186 MVS update

March 2002

In this issue

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

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

Contributions
Articles published in *MVS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, you can download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

MVS Update on-line
Code from *MVS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/mvs; you will need to supply a word from the printed issue.

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.

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 1998 issue, are available separately to subscribers for £29.00 ($43.50) each including postage.

Printed in England.

Monitoring LLA activity

PROBLEM

I have often wondered if there is an easy way to get library lookaside (LLA)-related information regarding which modules are being referenced and how often they are referenced. This could be used to verify and measure the benefit of LLA-management of libraries and determine what LLA is really doing with these libraries.

However, IBM does not seem to provide a tool to interrogate and/or monitor the status and the contents of LLA cache. Of course, there is the officially supported MVS command D LLA, which lists the LLA-managed datasets along with additional information, such as LNKLST status, freeze status of the library, remove status of the library, and whether the library is a PDSE (see *OS/390 V2R10: MVS System Commands*, GC28-1781-11).

Although this command is very useful, it does not actually supply the answer to my original question and it prompted me to look further for a tool that would provide me with a more substantial report. This report should provide information about how LLA/VLF handles the libraries so that I could adjust the LLA/VLF parameters accordingly. It should be remembered that LLA improves the performance of fetching modules from both LNKLST and non-LNKLST datasets, and that the most frequently used libraries and modules will receive LLA/VLF cacheing preference. One may be saving a lot of DASD I/O to other libraries and modules that are used more frequently than those one is immediately concerned with.

A SOLUTION

By investigating an undocumented form of the D LLA command I found that it provides only a partial answer to my initial question. I have used the D LLA,STATISTICS command since it is known that for each module that is fetched, LLA dynamically accumulates and logs statistics. What a user gets from the D LLA,STATISTICS command is just some basic statistics about DASD and VLF fetches,

but the output it produces was found to be a bit clumsy, therefore a simple REXX EXEC (LLASTAT) was written to give users a more readable report.

The EXEC is a two-part stream – in the first part the above mentioned MVS command is issued and its output is written to a file, while in the second part the captured output is formatted, totals calculated, and the result printed into a new file.

LLASTAT EXEC

```
/* REXX */
/*--------------------------------------------------------------------*/
/* Part I: issue MVS command and write the output to a file           */
/*--------------------------------------------------------------------*/
/*Trace ?r */
Address TSO
  userid=SYSVAR(SYSUID)
  cmdds  =userid||'.comds.out'          /* change dataset names    */
  outds  =userid||'.comfm.rep'          /* to fit your standards   */
  cpu_seconds = sysvar(syscpu)
  begin = time(R)
  cart   = sysvar('SYSUID')

/*--------------------------------------------------------------------*/
/*  console environment                                               */
/*--------------------------------------------------------------------*/

 x = MSG('OFF')
  SD=SYSVAR("SOLDISP")
  USD=SYSVAR("UNSDISP")
  If SD = 'YES' Then
     "CONSPROF SOLDISP(NO)"        /* Console Profile: solicited &  */
  If USD = 'YES' Then              /* unsolicited must be NO to be  */
     "CONSPROF UNSOLDISP(NO)"      /* able to catch command response*/
  "CONSPROF SOLNUM(400) UNSOLNUM(400)"

  x = MSG('ON')
  "CONSOLE ACTIVATE CART("cart")"      /* activate CONSOLE service */

  "CONSOLE SYSCMD(d lla,statistics) CART("cart")"
  getcode = GETMSG('cons_msg.','SOL',cart,,5)

  "CONSOLE DEACTIVATE"                 /* close console session    */

/*--------------------------------------------------------------------*/
/* copy the command response to output dataset                        */
```

```
/*-------------------------------------------------------------------*/
  If SD = 'YES' Then
     "CONSPROF SOLDISP("SD")"
  If USD = 'YES' Then
     "CONSPROF UNSOLDISP("USD")"
  x = MSG('OFF')

  if SYSDSN(cmdds) = 'OK'         /* check command-out DSN validity */
  Then "DELETE "cmdds" PURGE"
  if SYSDSN(outds) = 'OK'         /* check report DSN validity      */
  Then "DELETE "outds" PURGE"

  "ALLOC FILE(CMDOUT) DA("cmdds")",
     " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
     " REUSE LRECL(133) RECFM(F B) BLKSIZE(399Ø)"
  "ALLOC FILE(fout) DA("outds")",
     " UNIT(SYSALLDA) NEW TRACKS SPACE(2,1) CATALOG",
     " REUSE LRECL(8Ø) RECFM(F B) BLKSIZE(32ØØ)"

  "EXECIO * DISKW CMDOUT (stem cons_msg. FINIS"
  "FREE FI(CMDOUT)"
/*-------------------------------------------------------------------*/
/* Part II: Formatting captured output & printing it to a new file */
/*-------------------------------------------------------------------*/
  "ALLOC FILE(in)  DA("cmdds") shr reu"
  "EXECIO * DISKR in (stem row. "
/*-------------------------------------------------------------------*/
/* print header and labels                                           */
/*-------------------------------------------------------------------*/
 Out.1 = left(' ',2Ø,' '),
         ||center('Display LLA statistics ',22,),
         ||left(' ',15,' ')
 Out.2 = left(' ',1Ø,' '),
         ||center('Report produced on',18,),
         ||left(' ',1,' ')||left(date(),11),
         ||left(' ',1,' ')||left('at ',3,' '),
         ||left(time(),1Ø)
 Out.3 = '  '
 Out.4 =  left('LLA entry',28) left('members',8),
     left('members',8) left('members',8),
     left('dasd',8)  left('vlf',4)
 Out.5 =  left(' ',37) left('fetched',8) left('in vlf',7),
     left('fetched',8) left('retrieves',9)
 Out.6 = LEFT('-',73,'-')
 t = 6
/*-------------------------------------------------------------------*/
/* read command output                                               */
/*-------------------------------------------------------------------*/
  tmem  = Ø
  tmemf = Ø
```

```
  tmemv = Ø
  tdasd = Ø
  tvlfr = Ø
  i=4
 do while(i <= row.Ø)
       DSName = SUBSTR(row.i,11,25)
       i = i+1
       mem = SUBSTR(row.i,26,5)
       tmem = tmem + mem
       i = i+1
       memf = SUBSTR(row.i,26,5)
       memv = SUBSTR(row.i,54,5)
       tmemf = tmemf + memf
       tmemv = tmemv + memv
       i = i+1
       dasdF =SUBSTR(row.i,26,5)
       vlfr=  SUBSTR(row.i,54,5)
       tdasd = tdasd + dasdf
       tvlfr = tvlfr + vlfr
       i = i+1
 t = t+1
/*-------------------------------------------------------------------*/
/* formatting and printing list of LLA enties                        */
/*-------------------------------------------------------------------*/
 Out.t = left(dsname,27) left(mem,7) left(memf,8) left(memv,8),
      left(dasdf,7) left(vlfr,7)
  end
 t = t + 1
/*-------------------------------------------------------------------*/
/* print footer                                                      */
/*-------------------------------------------------------------------*/
 Out.t = LEFT('-',67,'-')
 t = t + 1
 Out.t = left('Totals',27) left(tmem,8) left(tmemf,9) left(tmemv,7),
      left(tdasd,6) left(tvlfr,7)
 t = t + 1
 Out.t = LEFT(' ',63,' ')
 t = t + 1
elapsed_seconds = Time(E)
cpu_seconds     = Sysvar(SYSCPU)-cpu_seconds
Out.t = 'This REXX has used' cpu_seconds 'CPU-seconds' ,
    'and' elapsed_seconds 'elapsed time seconds'
 Out.Ø = t
  "EXECIO Ø DISKW fout (OPEN)"
  "EXECIO * DISKW fout (STEM Out.)"
  "EXECIO Ø DISKW fout (FINIS)"
  Address ISPEXEC "ISPEXEC BROWSE DATASET("outds")"
  "free FILE(fout)"
 EXIT
```

By using the report a user will get only an overview of how LLA/VLF handles libraries. But to obtain a detailed monitoring of LLA-managed module libraries and the VLF data space used for modules cacheing there should be a tool that can be used to identify the modules used, by whom (jobs, STC, TSO), and when.

There is indeed such an (still unofficial) LLA monitor tool called the 'Module Fetch Monitor', which is available from the Washington Systems Center. It is possible to get a copy of the tool, but you will have to sign a licence agreement acknowledging that there is no warranty with the tool. Using this tool, a user can view module statistics in VLF and LLA through an ISPF dialog or batch interface.

Data is collected in the following way: each time the LLA fetches a module from an LLA library, it logs statistics and then calls the installation exit CSVLLIX1. LLA fetch passes to CSVLLIX1 the address of a parameter list containing fetch statistics, the address of a user work area, and a copy of the module's BLDL format PDS directory entry. The LLA fetch installation exit, when active, receives control after every LLA-managed module is fetched, in every address space.

No IPL is required to start the MFM if an authorized library is available. It can be executed either by starting a procedure, or submitting a job. If started as a procedure, it cannot be cancelled but only stopped. When started as a submitted job, it can be cancelled or stopped.

