OPEN VTOC FILE
         LTR   R15,R15
         BNZ   CLOSALL
         OPEN (OUTDCB,OUTPUT)             OPEN PRGPRINT FILE
         LTR   R15,R15
         BNZ   CLOSALL
         BR    R6
*********************************************************************
```

```
*** READ JFCB FOR CPUT1 AND CPUT2
*********************************
READFCB  DS    ØH
         RDJFCB DS1DCB
         LTR   R15,R15
         BNZ   NODD
         RDJFCB DS2DCB
         LTR   R15,R15
         BNZ   NODD
         BR    R6
NODD     DS    ØH
         MVC   OUTREC(L'MSGDD),MSGDD
         PUT   OUTDCB,OUTREC
         B     CLOSALL
*********************************************************************
*** GET NEXT RECORD FROM VTOC LIST
**********************************
NEXTDS   DS    ØH
         GET   INDCB,INREC
         BR    R6
*********************************************************************
*** UPDATE JFCB WITH NEXT DSNAME
********************************
UPDTJFCB DS    ØH
         MVC   USEFCB1.JFCBDSNM(44),DSNAME
         MVC   USEFCB2.JFCBDSNM(44),DSNAME
         OPEN  (DS1DCB,INPUT),TYPE=J
         OPEN  (DS2DCB,INPUT),TYPE=J
         CLOSE (DS1DCB)
         CLOSE (DS2DCB)
         BR    R6
*********************************************************************
*** ISSUE FREEPOOL FOR BOTH DCBS
********************************
FPOOL    DS    ØH
         FREEPOOL DS1DCB
         FREEPOOL DS2DCB
         BR    R6
**********************************************************************
*** CHECK RACF AUTHORIZATION FOR DSNAME
***************************************
*  RETURNS R4
*  R4 - Ø    INDICATES DATASET IS AUTHORIZED FOR UPDATE/CONTROL/ALTER
*  R4 - 4    INDICATES NO ACCESS TO DATASET
*  R4 - 8    INDICATES RACROUTE FAILED
**********************************************************************
CHKAUTH  DS    ØH
         LA    R4,DSNAME
         RACROUTE REQUEST=AUTH,ENTITY=((R4)),MF=(E,RACRT)
*        L     R3,RACRT
*        L     R7,RACRT+4
         LTR   R15,R15
         BNZ   AUTHFAIL
         L     R4,RACRT
         C     R4,=XL4'ØØ14'
         BNE   AUTHFAIL
         L     R4,RACRT+4
         C     R4,=XL4'ØØØ8'
         BE    AUTHOK
         C     R4,=XL4'ØØØC'
         BE    AUTHOK
         C     R4,=XL4'ØØ1Ø'
         BE    AUTHOK
         MVC   MSGDSNM,DSNAME
         MVC   OUTREC(L'MSGNAUTH),MSGNAUTH
         PUT   OUTDCB,OUTREC
         LA    R4,4                   NOT AUTHORIZED - R4 TO 4
         BR    R6
AUTHOK   LA    R4,Ø                   AUTHORIZED     - R4 TO Ø
         BR    R6
AUTHFAIL MVC   OUTREC(L'MSGRACFF),MSGRACFF
         PUT   OUTDCB,OUTREC
         LA    R4,8                   RACROUTE FAILED- R4 TO 8
         BR    R6
*********************************************************************
*** ISSUE ENQ REQUEST FOR DSNAME
********************************
GRSENQ   DS    ØH
         MVC   RNAMES,DSNAME
         L     R2,DSNLEN
         CLC   ENQFLAG,=C'E'
         BE    EXENQ
         ENQ   (QNAME,RNAMES,S,(R2),SYSTEMS)
         BR    R6
EXENQ    DS    ØH
         ENQ   (QNAME,RNAMES,E,(R2),SYSTEMS)
         BR    R6
*********************************************************************
*** ISSUE DEQ REQUEST FOR DSNAME
********************************
GRSDEQ   DS    ØH
         MVC   RNAMES,DSNAME
         L     R2,DSNLEN
         DEQ   (QNAME,RNAMES,(R2),SYSTEMS)
         BR    R6
*********************************************************************
*** CALL IEBCOPY FOR COMPRESSION
********************************
CALLIEBC DS    ØH
         LINK  EP=IEBCOPY,PARAM=(OPTLIST,DDNMELST),VL=1
         LTR   R15,R15
         BZ    COMPOK
         MVC   MSGDSNØ4,DSNAME
         MVC   OUTREC(L'OUTRECSP),OUTRECSP
         MVC   OUTREC(L'MSGCMPØ4),MSGCMPØ4
         PUT   OUTDCB,OUTREC
         BR    R6
COMPOK   DS    ØH
         MVC   MSGDSNØØ,DSNAME
         MVC   OUTREC(L'OUTRECSP),OUTRECSP
         MVC   OUTREC(L'MSGCMPØØ),MSGCMPØØ
         PUT   OUTDCB,OUTREC
         BR    R6
*********************************************************************
*** CHECK FOR PARAMETERS
************************
PARMCHK  DS    ØH
         L    R1,Ø(R1)
         LA   R2,Ø
         LH   R2,Ø(R1)
         LTR  R2,R2
         BZ   DEFENQ
         LA   R1,2(R1)
         CLC  Ø(1,R1),=C'S'
         BE   DEFENQ
         CLC  Ø(1,R1),=C'E'
         BNE  INVPARM
         MVI  ENQFLAG,C'E'
         BR   R6
DEFENQ   DS    ØH
         MVI  ENQFLAG,C'S'
         BR   R6
INVPARM  DS   ØH
         MVC  OUTREC(L'OUTRECSP),OUTRECSP
         MVC  OUTREC(L'MSGERPRM),MSGERPRM
         PUT  OUTDCB,OUTREC
         B    CLOSALL
*PARMCHK ENDS
*********************************************************************
** SUBROUTINES END
*********************************************************************
         YREGS
SAVEAREA DS    18F
*
ENQFLAG  DS    C
*
MSGRACFF DC    CL5Ø'RACROUTE FAILED WHEN CHECKING AUTH'
MSGNAUTH DS    ØCL1ØØ
         DC    CL56'COMPRESS SKIPPED. UPDATE AUTH. REQD FOR'
MSGDSNM  DS    CL44
```