MFM is a real-time data collector and statistics are saved only for the duration of the started task. When the program is stopped, the data is lost. The collected data (the library names, the module names, counters, size, and fetch's response time from DASD and VLF) is written into two sequential datasets used alternately. However, the amount of data collected can be huge, and, in order to reduce the amount of data, the monitor has a filter facility. The source of filter is provided with the package. Figure 1 shows the Main Menu displayed when you start the application.

```
                Module Fetch Monitor - Main Menu

Select an option:

    1.  MODULE List                          Date & Time : 99.337   15:3
    2.  DATA_SET List                Monitor started at : 99.335   16:Ø
    3.  LLA Allocated Data_Sets      Elapsed seconds    :         17122
    4.  Filter List                  SMF System ID      :           SC4
--------------------------------------------------------------------------
COUNTERS:

Number of LINK . . . .:      986  Number of LOAD . . . .:   557584
Number of XCTL . . . .:      655  Number of ATTACH . . .:      438
Number of ESR SVC. . .:       43

Module Names. . . . . :      241
Data_set Names. . . . :        7
Job Names . . . . . . :      7Ø1  Data Space Used .bytes:    71ØØ4    Ø%
Lost Events . . . . . :        Ø  Active Filter. . . . .:       Ø6

Command ===>
_____________________________________________________________________
F1=Help   F2=Split  F3=Exit   F9=Swap   F12=Cancel
```

Figure 1: Main Menu – ISPF application

The OPTION 1 shows the MODULES fetch statistics. The command L MODULE_NAME positions the table on the module name. The search is done by matching the beginning of a module name with the locate argument. Entering S on a row you get the statistics information for the module.

If you specify V on the command line, the display panel shows the modules that were resident in the VLF data space the last time they were referenced as well as the storage utilization. Entering J on a row gives the list of jobnames that requested the module with the first and last reference time. Option 2 of the main menu shows the modules' library dataset names managed by LLA. By selecting a single dataset with S, the modules fetch statistics for the selected library are presented.

Enter V on the command line and the display panel shows the modules of the selected library that were resident in the VLF data space the last time they were referenced. Selecting a module name using S you get the statistics information, and using J you get the information about the job names that referenced the module. Option 3 of the main menu shows the LLA allocated and managed datasets in alphabetical order. The list produced is similar to the list produced by the MVS command D LLA, but the MVS command gives more detail. The Option 4 of the main menu shows the filter list.

MFM provides a set of programs and JCL procedures to generate different report types from the collected data. The following reports are generated.

SEQUENTIAL TRACE REPORT

The output generated contains the following information:

- Date and time of the request
- SVC type
- Jobname of the requester
- UserID of the requester
- Stepname of the requester
- Program name of the requester
- Module authorization information
- Library DSname if LLA managed
- Called program
- Statistics information if LLA managed
- SMF system ID.

MODULE REPORT BY REFERENCE COUNT

For every module the following information is given:

- Count of module reference
- Module name
- Number of link, XCTL, load, and attach requests
- Library DSname if LLA managed
- Last reference date and time.

MFM is a really useful tool and it helped me to determine what modules should be placed in LLA libraries. Based on recommendation or through trial and error, modules that were added were seldom (if ever) removed. That was prior to using MFM. By watching MFM's module usage statistics it is now a routine job to add or remove modules without any guesswork.

For those of you who are interested to obtaining more information, MFM is described in a SHARE presentation at the following URL: http://www.share.org/proceedings/sh96/data/S2876.PDF.

Mile Pekic
Systems Programmer
Postal Savings Bank (Yugoslavia)

Using in-stream data to help a started task

INTRODUCTION

Have you ever felt the need to provide in-stream data to your started tasks? As everybody knows, STCs do not allow you to provide data by this means. The PROCALD utility provided in this article may be the answer.

Using data in-stream (under SYSIN or similar DDnames) is a problem within started tasks. In-stream data is not allowed and sometimes it could be useful for this data to be included. You could also find some applications of this utility in batch.

This program may help you generate dynamic files and feed them with data taken directly from the PARM JCL statement. Think about it: invoke utilities (such as IEBUPDTE, AMASPZAP, IDCAMS), user programs, or the widespread batch TSO (IKJEFT01), and pass data to them just from JCL, with no SYSIN or SYSTSIN required.

This utility could be useful to generate dynamic files with values obtained directly from the PARM of JCL. As an example, you could invoke utilities such as IEBUPDATE or AMASPZAP, user programs, TSO in batch, IDCAMS and so on, passing data from JCL to the started task. Different users at our installation have found a lot of applications for this program in their daily work. You will find instructions on how to use this program (parameters and special data) at the begining of the source code.

This program makes use of dynamic allocation services, calling SVC99. We have solved this problem by using another installation written module (MDALDIN), which is not supplied. You can use your own implementation of this function, or you can get any another shareware sample.

Here is an example you can use to provide your operations staff with a tool to delete or rename a dataset (USER.JCL) with just a START command.

```
CONSOLE -----> S PROCALD,FUN=DL,FIN1=USER.JCL                    >DELETE
CONSOLE -----> S PROCALD,FUN=RN,FIN1=USER.JCL,FIN2=USER.JCL.OLD  >RENAME

PROCLIB------> //PROCALD  PROC FUN=ZA,FIN1=,FIN2=
               //PROCALD  EXEC PGM=PROCALD,PARM=('&FUN|&FIN1|&FIN2')
               //SYSPRINT DD   SYSOUT=*
               //SYSTSPRT DD   SYSOUT=*
               //USRDATOS DD   DISP=SHR,DSN=SYST.PATTERN

SYST.PATTERN-> Seq. File of Partitioned member with following cards
               ....+....1....+....2....+....3....+
               DLPGM=IDCAMS
               DL DEL &P1
               *
               RNPGM=IKJEFTØ1
               RN RENAME '&P1' '&P2'
               ....+....1....+....2....+....3....+
```

SOURCE CODE

```
TITLE '@@ PROCALD @@ DYNAMIC SYSIN FOR PROCS '
//**** OS3Ø6A  ****  P R O C A L D ****************** <YR2ØØØ> ******/
//* NAME ......... PROCALD                                          */
//* LIBRARY   .... SYST.LINKLIB  NORENT - AMODE(24) RMODE(24)       */
//* VERSION ...... OS3Ø8B                                           */
//* FUNCTION  .... HELPING STARTED TASK TO USE INSTREAM DATASETS    */
//* LANGUAGE ..... ASSEMBLER                                        */
//* ATTRIBS   .... NONE                                             */
//* TYPE ......... PGM                                              */
//* FILES ........ OPEN DDNAME USRDATOS, WHICH MUST POINT TO A      */
//*                SEQUENTIAL FILE OR PDS MEMBER, WITH LRECL=8Ø.    */
//*                                                                 */
//*                DATA STRUCTURE INSIDE FILE WILL BE:              */
//*                                                                 */
//*                POS Ø1-Ø2  KEY FOR DATA TO BE SELECTED BY PROGRAM */
//*           (*) POS Ø3-71  DATA TO BE USED AND/OR MODIFIED BY     */
//*                          PROGRAM BEFORE ALLOC DDNAME SYSIN. (*) */
//*                          THEY CAN CONTAIN VARIABLE NAMES FROM   */
//*                          &1 TO &9 AND THAT WILL BE CHANGED BY   */
//*                                                                 */
//*                          SAME ORDER BY VALUES SPECIFIED IN PARM.*/
//*                          AFTER ITS EXPANSION, SELECTED DATA,    */
//*                          WILL BE TRANSFERRED TO THE ALLOC FILE  */
//*                          IN POSITION 1-69.                      */
//*                          AS SPECIAL CASE, IT IS POSIBLE TO USE  */
//*                          THE &D VARIABLE. IT MEANS THAT MUST BE */
//*                          SYSUT1 ALLOCATED IN DE JCL. PROCALD    */
//*                          WILL EXTRACT THE DS NAME REFERED IN    */
//*                          SYSUT1 AND WILL SUBSTITUTE &D VARIABLE */
//*                          IN THE TEXT.                           */
//*                POS 72    CONTINUATION CHARACTER THAT WILL BE    */
//*                          PASSED TO POSITION 72 IN DEFINITIVE    */
//*                          ALLOCATED FILE.                        */
//*                                                                 */
//*           (*)            SPECIAL CARDS                          */
//*                                                                 */
//*                POS-3     PGM=XXXXXX                            */
//*                          XXXXXX IS THE PGM NAME TO BE INVOQUED  */
//*                          IDCAMS/IKJEFTØ1/UTILITY IBM/PGMUSER    */
//*                                                                 */
//*                  POS-3   DDNAME=XXXXXX                         */
//*                          IF PRESENT, IT WILL BE USED TO ALLOC   */
//*                          TEMPORARY FILE INSTEAD OF SYSIN        */
//*                                                                 */
//*                  POS-3   DDNAME=$$XXXX                         */
//*                          IN CASE TWO FIRST CHARACTERS ARE $$,   */
//*                          IT WILL NO BE DYNAMIC ALLOCATION, SO   */
//*                          THE PROGRAM WILL USE OPEN DIRECTLY TO  */
//*                          THIS DDNAME, EXPLICITY ASIGNED IN JCL  */
```

```
//*                          BY THE USER. IF NOT PRESENT, THERE WILL*/
//*                          BE ABEND AND DD STATEMENT MISSING      */
//*                                                                 */
//* PARAMETERS ... MUST EXIST, WITH AT LEAST TWO POSITIONS THAT WILL */
//*                IDENTIFY DATA TO BE SELECTED FROM FILE USRDATOS.  */
//*                                                                 */
//* THE PROGRAM WILL TREAT ONLY THOSE RECORDS WITH THIS TWO CHARACTER*/
//* // PREFIX OTHER RECORDS WILL BE IGNORED.                        */
//*                                                                 */
//*             TWO POSITIONS MUST MATCH THE TWO FIRST POSITION IN  */
//*             THE FILE USRDATOS. ONLY THOSE WHICH MATCH IN        */
//*             SEQUENCE WILL BE TREATED, OTHERS WILL BE IGNORED.   */
//*                                                                 */
//*               MORE DATA IN PARM ARE OPTIONAL, AND WILL BE       */
//*               SEPARATED BY SPECIAL  CHARACTER "|". AS ITS       */
//*               POSITION WILL SUSTITUTE &1 TO &9 VARIABLES IN     */
//*               SELECTED DATA FROM USRDATOS FILE.                 */
//*                                                                 */
//* ERRORS ....... ABEND 77  NO PARM PRESENT OR LENGTH < 2          */
//*                ABEND 88  DYNAMIC ALLOCATION ERRORS              */
//*                                                                 */
//*******************************************************************
PROCALD  CSECT
RØ       EQU   Ø
R1       EQU   1                  ----- ATTENTION -----
R2       EQU   2
R3       EQU   3
R4       EQU   4                  PARAM LIST TRANSFERRED TO R2
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
         USING PROCALD,R15
         B     LØØØ1
         DC    AL1(IØØØ1-*)
         DC    CL8'PROCALD'
         DC    C'_SYSIN DYNAMICS IN PROCS. '
IØØØ1    EQU   *
SAVE     DS    18F
CR       DC    X'ØØ'
LØØØ1    DS    ØH
         STM   R14,R12,12(R13)    SAVE REGISTERS PREVIOUS TASK
         LA    R12,*+6             CAPPING
         BSM   R14,R12             CAPPING
```

```
         STCM  R14,B'1111',12(R13) CAPPING
         DROP  R15
         LR    R2,R1               R2 POINTS PARM LIST
         LR    R12,R15             ADDR: FIRST BASE REGISTER
         USING PROCALD,R12         USING PROGRAM NAME
         USING PROCALD+4Ø96,R11
         LA    R11,2Ø48(,R12)
         LA    R11,2Ø48(,R11)
         MVI   CR,X'ØØ'            RETURN CODE TO ZERO
         LA    R1,SAVE             LINKING
         ST    R13,4(,R1)          STANDARD
         ST    R1,8(,R13)          DE LAS
         LR    R13,R1              SAVE AREAS
*        +--------------------------+
*--------|   TTØØ                     |---------------------------------
*        |   VERIFICATION AND FORMAT  |
*        |   OF SENTENCE PARM         |
*        +--------------------------+
         L         R2,Ø(R2)            ADDR: PARM
         SLR       R1,R1               CLEAR REG.1
         LH        R1,Ø(R2)            PARM LENGTH
         LTR       R1,R1               IS ZEROS?
         BZ        ABEND77             MISSING PARM
         C         R1,KF2              ALMOST TWO CHARS?
         BNH       ABEND77             PARM ERROR
*
*        R3 -----> A(PARMADDR)  ADDR.FIELDS IN STORE PARM
*        R4 -----> A(PARMWORK)  ADDR.WORKA POSITION
*        R6 -----> A(PARM-REAL) ADDR.ADVANCING IN REAL PARM
*        R7 -----> L(PARM-REAL) PENDING LENGTH IN REAL PARM
*
         XC        PARMADDR,PARMADDR   CLEAR X'ØØ' AREA DIRECTIONS
         XC        PARMWORK,PARMWORK   CLEAR X'ØØ' AREA DIRECTIONS
*
         LA        R3,PARMADDR         R3-A(PARMADDR)
         LA        R4,PARMWORK         R3-A(PARMWORK)
         LA        R6,2(,R2)           R6-A(REST OF PARM)
         SLR       R7,R7               CLEAR R7
         LH        R7,Ø(,R2)           R7-L(REST OF PARM)
*
TTØØBUC  EQU       *
         LTR       R7,R7               STILL POS. IN  PARM ?
         BZ        TTØØB               END TTO PARM ENDS
*
         LR        R15,R6              STORE FIELD START
         SLR       R1,R1               CLEAR R1
TTØØA2Ø  EQU       *
         CLI       Ø(R6),C'|'          SPECIAL SEPARATOR CHAR
         BE        TTØØA2ØA
         LA        R6,1(,R6)           NEXT POSITION
         LA        R1,1(,R1)           ADD 1 TO POSIT. TOTAL
```

```
         BCT       R7,TT00A20          DO LOOP
*
         B         TT00A20B
*
TT00A20A EQU       *                   SPECIAL CHAR DETECTED
         BCTR      R7,R0               MINUS ON POSIT.CH.SPEC
         LA        R6,1(,R6)           ADDR NEXT POS. SPEC.CHAR.
TT00A20B EQU       *
         ST        R4,0(R3)            STORE ADDRESS
         LR        R14,R1              STORE FIELD LENGTH
         STCM      R1,B'0001',0(R4)
         LTR       R1,R1               NULL FIELD?
         BZ        TT00A20C            GO WITHOUT MOVING
         BCTR      R1,R0               MINUS ONE FOR EXECUTE
         EX        R1,TT00EX           EXECUTE VAR MVC
*
TT00A20C EQU       *                   CH.SPEC. DETECTED
         LA        R3,4(,R3)           NEXT ADDR.
         LA        R4,1(,R4)
         AR        R4,R14              FIELD LENGTH
*
         B         TT00BUC             LOOP AGAIN FOR NEW FIELD
*
TT00EX   MVC       1(0,R4),0(R15)
*
TT00B    EQU       *
         CLI       PARMWORK,X'02'      FIRST FIELD IN PARM MUST HAVE
         BNE       ABEND77             TWO CHARACTER IN LENGTH
         EJECT
*        +--------------------------+
*--------|   TT10                   |---------------------------------
*        |   ACCESS TO THE FILE WITH |
*        |   DDNAME USRDATOS        |
*        +--------------------------+
TT10     EQU       *
         OPEN      (USRDATOS,(INPUT))
*
TT10LEER EQU       *
         GET       USRDATOS
         LR        R3,R1               DATA AREA FOR USR FILE
         CLC       0(2,R3),PARMFUNC    LOOK CARD TYPE
         BNE       TT10LEER
         CLC       2(L'KPGM,R3),KPGM   SPECIAL CARD FOUND
         BNE       TT05A
         MVC       STPGM,2+L'KPGM(R3)  STORE PGMNAME
         B         TT10LEER            GET AGAIN
TT05A    CLC       2(L'KDDNAME,R3),KDDNAME  SPECIAL CARD FOUND
         BNE       TT10A10
         MVC       STDDNAME,2+L'KDDNAME(R3)
         MVI       MDDAT2A+3,C' '
         MVC       MDDAT2A+4(L'MDDAT2A-4),MDDAT2A+3
```

```
         MVC       MDDAT2A+3(8),2+L'KDDNAME(R3)  DYNAMIC ALLOC.
         MVC       DDNMESYS,2+L'KDDNAME(R3)      NAMES UTILITIES
         LA        R15,MISYSIN
         USING     IHADCB,R15
         MVC       DCBDDNAM,2+L'KDDNAME(R3)      DCB OPEN GENER.
         DROP      R15
         LA        R1,MDDAT2A+L'MDDAT2A-1
TTØ5B    EQU       *
         CLI       Ø(R1),C' '
         BNE       TTØ5C
         BCT       R1,TTØ5B
TTØ5C    EQU       *
         MVI       1(R1),C';'          NEEDED BY MODULE MDALDIN
         B         TT1ØLEER            GET ANOTHER REGISTER
*
*        EXPANDING/DELETING POSITIONAL PARAMETERS
*        FUNCTIONS &1/&9 --> BY DATA PARM IF FACILITATED
*        FUNCTION  &D -----> DSNAME FOR SYSUT1 DDNAME
*
TT1ØA1Ø  EQU       *
         MVC       AREAWR(7Ø),2(R3)
         MVC       AREAWR+71(1),71(R3) CONT CHARACTER
         LA        R3,AREAWR           WORKING IN AREAWR AREA
         SLR       R4,R4
         LA        R4,7Ø               MAXLENGTH DATA IN AREAWR
*
TT1ØBUC  EQU       *
         CLI       Ø(R3),X'5Ø'         FINDING AMPERSAND
         BNE       TT1ØBUCF
*
         CLI       1(R3),C'D'          &D
         BNE       TT1ØFUN
         BAL       R1Ø,PR1ØDSN         OBTAIN  DSNAME
         MVC       WORKSUS,STDSNAME    SETTING WORKA FOR SUSTITUC.
         B         TT1ØSUS             DO SUSTITUTION
TT1ØFUN  EQU       *                   &1 A &9
         CLI       1(R3),C'1'          BEYOND LIMITS ?
         BL        TT1ØBUCF            WITHOUT TTO.