```
MSGDD    DC    CL7Ø'CODE CPUT1/CPUT2 DD STATEMENTS !'
*
MSGCMPØØ DS    ØCL1ØØ
         DC    CL56'COMPRESS SUCCESSFUL FOR '
MSGDSNØØ DS    CL44
*
MSGCMPØ4 DS    ØCL1ØØ
         DC    CL56'CHECK IEBCOPY MESSAGE, NON-ZERO RC FOR'
MSGDSNØ4 DS    CL44
*
MSGERPRM DC    CL1ØØ'INVALID PARAMETER (VALID: S/E - SHARED/EXCLUSIVE)'
*
*SETUP PARM LIST FOR IEBCOPY
OPTLIST  DC  H'Ø'
DDNMELST DC  AL2(L'DDNMEND)
DDNMPARM DC  7XL8'Ø'
         DC  CL8'CPUT1   '
         DC  CL8'CPUT2   '
DDNMEND  EQU DDNMPARM,*-DDNMPARM
*
INREC    DS    ØCL121
DSNAME   DS    CL44
         DS    CL2Ø
DSORGN   DS    CL11
         DS    CL46
*
* DEFINITIONS FOR ENQ/DEQ
QNAME    DC    CL8'SYSDSN'
RNAMES   DS    CL44
DSNLEN   DC    F'44'
*
OUTREC   DS    CL121
*
OUTRECSP DS    121CL1' '
*
WAREA    DC      A(KBY2)
KBY2     DS      CL512
* JFCB1 LIST
JFCB1     DS    44F
JFCBPTR1  DC    X'87'
         DC    AL3(JFCB1)
* JFCB2 LIST
JFCB2     DS    44F
JFCBPTR2  DC    X'87'
         DC    AL3(JFCB2)
*
* DCB DEFINITIONS
DS1DCB   DCB   DSORG=PS,MACRF=E,EXLST=JFCBPTR1,EODAD=CLOSALL,         X
               DDNAME=CPUT1,BUFL=3276Ø
DS2DCB   DCB   DSORG=PS,MACRF=E,EXLST=JFCBPTR2,EODAD=CLOSALL,         X
```