         CLI       1(R3),C'9'          BEYOND LIMITS ?
         BH        TT1ØBUCF            WITHOUT TTO.
         PACK      WORKD,1(1,R3)       CONVERTING TO DECIMAL
         MP        WORKD,KP4           EVALUATING
         CVB       R1,WORKD
         LA        R14,PARMADDR        ADDRESSING PARM FIELDS
         AR        R14,R1              POINT SPECIAL AND TO SUBST.
         L         R14,Ø(R14)          LOAD ADDRESS
         XC        WORKSUS,WORKSUS
         LTR       R14,R14             FIELD EXISTS ?
         BZ        TT1ØSUS             NO, NO SUSTITUTION
         MVC       WORKSUS,Ø(R14)      SETTING WORK FOR REPLACEMENT
         B         TT1ØSUS             DO REPLACEMENT
```

```
*
TT1ØSUS  EQU       *
         SLR       R1,R1               CLEAR R1
         ICM       R1,B'ØØØ1',WORKSUS  LENGHT FIELD TO REPLACE
         LTR       R1,R1               IS IT A NULL FIELD?
         BNZ       TT1ØSUSB            NO, DO REPLACE
*
         LR        R15,R4              PENDING LENGTH
         BCTR      R15,RØ              SUBSTRACT
         BCTR      R15,RØ              SUBSTRACT THREE
         BCTR      R15,RØ              SUBSTRACT
         EX        R15,TT1ØMNUL        CLEARING &X
         MVI       AREAWR+68,C' '      RE-POSITIONING
         MVI       AREAWR+69,C' '      RE-POSITIONING
         B         TT1ØBUC             DO LOOP AGAIN
*
TT1ØSUSB EQU       *
         CR        R1,R4               ENOUGH BYTES TO REPLACE ?
         BH        TT1ØBUCF            NO, LEAVING AMPERSAND
         LR        R1,R4
         BCTR      R1,RØ
         EX        R1,TT1ØMWOR         STORE REST IN WORK
         SLR       R1,R1
         ICM       R1,B'ØØØ1',WORKSUS  LENGTH FILED TO REPLACE
         BCTR      R1,RØ
         EX        R1,TT1ØMDAT         DO REPLACEMENT
         SLR       R1,R1
         ICM       R1,B'ØØØ1',WORKSUS  LENGTH FIELD TO REPLACE
         AR        R3,R1               POSITIONING ...
         SR        R4,R1               SET PENDING LENGTH
         LTR       R4,R4               MORE DATA?
         BZ        TT1ØWRIT
         LR        R1,R4
         BCTR      R1,RØ
         EX        R1,TT1ØMARE
         B         TT1ØBUC             DO LOOP
*
TT1ØBUCF LA        R3,1(,R3)           ADD 1 BYTE
         BCT       R4,TT1ØBUC          CLOSING LOOP
*
TT1ØWRIT EQU       *
         BAL       R9,PRØ9WRT
         B         TT1ØLEER            READING ANOTHER REGISTER
*
TT1ØMWOR MVC       STWORKA(Ø),2(R3)    STORING
TT1ØMDAT MVC       Ø(Ø,R3),WORKSUS+1   MOVE DATA TO AREA
TT1ØMARE MVC       Ø(Ø,R3),STWORKA     VOLVER WORKA A AREA
TT1ØMNUL MVC       Ø(Ø,R3),2(R3)
*
```

```
*        +---------------------------+
*--------|   TT2Ø                    |---------------------------------
*        |   CLOSING FILES AND       |
*        |   CALLING PGM/UTILITY     |
*        +---------------------------+
USREOF   EQU       *
         CLOSE     (USRDATOS)
         TM        SWALLOC,X'8Ø'       WAS IT DATA FOR FILE?
         BZ        ENDOFPGM            NO, END
         CLOSE     (MISYSIN)           CLOSE FILE
         L         R1,MDALWORK
         LTR       R1,R1               PREVIOUS WORK FOR MODULE ?
         BZ        TT2ØAA              YES, JUMP OVER
         CALL      MDALDIN,(MDALWORK,KOPEND),VL
TT2ØAA   EQU       *
*
         CLI       STPGM,C' '          CALL PGM TYPE?
         BE        TT2ØEND
         CLC       STPGM(L'KTSO),KTSO  CALL TSO TYPE?
         BE        TT2ØAT
         CLC       STPGM(L'KIDCAMS),KIDCAMS
         BE        TT15A
         CLC       STPGM(L'KUTILIT1),KUTILIT1
         BE        TT15A
         CLC       STPGM(L'KUTILIT2),KUTILIT2
         BE        TT15A
         B         TT18A               USER PGM
*
*        CALLING TO TSO IN BATCH
*
TT2ØAT   EQU       *                   CALLING TSO
         LA        R1,*+4+4+2          CHANGING  AMODE(31)
         O         R1,KBSM
         BSM       RØ,R1
*
         LINK      EP=IKJEFT1B,PARAM=(TSOPARM),VL=1
         L         R13,4(,R13)
         LM        RØ,R12,2Ø(R13)
*
         STC       R15,CR
*
         LA        R1,*+4+4+2            GO AMODE(24)
         N         R1,KBSMNO
         BSM       RØ,R1
         B         TT2ØEND
*
*        CALL UTILITY / IDCAMS
*
TT15A    EQU       *
         LINK      EPLOC=STPGM,PARAM=(OPTLST,DDNMELST),VL=1
         STC       R15,CR
```

```
         B         TT2ØEND
*
*        CALL OTHERS (USERS PROGRAMS)
*
TT18A    EQU       *
         LINK      EPLOC=STPGM
         STC       R15,CR
         B         TT2ØEND
*
TT2ØEND  EQU       *
         EJECT
*        +-----------------------------------+
*--------|   ENDING PROGRAM                  |--------------------------
*        +-----------------------------------+
ENDOFPGM EQU       *
         L     R13,SAVE+4         PREVIOUS SAVE
         SR    R15,R15            CLEAR RETURN CODE
         IC    R15,CR             LOAD RETURN CODE
         L     R14,12(R13)        RESTORE RETURN ADDRESS
         LM    RØ,R12,2Ø(R13)     RESTORE PREVIOUS SAVE AREA
         BSM   RØ,R14             RETURN
*
ABEND77  ABEND     77                  <> NO PARM/ERROR IN FUNCTION
ABEND88  ABEND     88                  <> ERROR MDALDIN MODULE
*
         EJECT
*        +-----------------------------------+
*--------|   WRITE DATA TO SYSIN FILE        |--------------------------
*        |                                   | ENLACE: R9
*        +-----------------------------------+
PRØ9WRT  EQU       *
         TM        SWALLOC,X'8Ø'       WAS A FILE?
         BO        PRØ9W5Ø             WRITE
         CLI       STDDNAME,C'$'       SPECIAL DDNAME PREALLOC?
         BE        PRØ9OPEN            DO OPEN
*
*        DYNAMIC SYSIN ALLOCATION
*
         L         R1,MDALWORK
         LTR       R1,R1               HAVE WE WORK?
         BZ        PRØ9WRT1
         CALL      MDALDIN,(MDALWORK,KOPINIT),VL
         LTR       R15,R15
         BNZ       ABEND88
PRØ9WRT1 EQU       *
         CALL      MDALDIN,                                            X
               (MDALWORK,KOPALLOC,MDALINFO,MDDAT2A,MDDAT2B,MDDAT2C),VL
         LTR       R15,R15             RETURN CODE MDALDIN
         BNZ       ABEND88             POSITIVE, ERROR
*
PRØ9OPEN DS        ØH
```

```
         OPEN      (MISYSIN,(OUTPUT))  OPEN INTERNAL READER
         OI        SWALLOC,X'8Ø'       SET FLAG
*
PRØ9W5Ø  EQU       *
         PUT       MISYSIN
         MVC       Ø(8Ø,R1),AREAWR     MOVE WRITE AREA
*
         MVI       AREAWR,C' '
         MVC       AREAWR+1(L'AREAWR-1),AREAWR
*
         BR        R9                  RETURN VIA R9
*        +----------------------------------+
*--------|   OBTAIN DSNAME WHICH IS IN      |-------------------------
*        |   DDNAME SYSUT1                  | ENLACE: R1Ø
*        |   USING MODULE                   |
*        +----------------------------------+
PR1ØDSN  EQU       *
         CLI       STDSNAME,X'ØØ'      OBTAINED DSN PREVIOUSLY ?
         BNE       Ø(R1Ø)              YES, RETURN
         L         R1,MDALWORK
         LTR       R1,R1               WORKA ?
         BZ        PR1ØDSN1
         CALL      MDALDIN,(MDALWORK,KOPINIT),VL
         LTR       R15,R15
         BNZ       ABEND88
PR1ØDSN1 EQU       *
         MVI       MDALINFO,C' '
         MVC       MDALINFO+1(L'MDALINFO-1),MDALINFO
         CALL      MDALDIN,                                          X
               (MDALWORK,KOPINFO,MDALINFO,MDDAT1),VL
         LTR       R15,R15             RETURN MODULE MDALDIN
         BNZ       ABEND88             SI NO CERO, ERROR
         MVC       STDSNAME+1(44),MDALINFO+8 <> STORE DSNAME
         LA        R15,STDSNAME+44     ENDING POSITION DSNAME
PR1ØDSNB EQU       *
         CLI       Ø(R15),C' '         NON BLANK?
         BNE       PR1ØDSNX
         BCT       R15,PR1ØDSNB        LOOP
PR1ØDSNX EQU       *
         LA        R15,1(R15)          POINT FIRST BLANCK
         LA        R1,STDSNAME+1       STARTING FIELD
         SR        R15,R1              DSNAME LENGTH
         STCM      R15,B'ØØØ1',STDSNAME  STORE LENGTH
         BR        R1Ø                 RETURN
         EJECT
*        +----------------------------------+
*--------|   POINTING FIRST BLANK           |-------------------------
*        |   IN WORKAREA AREAWR             | LINK VIA R1Ø
*        |   RETURNING A(AREAWR) EN R1      |
*        +----------------------------------+
```

```
PR1ØBLA  EQU       *
         SLR       R1,R1
         LA        R1,AREAWR+L'AREAWR-1 POINT LAST POSITION
PR1ØBLA1 EQU       *
         CLI       Ø(R1),C' '
         BNE       PR1ØBLA2
         BCT       R1,PR1ØBLA1
PR1ØBLA2 EQU       *
         LA        R1,1(,R1)
         BR        R1Ø                 RETURN CONTROL
         EJECT
* ---------------------------------------------------------------------
*        STATIC AREA DEFINITION
* ---------------------------------------------------------------------
KVERSION DC    CL7'OS39Ø6A'
         DS        ØF
KBSM     DC        X'8ØØØØØØØ'
KBSMNO   DC        X'7FFFFFFF'
KF2      DC        F'2'
KF3      DC        F'3'
KP4      DC        PL1'4'
KPGM     DC        C'PGM='
KDDNAME  DC        CL7'DDNAME=='
KTSO     DC        CL8'IKJEFTØ1'
KSTCNAME DC        CL8'SGCO    '
KUTILIT1 DC        CL3'IEB'
KUTILIT2 DC        CL3'IEH'
KIDCAMS  DC        CL8'IDCAMS  '
KSTC     DC        CL3'STC'
K$$      DC        C'$$'
TSOPARM  DC        H'4',CL4'TIME'
*
WORKD    DS        D
WORKF    DS        F
WORKH    DS        H
*
WORKSUS  DS        CL111          WORKA FOR SUSTITUTIONS
*
SWALLOC  DS        CL1            ASSIGNING FILES
*
AREAWR   DC        CL8Ø' '        WRITTIN AREA
*
STPGM    DC        CL8' '         PGMNAME TO CALL
STDSNAME DC        X'ØØ',CL44' '  DSNAME LENGTH
STDDNAME DC        CL8' '         DDNAME LENGTH
STVOLSER DC        CL6' '         VOLSER
STWORKA  DC        CL1Ø2' '       WORKAREA TO STORE PARAM
*
OPTLST   DC        H'Ø'           CALL UTILITIES WITHOUT PARMS
DDNMELST DC        H'4Ø'
```

```
         DC        XL8'0000000000000000'
         DC        XL8'0000000000000000'
         DC        XL8'0000000000000000'
         DC        XL8'0000000000000000'
DDNMESYS DC        CL8'SYSTSIN '  VALID TO UTILITIES/TSO
*
KOPINIT  DC        CL10'INIT '
KOPEND   DC        CL10'END  '
KOPINFO  DC        CL10'INFO '
KOPALLOC DC        CL10'ALLOCR '
MDDAT1   DC        CL12'DD=SYSUT1; '
MDDAT2A  DC        CL50'DD=SYSTSIN; '
MDDAT2B  DC        CL50'DSN=&&&&TEMPOR NEW DELETE UNIT=VIO ; '
MDDAT2C  DC        CL50'CYL PRIM=1 SEC=1 FB LRECL=80 BLKSIZE=80; '
MDALINFO DC        CL70' '
MDALWORK DC        F'0'                WORK RETURNING VIA MODULE
*
MISYSIN  DCB       DDNAME=SYSTSIN,DSORG=PS,MACRF=PL
USRDATOS DCB       DDNAME=USRDATOS,DSORG=PS,MACRF=GL,EODAD=USREOF
         LTORG
*
*        AREA FOR PROCESSING OF THE PARM
*
PARMWORK DS        0CL128              WORKA TO STORE PARAMETERS.
*                                      AFTER FORMATTING THEM
*                                      BEFORE EACH THERE WILL BE ONE
*                                      BYTE WITH ITS LENGTH
         DS        CL1                 REDEF. PARMWORK
PARMFUNC DS        CL2                 REDEF. PARMWORK - FUNCTION
         DS        CL125               REDEF. PARMWORK - REMAINING
PARMADDR DC        100F'0'             ADDR FROM 0 TO 100 PARAM.POS.
*
         INSISTMA
         DCBD      DSORG=PS
         IHAASCB   LIST=YES
         IEECHAIN
*
       END         PROCALD
```