```
               DDNAME=CPUT2,BUFL=3276Ø
INDCB    DCB   DSORG=PS,MACRF=GM,EODAD=CLOSALL,                        X
               DDNAME=VTOCDD,LRECL=121,RECFM=FB
OUTDCB   DCB   DSORG=PS,MACRF=PM,EODAD=CLOSALL,                        X
               DDNAME=PRGPRINT,LRECL=121
*
* RACROUTE LIST FORM
RACRT    RACROUTE REQUEST=AUTH,CLASS='DATASET',STATUS=ACCESS,          X
               DSTYPE=M,WORKA=WAREA,RELEASE=1.9,MF=L
* MAPPING MACRO FOR JFCB READ BY RDJFCB
         IEFJFCBN
         END
```

GENCOMP JOB

```
//GENCOMP JOB  (ACCT),'IEBC JOB',CLASS=H,
//         MSGCLASS=X,NOTIFY=&SYSUID,REGION=ØM
//****************************************************************
//************ GENERATES IEBCOPY COMPRESS JCL
//****************************************************************
//************ VOLSER CHANGES SHOULD BE UNIQUE IN ALL THE PLACES
//OUT1 OUTPUT JESDS=ALL,DEST=LOCAL,CLASS=T,DEFAULT=YES
//OUT2 OUTPUT JESDS=ALL,DEST=LOCAL,CLASS=J,DEFAULT=YES
//VTOCILST EXEC PGM=IEHLIST
//SYSPRINT DD DSN=&&VTOCLST,DISP=(,PASS),
//      UNIT=DISK,
//         SPACE=(TRK,(4,1)),
//         DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6Ø5Ø)
//CATPACK  DD UNIT=DISK,DISP=SHR,VOL=SER=XXXXXX          <==== VOLSER
//SYSIN    DD *
 LISTVTOC VOL=DISK=XXXXXX                                 <==== VOLSER
/*
//VTOCISRT EXEC PGM=SYNCSORT
//SYSPRINT DD SYSOUT=(,)
//SYSOUT   DD SYSOUT=(,)
//SORTIN   DD DSN=&&VTOCLST,DISP=(OLD,PASS)
//SORTOUT  DD DSN=&&PDSLST,DISP=(,PASS),
//            SPACE=(TRK,(1,1)),
//            DCB=(RECFM=FB,LRECL=121,BLKSIZE=6Ø5Ø)
//SORTWKØ1 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SORTWKØ2 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SORTWKØ3 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SORTWKØ4 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SORTWKØ5 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SORTWKØ6 DD UNIT=DISK,SPACE=(CYL,(15,5))
//SYSIN    DD *
 SORT  FIELDS=(2,44,CH,A)
 INCLUDE COND=((114,4,CH,EQ,C'NONE'),AND,(66,11,CH,EQ,C'PARTITIONED'))
/*
```

```
//IEBCPY  EXEC PGM=IKJEFTØ1
//SYSPROC DD DSN=AAAA.BBBB.CCC,DISP=SHR              <====== REXX
PROGRAM DSN
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 GENCPCMP
/*
//VTOCDD  DD DSN=&&PDSLST,DISP=SHR
//VOLINFO DD *
XXXXXX                 <====== VOLSER
/*
//IEBCDD  DD SYSOUT=(B,INTRDR)
//EXCLDD  DD *
exclude ds list
/*
```

REXX EXEC

```
/*REXX*/
"execio * diskr vtocdd (stem vtocarr. finis"
if rc¬=Ø then do
   say 'Error opening Vtoc list'
   exit 2Ø
end
"execio * diskr excldd (stem exclarr. finis"
if rc¬=Ø then do
   say 'Error opening Exclude list'
   exit 2Ø
end
"execio * diskr volinfo (stem tempvol. finis"
if rc¬=Ø then do
   say 'Error opening Vol information'
   exit 2Ø
end
vtocvol=strip(substr(tempvol.1,1,6))
"newstack"
call conscards
call varcards
"execio "queued()" diskw iebcdd ( finis"
if rc¬=Ø then do
   say 'Error writing to IEBC JCL'
   "delstack"
   exit 2Ø
end
"delstack"
return
/**end of main prog***/
```