Angel Domínguez
Systems Analyst
BBVA (Spain)

z/OS HiperSockets Accelerator

A feature of the z/Architecture that has gained renown is the Hipersockets function which allows very high-speed TCP/IP communication among virtual machines, and logical partitions within the same zSeries.

One element of HiperSockets is the HiperSockets Accelerator, which is part of the z/OS Version 1 Release 2 Communications Server. It allows a z/OS TCP/IP router stack to route IP packets from an OSA-Express (QDIO) interface to a HiperSockets (IQDIO) interface and vice versa. Because the routing is done by the Communications Server device driver at the lowest possible software data link control level IP packets do not have to be processed at the higher level TCP/IP stack routing function. This is one reason for the massive performance increases associated with Hipersockets.

To activate the HiperSockets Accelerator you need to configure the IQDIORouting option in the TCP/IP profile using the IPCONFIG statement. Remember that the HiperSockets Accelerator cannot be enabled if FIREWALL or NODATAGRAMFWD are specified on the IPCONFIG statement.

Once activated the TCP/IP stack will detect IP packets prerouting across a HiperSockets Accelerator eligible route automatically. As mentioned previously the routes deemed eligible are from OSA-Express (QDIO) to HiperSockets (IQDIO) and from HiperSockets (IQDIO) to OSA-Express (QDIO). The TCP/IP routing stack creates IQDIORouting route entries for packets for which it is not the destination stack. The destination stack must be reachable through HiperSockets. All subsequent packets for the same destination will take the optimized device driver path, and will not traverse the routing function of the TCP/IP routing stack.

The number of OSA-Express ports on a zSeries is a function of the bandwidth, security, and back-up requirements. One OSA-Express QDIO port can handle up to 240 I/O devices. Each OSA-Express QDIO connection requires three I/O devices, the same as for HiperSockets (IQDIO). Therefore, up to 80 z/VM and Linux TCP/IP stacks can share one OSA-Express QDIO port.

Remember that in order for a TCP/IP router stack to forward IP packets from an OSA device to a HiperSockets device, the OSA port must be defined as PRIRouter on the DEVICE statement in the TCP/IP profile. If no PRIRouter option is defined to the OSA port, IP packets will not be forwarded. Only one PRIRouter can be defined to an OSA port.

A new z/OS Unix operator

We have been long-term users of OS/390 Unix Systems Services and now z/OS Unix. We currently run a z/OS Unix Version 1 Release 2 environment. We recently needed to scan a directory in z/OS Unix and display only the files that have the APF authorization bit on. Before Release 2 we had to write a pipeline to achieve this. However, Release 2 contains a new operator to do this on the find shell command. For example:

```
find /usr/bin -ext a
```

Furthermore, in Version 1 Release 2, there are more operators to test for the extended attributes in conditional expressions on the commands:

```
test
[...]      (square brackets)
[[...]]    (double square brackets)
filetest   (tcsh built-in)
file inquiry options   (in tcsh)
```

These are useful for shell scripts. For example:

```
if [[ -Ea $file ]]
then
   echo $file is APF-authorized
fi
```

We have found these new operators to be very useful indeed.

Automating SMS configuration activation – part 2

In the previous edition of MVS Update *we provided an overview of the functionality of a utility to automate SMS configuration activation. The source is provided below.*