```
/************************************************************/
conscards:
queue "//IEBC"||substr(vtocvol,1,3)||,
    " JOB  (ACCT),'IEBC JOB',CLASS=H,"
queue "//         MSGCLASS=T,NOTIFY=&SYSUID,REGION=0M"
queue "//IEBCPY  EXEC PGM=IEBCOPY"
queue "//SYSPRINT DD  SYSOUT=*"
queue "//SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(1))"
queue "//SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(1))"
return
/************************************************************/
varcards:
ddcnt=0
do i=1 to vtocarr.0
   exclflag=0
   tvtoc=strip(substr(vtocarr.i,2,44))
   do j=1 to exclarr.0
      texcl=strip(substr(exclarr.j,1,44))
      if texcl=tvtoc then do
         exclflag=1
         leave
      end
   end
   if exclflag=1 then iterate
   ddcnt=ddcnt+1
   str='//IN'||ddcnt||' DD UNIT=DISK,DSN='||tvtoc||','
   queue str
   str='//      VOL=SER='||vtocvol||',DISP=SHR'
   queue str
   str='//OUT'||ddcnt||' DD UNIT=DISK,DSN='||tvtoc||','
   queue str
   str='//      VOL=SER='||vtocvol||',DISP=SHR'
   queue str
end
queue '//SYSIN DD *'
do i=1 to ddcnt
   str='  COPY OUTDD=OUT'||i||',INDD=IN'||i
   queue str
end
queue '/*'
return
/************************************************************/
```

Sridhar Nelliyappan Manivel
MVS Systems Programmer
R Systems Inc (USA)

Advanced PDF line commands

INTRODUCTION

In edit mode, you can enter line commands in the left margin (the line number area). You already know the most common ones such as ‘C’opy, ‘A’fter, ‘B’efore, ‘M’ove, ‘D’elete, ‘I’nsert, ‘R’epeat, etc and their block equivalents such as ‘CC’ and ‘DD’. We have considered some of the other elements such as: ‘O’verlay, ‘)’ shift right, ‘(’ shift left, as well as ‘TS’ text split, and ‘TF’ text flow. Here are a few other less common tools.

X

‘X’ will temporarily hide (or e‘X’clude) a line. ‘XX’ is the group equivalent. This is handy to get out of your way some intervening lines so as to get two other groups of lines closer together, perhaps on the same screen. The lines are replaced with a single line of dashes. You can enter the following commands in the left margin of that line (if ‘n’ is not specified, it defaults to ‘1’):

- Fn – will redisplay the first ‘n’ lines of excluded text.
- Ln – redisplays the last ‘n’ lines.
- Sn – redisplays ‘n’ lines with the leftmost indentation in a block of excluded lines. For example:

```
ØØ1ØØ    PERFORM 91Ø-READ-NEXT-RECORD.
XX2ØØ    IF X = '2'
ØØ3ØØ      MOVE A  TO B
ØØ4ØØ    ELSE
XX5ØØ      MOVE C  TO B.
00600    PERFORM 900-WRITE-A-LINE.
```

If we press <Enter>:

```
ØØ1ØØ    PERFORM 91Ø-READ-NEXT-RECORD.
—— ———————————————-
ØØ6ØØ    PERFORM 900-WRITE-A-LINE.
```

Then we show the left-most lines

```
ØØ1ØØ    PERFORM 91Ø-READ-NEXT-RECORD.
S3– ———————————————-
00600    PERFORM 900-WRITE-A-LINE.
```

After we press <Enter>:

```
ØØ1ØØ    PERFORM 91Ø-READ-NEXT-RECORD.
ØØ2ØØ    IF X = '2'
—— ———————————————-
ØØ4ØØ    ELSE
—— ———————————————-
ØØ6ØØ    PERFORM 9ØØ-WRITE-A-LINE.
```

UC/LC

‘UC ’ will convert all the characters on a line to upper case. ‘UCC’ and ‘UCUC’ are the group commands. ‘LC’, ‘LCLC’, and ‘LCC’ are the ‘lower-case’ equivalents.

TE

‘TE’ inserts a page full of blank lines for power typing when <Enter> is pressed. The lines precede the line where the ‘TE’ is entered. This is usually used for text entry. When the next <Enter> is pressed, any unused lines are removed. If you run out of lines, pressing ‘page down’ (usually F8) will give you a new page full of empty lines.

Alan Kalar
Systems Programmer (USA)

In-line performs

THE PROBLEM

Suppose we wish to step through a working-storage table for the purpose of doing something simple for each row of the table. Normally, we would PERFORM another paragraph using the VARYING verb to step the subscript through the table. But, in this case, the task is so simple we don't want to waste a paragraph on it. The program would be easier to understand if we were able to include the few lines within the controlling code.

THE SOLUTION

The gurus of COBOL have finally allowed us to do just that. It is very easy. Here is an example that should make everything clear:

```
1   IF START-OF-NEW-BATCH
2       PERFORM VARYING X FROM 1 BY 1
3                 UNTIL X > 20
4           MOVE SPACES       TO  TABLE-ITEM (X)
5           PERFORM VARYING Y FROM 1 BY 1
6                     UNTIL Y > 100
7               MOVE SPACES   TO TABLE-DETAIL (X, Y)
8               PERFORM 100-CLEAR-ASSOCIATED-DATA
9               END-PERFORM          ** very important **
10          END-PERFORM
11      END-PERFORM
12  ELSE
13      MOVE INPUT-VAR        TO TABLE-ITEM (X)
14      PERFORM VARYING Y FROM 1 BY 1
15                UNTIL Y > 100
16          MOVE INPUT-D (Y)  TO TABLE-DETAIL (X, Y)
17      END-PERFORM
18      SET X                 UP BY 1
19  END-IF.
```

Notice the 'END-PERFORM' on line 9. Even though line 8 is not an in-line Perform, we still need the scope delimiter so that the compiler knows that the next scope delimiter is for the in-line perform on line 5. It's like matching up IF/ELSE/END-IF sets.

Alan Kalar
Systems Programmer (USA)

Disaster recovery planning issues

INTRODUCTION

This article provides a basis for planning a disaster recovery project. It considers the issues that need to be evaluated prior to starting and those which should be considered during implementation. It does not provide information regarding non-technical issues such as who should be involved in the project, although careful consideration should be given to this.

Two major projects are providing impetus for disaster recovery projects:

- Planning for the continuity of the systems for 24 x 7 to support e-commerce development.

- Security over multi-platform networks raised by the Open-Edition in the OS/390 environment.

BUSINESS ANALYSIS

The first step before starting a disaster recovery plan is to analyse the nature of your business. It is essential to establish the following elements for your business:

- What is the hardware mix in your site?

- Which operating systems are crucial for your business continuity and your application business demands?

- What is the status of your business connectivity to the world? You need to review which sites or companies are connected to your site and need to gain access to your application data or *vice versa*.

- Is your business located at a single site or spread across several sites? What is the distance between these sites?

- What is your most important mission-critical business application process? This will be one of the most important issues in your disaster recovery action plan.

- After studying your business you should consider your geographical area (is the area of your site prone to any particular phenomena such as tornados, earthquakes, or rivers liable to flood?).

TYPES OF DISASTER RECOVERY PLAN

Your site probably already has some form of disaster recovery plan, but it is necessary to consider if this plan is satisfactory. Disaster recovery plans can be divided into three major types:

- In-house recovery plan – this is the standard legacy process of ‘back-up/restore’. It is a simple solution that is only applicable for restoring data for a limited number of datasets or projects. This methodology will be of little use in the recovery from a total business crash.

- Cold site recovery – this refers to sites that have a contract with a third-party vendor. The vendor provides hardware and an operating system, and can restore the data from tapes and continue working from the time of the last back-up. This process relies on taking full or incremental back-ups daily that can be restored to a new site.

  This restore process is quite long, it could take days before a company can return to full business. Morever, there is likely to be some data loss because of the probable time difference from the back-up.

- Hot site recovery – this refers to sites that have mirrored data across multiple sites. The system continuity is almost immediate and with minimal disruption. The recovered site needs only to be switched to the outside world to which it is already connected.

LIMITATIONS

It is essential to ascertain which of the above methodologies your site has adopted, and if these can support your future management needs.

- If your site has adopted ‘in-house recovery’ it is unlikely that this will be satisfactory for future needs. With businesses exploiting e-business, it is unlikely that a total system crash will be recovered quickly enough for market requirements.

- There are two primary considerations when evaluating cold site recovery. First, the back-up window's daily process will shorten. The maintenance window as a whole is getting shorter every day. This is the reason why IBM uses the Parallel Sysplex methodology. This supports total continuity during several disaster scenarios.

  Secondly, you cannot backup every DASD in your site as 'full back-up'; normally you would backup your starter system and mission-critical data with the standard utilities provided by your vendors as 'full' and add the incremental back-up for the rest of the data. You will soon notice that the 'full' back-up process is getting larger and longer every day, gathering more DASD devices to backup, more cartridges to back-to, and more tape drives and silos to use with the back-up process, while the restore time required is also getting longer. Then again, the next time you are testing your recovery site you notice that some data that was delivered as 'non-mission-critical' becomes the most critical data. For example, if you are ready to restore your production CICS as mission-critical but are a financing firm that comes on-line only from 08:00 to 16:00 and comes to recovery process because of a system crash at 18:00, you are sure to start the recovery of your CICS data. However, your management will force you to recover your batch data first for updating the data needed for your employees for crucial reporting.)

If you already exploit hot site recovery you have probably persuaded your management of the necessity of the project. You have probably spent considerable time justifying the economics of the project. If you have not, the following will provide a brief overview.

JUSTIFYING HOT SITE RECOVERY

When justifying hot site recovery, try not to build an academic recovery plan that tries to cover as many scenarios as possible; you will never finish it. It is advisable to build the kernel of your plan and move it forward to management for approval. You can expand the plan later with future experience. Secondly, do not expect one solution for the entire organization – every one of us is different from the other in our platforms, operating systems, and applications.

Justifying the need for a recovery site to management is not easy. You will have to explain why they need to invest a great deal of money in a ‘disaster recovery plan’. Do not try to ‘talk techniques’ with them and use buzzwords like ‘remote copy’, ‘PPRC’, ‘WDM’, etc, but try to point out the closest scenarios that have happened and their consequences.

The following research has been undertaken by IBM (which can be found at the following URL: http://www1.ibmlink.ibm.com/html/spec/g1222410.html) and provided the following results. Companies in the following sectors who experience a total system crash need to be operational within the time limits show below:

- Finance – 2 days
- Commerce – 3.5 days
- Industrial – 5 days
- Insurance – 5.5 days.

Research undertaken after the 1994 earthquake in Los Angeles, the 1995 floods in the Netherlands and Germany, and the World Trade Center in New York provided the following results.