SOURCE CODE

```
//*******************************************************************/
//* STEPs : Paso1, Paso2, Paso3, Paso4                              */
//*                                                                 */
//* These steps get all ACS routine members and header/trailer cards  */
//* together and create 4 unique members : DATACLAS,MGMTCLAS,STORCLAS */
//* and STRGROUP. All these members will have the FILTLIST in common. */
//*******************************************************************/
//*
//PASO1234 EXEC PGM=IDCAMS
//DCI      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(YDC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(FILTLIST)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS($$DC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(ZDC)
//DCO      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(DATACLAS)
//* ----------------------------------------------------------------
//MCI      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(YMC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(FILTLIST)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS($$MC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(ZMC)
//MCO      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(MGMTCLAS)
//* ----------------------------------------------------------------
//SCI      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(YSC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(FILTLIST)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS($$SC)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(ZSC)
//SCO      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(STORCLAS)
//* ----------------------------------------------------------------
//SGI      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(YSG)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(FILTLIST)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS($$SG)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(ZSG)
//SGO      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS(STRGROUP)
//*----------------------------------------------------------------
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=131,BLKSIZE=135,RECFM=VA)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSINØ)
//*
//*******************************************************************/
```

```
//* STEP : Paso5  (TRANSLATE ACS ROUTINES)                          */
//*                                                                 */
//* This step translates ACS routines and stores them in the SCDS.  */
//*                                                                 */
//* ACS routines are translated into table format (called an ACS    */
//* object) which is stored in the SCDS and may then be used by the */
//* system to manage storage. ACS routines must be translated before*/
//* validating them.                                                */
//*                                                                 */
//* This step performs the same operation as the one realized by    */
//* using the following action from the ISMF panels. But unlike the */
//* ISMF, here it is realized IN BATCH.                             */
//*                                                                 */
//*  <<<   Translate ACS routines  (ISMF panel option 7.2 )   >>>   */
//*                                                                 */
//* Later on, the REXX SmsREXX4 is called to interpret the result of*/
//* the operation. if any translate operation ends with a non-zero  */
//* RC, an error message is written to the spool dataset 'LOOKAT1'. */
//*                                                                 */
//* Parameters used in the Systsin DD card (member SYSIN8):         */
//* ------------------------------------------------------          */
//* Acssrc   - PDS containing ACS routines to be translated.. (Input) */
//* Member   - Member name of the routine to be translated... (Input) */
//* Scdsname - Name of SCDS into which the ACS routines are         */
//*            to be translated.............................. (Output) */
//* Listname - Translate Listing ............................ (Output) */
//*******************************************************************/
//*
//PASO5    EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=(6144K),TIME=(300)
//SYSPROC  DD   DISP=SHR,DSN=SYS1.DGTCLIB
//         DD   DISP=SHR,DSN=ISP.SISPCLIB
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//         DD   DISP=SHR,DSN=SYS1.DGTPLIB
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//         DD   DISP=SHR,DSN=SYS1.DGTMLIB
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU,BLKSIZE=0
//         DD   DISP=SHR,DSN=ISP.SISPSLIB
//         DD   DISP=SHR,DSN=SYS1.DGTSLIB
//ISPTLIB  DD   DISP=NEW,DSN=&&TEMP,UNIT=SYSDA,BLKSIZE=0,
// SPACE=(TRK,(1,1,1)),DCB=(ISP.SISPTENU)
//         DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//         DD   DISP=SHR,DSN=ISP.SISPTENU
//         DD   DISP=SHR,DSN=SYS1.DGTTLIB
//ISPTABL  DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSUDUMP DD   SYSOUT=*
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//*
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS),UNIT=SYSDA,
```

```
// SPACE=(TRK,(1,1)),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//*
//LOOKAT1  DD   SYSOUT=*,DCB=(LRECL=81,RECFM=F)
//*
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN8)
//*
//*******************************************************************/
//* STEP : Paso6                                                    */
//*                                                                 */
//* This step copies four translate listings (DC/MC/SC/SG) to the   */
//* spool dataset 'TRANSLAT'.                                       */
//*******************************************************************/
//*
//PASO6    EXEC PGM=IDCAMS
//INPUT    DD   DSN=EXP.D31Ø.SMS.DC.LISTING,DISP=SHR
//         DD   DSN=EXP.D31Ø.SMS.SC.LISTING,DISP=SHR
//         DD   DSN=EXP.D31Ø.SMS.MC.LISTING,DISP=SHR
//         DD   DSN=EXP.D31Ø.SMS.SG.LISTING,DISP=SHR
//TRANSLAT DD   SYSOUT=*,DCB=(LRECL=133,RECFM=FA)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN1)
//*
//*******************************************************************/
//* STEP : Paso7  (VALIDATE ACS ROUTINES)                           */
//*                                                                 */
//* This step validates ACS routines against storage constructs. So,  */
//* you verify that all possible constructs chosen by an ACS routine  */
//* are defined to an SCDS.                                         */
//*                                                                 */
//* Actually this step does the same operation as the one realized by */
//* using the following action from the ISMF panels. But unlike the   */
//* ISMF, here it is realized IN BATCH.                             */
//*                                                                 */
//*  <<< Validate  ACS routines  (ISMF panel option 7.3 or 8.4) >>>   */
//*                                                                 */
//* Later on, the REXX SmsREXX4 is called to interpret the result of  */
//* the operation. If the validation operation ends with a non-zero   */
//* RC, an error message is written to the spool dataset 'LOOKAT2'    */
//*                                                                 */
//* Parameters used in the Systsin DD card (member SYSIN9):         */
//* ------------------------------------------------------          */
//* Scdsname - Name of SCDS that contains the translated ACS routines */
//*            to be validated. ........................... (Input)  */
//*                                                                 */
//* Type     - Type of ACS routine to be validated ......... (Input)  */
//*            Type(*) means all 4 ACS routines (DC/SC/MC/SG).       */
//*                                                                 */
//* Listname - Validate Listing ........................... (Output) */
//*******************************************************************/
//*
//PASO7    EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=(6144K),TIME=(3ØØ),
// COND=(Ø,NE,PASO5)
```

```
//STEPLIB  DD   DISP=SHR,DSN=SYS1.DGTLLIB
//SYSPROC  DD   DISP=SHR,DSN=SYS1.DGTCLIB
//         DD   DISP=SHR,DSN=ISP.SISPCLIB
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//         DD   DISP=SHR,DSN=SYS1.DGTPLIB
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//         DD   DISP=SHR,DSN=SYS1.DGTMLIB
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU,BLKSIZE=Ø
//         DD   DISP=SHR,DSN=ISP.SISPSLIB
//         DD   DISP=SHR,DSN=SYS1.DGTSLIB
//ISPTLIB  DD   DISP=NEW,DSN=&&TEMP,UNIT=SYSDA,BLKSIZE=Ø,
// SPACE=(TRK,(1,1,1)),DCB=(ISP.SISPTENU)
//         DD   DISP=SHR,DSN=INEXØØ4.ISPF.ISPPROF
//         DD   DISP=SHR,DSN=ISP.SISPTENU
//         DD   DISP=SHR,DSN=SYS1.DGTTLIB
//ISPTABL  DD   DISP=SHR,DSN=INEXØØ4.ISPF.ISPPROF
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSUDUMP DD   SYSOUT=*
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS)
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS)
//*
//LOOKAT2  DD   SYSOUT=*,DCB=(LRECL=81,RECFM=F)
//*
//SYSTSIN  DD   DISP=SHR,DSN=EXP.DØ31Ø.SMS.SOURCE(SYSIN9)
//*
//*******************************************************************/
//* STEP : Paso8                                                    */
//*                                                                 */
//* This step copies the validate listing to the spool dataset      */
//* 'VALIDATE'.                                                     */
//*******************************************************************/
//*
//PASO8    EXEC PGM=IDCAMS,COND=(Ø,NE,PASO5)
//INPUT    DD   DSN=EXP.DØ31Ø.SMS.VALIDATE.LISTING,DISP=SHR
//VALIDATE DD   SYSOUT=*,DCB=(LRECL=133,RECFM=FA)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.DØ31Ø.SMS.SOURCE(SYSIN2)
//*
//*******************************************************************/
//* STEP : Paso9  (TEST ACS ROUTINES)                               */
//*                                                                 */
//* This step tests ACS routines using test cases given in the test */
//* library, which is indicated by the Testbed parameter.           */
//*                                                                 */
//* Actually this step does the same operation as the one realized by */
//* using the following action from the ISMF panels. But unlike the */
//* ISMF, here it is realized IN BATCH.                             */
//*                                                                 */
//* <<<      Test ACS routines  (ISMF panel option 7.4.3 )     >>>  */
//*                                                                 */
```

```
//* Parameters used in the Systsin DD card (member SYSINA):           */
//* ------------------------------------------------------            */
//* Scdsname - Name of SCDS that will be based on the test . (Input)  */
//* Testbed  - PDS dataset containing test cases ........... (Input)  */
//* Member   - Members to be tested in Testbed library ..... (Input)  */
//* DC/SC/MC/SG  - Routines to be tested ................... (Input)  */
//* Example: DC(Y) Test the DC routines.                              */
//*          DC(N) Do not test the DC routines.                       */
//* Listname - Test Result listing ......................... (Output) */
//*********************************************************************/
//*
//PASO9    EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=(6144K),TIME=(300),
// COND=((0,NE,PASO7),(0,NE,PASO5))
//STEPLIB  DD   DISP=SHR,DSN=SYS1.DGTLLIB
//SYSPROC  DD   DISP=SHR,DSN=SYS1.DGTCLIB
//         DD   DISP=SHR,DSN=ISP.SISPCLIB
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//         DD   DISP=SHR,DSN=SYS1.DGTPLIB
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//         DD   DISP=SHR,DSN=SYS1.DGTMLIB
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU,BLKSIZE=0
//         DD   DISP=SHR,DSN=ISP.SISPSLIB
//         DD   DISP=SHR,DSN=SYS1.DGTSLIB
//ISPTLIB  DD   DISP=NEW,DSN=&&TEMP,UNIT=SYSDA,BLKSIZE=0,
// SPACE=(TRK,(1,1,1)),DCB=(ISP.SISPTENU)
//         DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//         DD   DISP=SHR,DSN=ISP.SISPTENU
//         DD   DISP=SHR,DSN=SYS1.DGTTLIB
//ISPTABL  DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSUDUMP DD   SYSOUT=*
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS)
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS)
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSINA)
//*
//*********************************************************************/
//* STEP : Paso10                                                     */
//*                                                                   */
//* This step copies the test listing to the spool dataset 'TEST'.    */
//*********************************************************************/
//*
//PASO10   EXEC PGM=IDCAMS,COND=((0,NE,PASO7),(0,NE,PASO5))
//INPUT    DD   DISP=SHR,DSN=EXP.D310.SMS.TEST.TITLE(TITLE1)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.TEST.RESULT1
//         DD   DISP=SHR,DSN=EXP.D310.SMS.TEST.TITLE(TITLE2)
//         DD   DISP=SHR,DSN=EXP.D310.SMS.TEST.RESULT2
//TEST     DD   SYSOUT=*,DCB=(LRECL=133,RECFM=FA)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSIN3)
//*
```

```
//*********************************************************************/
//* STEP : Paso11   (SCDS ──> ACDS)                                   */
//*                                                                   */
//* This step Activates new SMS configuration from current SCDS       */
//* dataset, which was just updated.                                  */
//*                                                                   */
//* If Translate and Validate processings are OK, then it is time     */
//* to activate the new SMS configuration from the SCDS dataset.      */
//* (Because SCDS will have the new configuration after a successfull */
//* Translation & Validation operations.)                             */
//*                                                                   */
//* The REXX SmsREXX1 is called in this step to do all necessary      */
//* tasks. Actually this step does the same operation as the one      */
//* realized by the following ISMF function online:                   */
//*                                                                   */
//*  <<<    Activate the CDS....... (ISMF panel option 8.5 )   >>>    */
//*                                                                   */
//* But in the REXX this activation operation, unlike the one with    */
//* ISMF, is carried out IN BATCH in such a controlled way. Moreover  */
//* the REXX saves current ACDS before changing its contents with the */
//* new configuration structures.                                     */
//*                                                                   */
//* This step issues the following Setsms commands in the SYStem LOG  */
//* by calling SDSF environment from the REXX. In addition, for each  */
//* SMS command issued, the following successful-operation messages   */
//* are searched through the 'Message Line' of SDSF screen, if they   */
//* arrive within the delay interval.                                 */
//*                                                                   */
//* For some reason, if response messages do not come within the      */
//* delay interval, then a 'LOG' command issued and those messages    */
//* are searched through the SYStem LOG.                              */
//*                                                                   */
//* 1st COMMAND          : 'Setsms Acds(Sis.D31Ø.Acds)'               */
//*     EXPECTED MESSAGE : IGDØØ9I Acds switched to Sis.D31Ø.Acds     */
//*     FUNCTION         : Set the correct ACDS dataset.              */
//*                                                                   */
//* 2nd COMMAND          : 'Setsms Saveacds(Sis.D31Ø.Acds.Spare)'     */
//*     EXPECTED MESSAGE :                                           */
//*     IGDØ14I Active configuration saved in Sis.D31Ø.Acds.Spare     */
//*                                                                   */
//*     FUNCTION         : Save the current SMS configuration (ACDS   */
//*                          dataset) before activating a new one.    */
//*                                                                   */
//* 3rd COMMAND          : 'Setsms SCDS (Sis.D31Ø.Scds)'              */
//*     EXPECTED MESSAGE :                                           */
//*     IGDØØ8I New configuration activated from SCDS Sis.D31Ø.Scds   */
//*                                                                   */
//*     FUNCTION         : Activate ACDS from current SCDS.           */
//*                                                                   */
//* All messages related to the SETSMS commands and their responses   */
//* are written to the Lookat3 spool dataset.                         */
```

```
//*******************************************************************/
//*
//PASO11   EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=4096K,
// COND=((0,NE,PASO7),(0,NE,PASO5))
//SYSEXEC  DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD   DISP=SHR,DSN=ISP.SISPTENU
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,UNIT=SYSDA,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1))
//*
//LOOKAT3  DD   SYSOUT=*,DCB=(LRECL=81,RECFM=F)
//*
//SYSUDUMP DD   DUMMY
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS)
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS)
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSINB)
//*
//*******************************************************************/
//* STEP : Paso12                                                   */
//*                                                                 */
//* This step and the next 2 steps save ACS routines and the current  */
//* SCDS dataset for Back-up/Recovery purposes. The SCDS now includes */
//* all changes you have made on your SMS configuration.            */
//*                                                                 */
//* Note that Steps Paso12, Paso13, and Paso14 can be executed only   */
//* if new configuration activated successfully in step Paso11.     */
//* Otherwise they will not be executed.                            */
//*                                                                 */
//* In step Paso12, the current SCDS (Sis.D310.Scds) is copied      */
//* to sequential dataset Exp.D310.Sms.Temp1.                       */
//*                                                                 */
//* In the next step (Paso13) we save this SCDS-copy as a member of a */
//* PDS dataset along with the ACS routines.                        */
//*                                                                 */
//* Finally, you will have all ACS routines and SCDS dataset in a     */
//* single PO dataset Exp.D310.Sms.Temp2, which is allocated in this  */
//* step.                                                           */
//*******************************************************************/
//*
//PASO12   EXEC PGM=IDCAMS,
// COND=((0,NE,PASO7),(0,NE,PASO5),(0,NE,PASO11))
//OUT      DD   DISP=(,CATLG),DSN=EXP.D310.SMS.TEMP1,
// DCB=(RECFM=VBS,LRECL=4100,BLKSIZE=0),UNIT=SYSDA,
// SPACE=(CYL,(1,1))
//*
//ALOCTMP2 DD   DSN=EXP.D310.SMS.TEMP2,DISP=(,CATLG),UNIT=SYSDA,
// SPACE=(CYL,(2,1,1)),DCB=(LRECL=4100,BLKSIZE=0,RECFM=VB)
//*
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
```

```
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN4)
//*
//*******************************************************************/
//* STEP : Paso13                                                   */
//*                                                                 */
//* This step copies the current SCDS (newly-updated SCDS) and the  */
//* current (latest) ACS routines into the Exp.D31Ø.Sms.Temp2 data  */
//* set (Members: $$DC / $$MC / $$SC / $$SG and FILTLIST).          */
//*******************************************************************/
//*
//PASO13   EXEC PGM=IDCAMS,
// COND=((Ø,NE,PASO7),(Ø,NE,PASO5),(Ø,NE,PASO11))
//I1       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP1
//I2       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$DC)
//I3       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$MC)
//I4       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$SC)
//I5       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$SG)
//I6       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($HISTORY)
//I7       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS(FILTLIST)
//*
//O1       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2(SCDS)
//O2       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2($$DC)
//O3       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2($$MC)
//O4       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2($$SC)
//O5       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2($$SG)
//O6       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2($HISTORY)
//O7       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2(FILTLIST)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN5)
//*
//*******************************************************************/
//* STEP : Paso14                                                   */
//*                                                                 */
//* Create a new generation of the GDG dataset Exp.D31Ø.Sms.Bkpcds  */
//* from the dataset Exp.D31Ø.Sms.Temp2, by using the REXX SmsREXX5.*/
//*                                                                 */
//* Here, besides the new generation (OUT1), the current generation */
//* (OUT2) of the BKPCDS is allocated as well.  Each generation of  */
//* the BKPCDS represents a unique day's configuration. In other    */
//* words, no matter how many times a day you change your SMS config.*/
//* and activate it through SMSNEW, only one generation will be saved*/
//* in the BKPCDS.                                                  */
//*                                                                 */
//* This way, you will have all your back-level configurations intact.*/
//*******************************************************************/
//*
//PASO14   EXEC PGM=IKJEFTØ1,DYNAMNBR=5Ø,REGION=(6144K),TIME=(3ØØ),
// COND=((Ø,NE,PASO7),(Ø,NE,PASO5),(Ø,NE,PASO11))
//IN       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP2
//*-----------------------------------------------------------------
//OUT1     DD   DSN=EXP.D31Ø.SMS.BKPCDS(+1),DISP=(,CATLG),
```

```
// SPACE=(TRK,(45,5,2),RLSE),UNIT=SYSDA,FREE=CLOSE,
// DCB=(LRECL=4100,RECFM=VB,BLKSIZE=0)
//*-------------------------------------------------------------------
//OUT2     DD   DSN=EXP.D310.SMS.BKPCDS(0),DISP=MOD,UNIT=SYSDA
//SYSTSPRT DD   DSN=&&TEMP4,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=81,RECFM=FBA)
//*-------------------------------------------------------------------
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSINF)
//*
//*******************************************************************/
//* STEP : Paso15                                                   */
//*                                                                 */
//* This step copies the previous step's SYSTSPRT output to the     */
//* LOOKAT4.                                                        */
//*******************************************************************/
//*
//PASO15   EXEC PGM=IDCAMS,
// COND=((0,NE,PASO7),(0,NE,PASO5),(0,NE,PASO11))
//IN       DD   DISP=(MOD,DELETE),DSN=&&TEMP4
//*
//LOOKAT4  DD   SYSOUT=*
//*
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSING)
//*
//*******************************************************************/
//* STEP : Paso16                                                   */
//*                                                                 */
//* Delete the following temporary datasets:                        */
//*     Exp.D310.Sms.Temp1                                          */
//*     Exp.D310.Sms.Temp2                                          */
//*******************************************************************/
//*
//PASO16    EXEC PGM=IEFBR14,
// COND=((0,NE,PASO7),(0,NE,PASO5),(0,NE,PASO11),(0,NE,PASO12))
//DELTMP1   DD   DSN=EXP.D310.SMS.TEMP1,DISP=(OLD,DELETE)
//DELTMP2   DD   DSN=EXP.D310.SMS.TEMP2,DISP=(OLD,DELETE)
//*
//*******************************************************************/
//* STEP : Paso17                                                   */
//*                                                                 */
//*  Compress the ACS library.                                      */
//*  Because of frequent use of the ACS library, it is better to    */
//*  compress to reclaim the space in this library.                 */
//*******************************************************************/
//*
//PASO17   EXEC PGM=IEBCOPY
//IN       DD   DISP=SHR,DSN=EXP.D310.SMS.ACS
//OUT      DD   DISP=SHR,DSN=EXP.D310.SMS.ACS
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSIN6)
```

```
//*
//*******************************************************************/
//* STEP : Paso18                                                   */
//*                                                                 */
//*  Copy the cumulative ISPLOG    datasets (&&TEMP1) to the spool.  */
//*  Copy the cumulative SYSTSPRT  datasets (&&TEMP2) to the spool.  */
//*  Copy the cumulative SYSPRINT  datasets (&&TEMP3) to the spool.  */
//*                                                                 */
//*  Note : This step will be executed only if the step Paso11 ends  */
//*         with a non-zero RC, in other words, if the new SMS       */
//*         configuration activation doesn't occur, this step is run. */
//*                                                                 */
//*         The reason that prevents activation of new configuration  */
//*         can be searched through those files generated in this    */
//*         step.                                                   */
//*******************************************************************/
//*
//PASO18   EXEC PGM=IDCAMS,COND=(Ø,EQ,PASO11)
//IN1      DD   DISP=(OLD,DELETE),DSN=&&TEMP1
//IN2      DD   DISP=(OLD,DELETE),DSN=&&TEMP2
//IN3      DD   DISP=(OLD,DELETE),DSN=&&TEMP3
//ISPLOGS  DD   SYSOUT=*,DCB=(LRECL=125,RECFM=VA)
//TSOLOGS  DD   SYSOUT=*,DCB=(LRECL=125,RECFM=VA)
//JOBLOGS  DD   SYSOUT=*,DCB=(LRECL=131,RECFM=VA)
//SYSPRINT DD   DUMMY
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN7)
//*
//         PEND
```