Of the companies whose systems or back-up systems were not operational within the time limits shown, over 25% went bankrupt immediately, a further 40% closed their business in the next two years, and the remainder closed down within five years.

Using the data shown above it is necessary to evaluate the impact of downtime at your site. Evaluate the consequences of any one of the divisions in your firm being non-operational. What will be the cost of not being accessible to the outside world? Do not hesitate to get the numbers and show the dollars and cents to the management, this will be the real issue of justifying the project – what is the potential risk of not having it?

DEVELOP A RECOVERY PLAN

After you have positioned your company, try drawing a time line of recovery process that you think will be suitable for your company.

If you have a cold site, put yourself in a ‘disaster happens now’ position and try to watch, step by step, the recovery process. It will probably be similar to the following:

- *The disaster strikes.* At the beginning of it you are not sure that this is ‘the disaster’ that might shut you down. The professionals are probably gathered in the computer room, trying to figure out what is happening and what might be the proper action to be taken in order to solve the problem. This step can take 2-3 precious hours that are wasted and in the end they will not give you any benefit. This step is never covered or predicted in your current recovery plans and the length of time is never forecast.
- *The management decision.* At this point your CFO comes and, using his management power, orders you to start the recovery process at the cold site you are prepared for. To pack all of your tape data, move your personnel to the new location, and start your kernel system, can still take hours or days until you are up and operating. This process is well known to your professional personnel; it has been tested many times (probably twice during the year), and all of the personnel are ready to wait until you have finished your restores or even, in some cases, until you know that there is data that cannot be recovered!
- *Recover from the recovery.* Remember that you started your recovered system with the back-up tapes from some time yesterday (end of day), but from then the data kept on coming and updating your files. Now is the time to start forwarding your data to the point of time the disaster struck. This process might take days or even weeks.

When drawing these scenarios, you begin to see how you can improve and shorten the time line.

If you have a hot site recovery plan, close any remote copy connections from the beginning of the disaster. Use any automated message capture products that you find accurate, but close the connection to avoid a rolling disaster scenario at the hot site (especially human error disaster). If you have done that, you immediately have an updated copy at the remote site accurate to the second. Let the professionals try to locate the problem ‘in house’ but be ‘ready to go’ in your new location within minutes of your CFO’s decision.

Your CFO will probably approve your immediate relocation because of your readiness.

These two points alone have already saved you hours or days of downtime.

There is not any recovery from the recovery in this scenario. The data is updated within milliseconds or seconds in the hot recovery site, so you are actually up and operating within minutes.

ASCERTAIN THE TECHNOLOGY AT HAND

Let's assume that you have finally received the management approval to start building the disaster recovery plan. The first thing you must do is learn the technology at hand. Investigate the CPU requirements to meet the objective of the recovery site. Try to figure out if you need to buy a new machine or if you can use a contract like Capacity Backup (CBU from IBM and some other vendors world-wide). Learn the number of processors you need to start your recovery system and how much real storage you will need.

Investigate the I/O ratio in your workload; try to calculate the efficiency of your channels. Ascertain whether you need to add more channels or ESCON cards to your disk control units or maybe you need to move to FICON channels (wherever possible).

Try checking at your location the need to use the wave division multiplexors from local vendors; it will cost you at first but you will save the channel spreading across the two sites. These multiplexors are fully redundant, but do not forget to use pairs in two different paths with your local fibre vendor (in case something happens to one channel because of disaster).

Choose the RAID DASD control units you want to work with which are the most suitable for your workload. Learn the methods of data mirroring using Peer to Peer Remote Copy or Extended Remote Copy by pointing out the pros and cons for every method. At the end, choose the one you find best suited to your organization.

Check the Open System environment requirements. There are DASD vendor trends to open the mainframe DASDs to the Unix, AS/400, and even MS/Windows within host client interfaces to the DASD control units. This openness is using SCSI or even Fiber Channel Arbitrated Loop cards, with or without hubs, to connect to the DASD control unit. Figure out what are the local production servers at your site and see if your vendor supports the remote copy to this platform. All there is to do is to install the appropriate platform at the hot site and switch to the remote copy as disaster strikes.

If you are using Automated Tape Libraries, you can use several of them at the remote hot site, holding a copy of your full back-up or application tapes instead or sending those copies to the vault.

Any communication devices can be relocated around several sites, installing at least one at the hot site, while you might install them in local mode connected with ESCON attached channels or with one of the multiplexors. Any other equipment such as printers should also be examined in the disaster plans.

THE BOTTOM LINE

Your IT division is central and the most valuable holding within a business. For most companies worldwide, data is their business. You must address management about the potential risk you are living with, and about using your current recovery methods, and modify these in parallel with the evolving architecture developed by the application requirements in your organization (e-business, Web connection, openness, etc) via the technical development by your hardware vendors.

You must prioritize your production application and operating systems because this will be the order that you will start recovering your new location hot site recovery.

Any investment you are forecasting should evaluate the relative cost in your recovery site. Do not hesitate to point out to management the negative public image impact of non-recovery of a disaster at your site, and if and when disaster strikes what the consequences will be if you are not ready to operate within minutes.

Oded Tagger
Systems Programmer (Israel)

Split, flow, and programming PF keys

INTRODUCTION

Split and flow were created for use with early text editing in ISPF. With the introduction of personal computers, mainframe text editors such as IBM’s Script® have faded into obscurity, but the commands remain. They can be of some use when coding free-form languages, such as COBOL and PL/I. If you develop a fondness for these or any other commands, you can create short-cuts using PF keys.

HOW THEY WORK

Let’s suppose we have created a line of code as follows:

```
....5...1Ø...15...2Ø...25...3Ø...35...4Ø...45....5Ø...55...6Ø...65...7Ø....75...
ØØØ1ØØ     MOVE SUPER-LONG-NAMED-VARIABLE TO ANOTHER-SUPER-LONG-NAMED-VARIABLE.
```

As you can see, it is too long for one line. COBOL will ignore anything after column 72. Rather than re-type or copy (or repeat) and edit the line, we can split it and realign the split-off portion . . . as follows:

```
tsØ1ØØ     MOVE SUPER-LONG-NAMED-VARIABLE TO ANOTHER-SUPER-LONG-NAMED-VARIABLE.
```

We entered ‘ts’ in the line command area. The ‘ts’ stands for ‘text split’. We also moved our cursor to the spot where we want the split to take place. In this case, we put it under the ‘T’ in ‘TO’. After you press <Enter> it will look like this:

```
....5...1Ø...15...2Ø...25...3Ø...35...4Ø...45....5Ø...55...
ØØØ1ØØ     MOVE SUPER-LONG-NAMED-VARIABLE
ØØØ2ØØ TO ANOTHER-SUPER-LONG-NAMED-VARIABLE.
```

In this case, our bounds (BNDS) are set for 8 to 80. If they were set for 8 to 72, the ‘IABLE’. would have been left where it was. (Refer to *Using overlays* in *MVS Update*, November 1999 for a discussion on the ‘BNDS’ line command.)

Now, all we have to do is shift the second line to where we want it (say 29 spaces to the right). . .

```
....5...1Ø...15...2Ø...25...3Ø...35...4Ø...45...
ØØØ1ØØ     MOVE SUPER-LONG-NAMED-VARIABLE
)292ØØ TO ANOTHER-SUPER-LONG-NAMED-VARIABLE.
```

giving:

```
....5...1Ø...15...2Ø...25...3Ø...35...4Ø...45....5Ø...55...6Ø...65...7Ø....75...
ØØØ1ØØ     MOVE SUPER-LONG-NAMED-VARIABLE
ØØØ2ØØ                                TO ANOTHER-SUPER-LONG-NAMED-VARIABLE.
```

Too much work? OK, let’s re-program a Function key to do the split. While in an edit screen, we’ll enter ‘KEYS’ on the command line:

```
     COMMAND===> KEYS
```

Next we’ll find the key we want to change. For this example we’ll use F22, so we have to page down (F8) to the next set of keys. On the line for the ‘22’ key we will enter. . .

```
     F22  . .  :ts
```

Then press F3 (end). Now, we can get the same results by just positioning the cursor where we want to split the line and pressing F22 (shift-F10). No need to enter ‘ts’ in the line command area. The ‘:’ (colon) means the command is assumed to be a line command.

Without the ‘:’, ISPF would interpret the instruction to be in the COMMAND line as in: ‘swap;=s.h;bottom’, assuming that ‘;’ (semi-colon) is our command concatenation character (which is the default). This particular command assumes a split window environment. It will swap from the current window to the other (‘swap’), go to the held output listings (‘s.h’) and page to the end of the list (‘bottom’).

Want to combine lines? ‘tf’ (text flow) line command will combine lines until a blank line or full stop (‘.’) is encountered (save your work before you try this the first time.) It starts with the line it is entered on, or the line the cursor is in if you use a re-programmed function key similar to the one we set up for the ‘split’ line command. The boundaries (BNDS) define the column limits for the operation.

The function keys in ISPF/PDF EDIT are not the same keys used elsewhere, so our changes in EDIT will not necessarily be available in other functional areas within PDF.

Alan Kalar
Systems Programmer (USA)

MVS news

IBM has announced Version 2.1 of its Payment Gateway for OS/390, part of the Payment range promising end-to-end solutions for electronic commerce over the Internet. Payment Gateway is described as an intelligent interface between merchant Web sites and existing non-Internet credit and debit card processing systems.

Specific features include the ability to help check the validity of transactions arriving from merchants, to translate these messages into formats recognized by the card processing systems, and route them for further handling.

It carries out routing to manage the authorization and acceptance of encrypted payment messages over the Internet. There's support for SET Secure Electronic Transaction Version 1.0 and Secure Socket Layer (SSL) Version 3. Prerequisites include an S/390 Parallel Enterprise Server running OS/390 Version 2 Release 7 or later, and DB2 Version 5 Release 1 or later.

* * *

IBM also announced its Program Restart Facility for IMS, designed to restart applications faster, reduce errors during restart, and reduce overhead caused by checkpointing. It allows restarts in sysplex environments and allows automatic restart operations.

Specifically, it helps ensure that the correct restart checkpoint ID is used to restart IMS applications using IMS Extended Restart Facilities.

* * *

IBM has announced Release 2 of its IMS/ESA Year 2000 Exit Tool to help address data sequencing for IMS database segments. It handles sequencing segments that use date data as part of the key field, is transparent to IMS applications, and works with IMS 5, 6, and 7.

Part of an interim means for the correct sequencing of date-keyed segments, it can help avoid changing IMS applications and database definitions to accommodate larger date fields. Specifically, it allows companies to identify IMS key field dates for special handling, convert the dates identified for proper sequencing, and intercept and translate date data when accessed by an IMS application. Date data is converted for proper segment sequencing and translated for use with IMS applications.

The tool's code connects to the IMS Data Conversion Exit, and databases using the exit must be unloaded without the exit code and reloaded with the exit. Once implemented, dates can be sequenced properly.

Developed with Telcordia (once Bellcore) the product lets IMS applications retrieve and insert date-keyed segments in the desired order. The view of the key data by the application remains unchanged, but applications needing to display four-digit year data, or to carry out calculations using dates, may still need to be changed.

For further information contact your local IBM representative.

http://www.ibm.com

* * *