PROCEDURE: SMSREST

```
//SMSREST   PROC VER=Ø
//*******************************************************************/
//*     PROCEDURE TO FALL-BACK TO PREVIOUS SMS CONFIGURATIONS       */
//* ----------------------------------------------------------------*/
//* PROCEDURE NAME : SmsRest                                        */
//*                                                                 */
//* This procedure is to return to previous SMS configurations.     */
//* This means that all ACS routines and the SCDS dataset that      */
//* were forming a previous SMS configuration will be recovered upon */
//* successful completion of this procedure.                        */
//*                                                                 */
//* Parameters                                                      */
//* ----------                                                      */
//* VER: Generation index (Back-level SMS configuration)            */
//*******************************************************************/
//*
//*******************************************************************/
//* STEP  : Paso1                                                   */
//*                                                                 */
```

```
//* This step gets one of the previous SMS configuration, requested  */
//* by the VER parameter,from the BKPCDS Gdg dataset and puts it     */
//* into a temporary dataset.                                        */
//*                                                                  */
//* The temporary dataset will have the SCDS dataset (in sequential  */
//* format) and ACS routines of back-level SMS configurations.       */
//*                                                                  */
//* USED DATASETS                                                    */
//* -------------                                                    */
//* Exp.D310.Sms.Bkpcds  (INPUT)  -> Bkup GDG PO dset. (SCDS + ACS)  */
//* Exp.D310.Sms.Temp3   (OUTPUT) -> Temp PO dset.     (SCDS + ACS)  */
//*******************************************************************/
//*
//PASO1    EXEC PGM=IEBCOPY
//IN       DD   DSN=EXP.D310.SMS.BKPCDS(&VER),DISP=SHR
//OUT      DD   DSN=EXP.D310.SMS.TEMP3,DISP=(,CATLG),UNIT=SYSDA,
// DCB=(RECFM=VB,LRECL=4100,BLKSIZE=0),SPACE=(TRK,(1,1,1))
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=131,BLKSIZE=135,RECFM=VA)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSIN6)
//*
//*******************************************************************/
//* STEP  : Paso2                                                   */
//*                                                                 */
//* This step first deletes the SCDS dataset and then imports it    */
//* from one of the generations of the BKPCDS dataset.              */
//*                                                                 */
//* USED DATASETS:                                                  */
//* --------------                                                  */
//* Exp.D310.Sms.Temp3 (INPUT)  -> Backup SCDS/ACS routines dset.   */
//* (Member : SCDS) -------------> Previous SCDS (sequential)       */
//*                                                                 */
//* Sis.D310.Scds      (OUTPUT) -> SCDS dataset that pertains to a  */
//*                                back-level SMS configuration.    */
//*******************************************************************/
//*
//PASO2    EXEC PGM=IDCAMS,COND=(0,NE,PASO1)
//SOURCE   DD   DISP=SHR,DSN=EXP.D310.SMS.TEMP3(SCDS)
//SYSPRINT DD   DSN=&&TEMP4,DISP=(,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=131,BLKSIZE=135,RECFM=VA)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSINC)
//*
//*******************************************************************/
//* STEP  : Paso3                                                   */
//*                                                                 */
//* This step copies the previous step's listing to the spool       */
//* dataset LOOKAT1.                                                */
//*******************************************************************/
//*
//PASO3    EXEC PGM=IDCAMS,COND=(0,NE,PASO1)
//LOOKAT1  DD   SYSOUT=*,DCB=(LRECL=131,BLKSIZE=135,RECFM=VA)
```

```
//INPUT    DD   DSN=&&TEMP4,DISP=(OLD,DELETE)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSINH)
//*
//*******************************************************************/
//* STEP  : Paso4                                                   */
//*                                                                 */
//* This step gets the ACS routines that used to be part of one of  */
//* the previous SMS configurations (back-level).                   */
//*                                                                 */
//* USED DATASETS:                                                  */
//* --------------                                                  */
//* Exp.D31Ø.Sms.Temp3   (INPUT)  -> Backup SCDS/ACS dataset.       */
//*                                                                 */
//* Members:                                                        */
//* ========                                                        */
//* $$DC           -> Back-level ACS routine (For Data Class)       */
//* $$MC           -> Back-level ACS routine (For Management Class) */
//* $$SC           -> Back-level ACS routine (For Storage Class)    */
//* $$SG           -> Back-level ACS routine (For Storage Group)    */
//* FILTLIST       -> Back-level common Filtlist.                   */
//*                                                                 */
//* Exp.D31Ø.Sms.Acs     (OUTPUT) -> The 'Main' ACS dataset.        */
//*                                                                 */
//* Members:                                                        */
//* ========                                                        */
//* The same members as above.                                      */
//*******************************************************************/
//*
//PASO4    EXEC PGM=IDCAMS,COND=(Ø,NE,PASO2)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//I1       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3($$DC)
//O1       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$DC)
//*
//I2       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3($$MC)
//O2       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$MC)
//*
//I3       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3($$SC)
//O3       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$SC)
//*
//I4       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3($$SG)
//O4       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($$SG)
//*
//I5       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3($HISTORY)
//O5       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS($HISTORY)
//*
//I6       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.TEMP3(FILTLIST)
//O6       DD   DISP=SHR,DSN=EXP.D31Ø.SMS.ACS(FILTLIST)
//*
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIND)
//*
```

```
//*********************************************************************/
//* Step : Paso5  (Validate ACS routines)                             */
//*                                                                   */
//* This step validates ACS routines against storage constructs.      */
//*********************************************************************/
//*
//PASO5    EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=(6144K),TIME=(300),
// COND=((0,NE,PASO1),(0,NE,PASO2))
//STEPLIB  DD   DISP=SHR,DSN=SYS1.DGTLLIB
//SYSPROC  DD   DISP=SHR,DSN=SYS1.DGTCLIB
//         DD   DISP=SHR,DSN=ISP.SISPCLIB
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//         DD   DISP=SHR,DSN=SYS1.DGTPLIB
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//         DD   DISP=SHR,DSN=SYS1.DGTMLIB
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU,BLKSIZE=0
//         DD   DISP=SHR,DSN=ISP.SISPSLIB
//         DD   DISP=SHR,DSN=SYS1.DGTSLIB
//ISPTLIB  DD   DISP=NEW,DSN=&&TMP,UNIT=SYSDA,BLKSIZE=0,
// SPACE=(TRK,(1,1,1)),DCB=(ISP.SISPTENU)
//         DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//         DD   DISP=SHR,DSN=ISP.SISPTENU
//         DD   DISP=SHR,DSN=SYS1.DGTTLIB
//ISPTABL  DD   DISP=SHR,DSN=INEX004.ISPF.ISPPROF
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1)),UNIT=SYSDA
//SYSUDUMP DD   SYSOUT=*
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS),UNIT=SYSDA,
// SPACE=(TRK,(1,1)),DCB=(LRECL=125,BLKSIZE=129,RECFM=VA)
//LOOKAT2  DD   SYSOUT=*,DCB=(LRECL=81,RECFM=F)
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSIN9)
//*
//*********************************************************************/
//* STEP : Paso6                                                      */
//*                                                                   */
//* This step copies the validate listing to the spool dataset        */
//* 'VALIDATE'.                                                       */
//*********************************************************************/
//*
//PASO6    EXEC PGM=IDCAMS,COND=(0,NE,PASO2)
//INPUT    DD   DSN=EXP.D310.SMS.VALIDATE.LISTING,DISP=SHR
//VALIDATE DD   SYSOUT=*,DCB=(LRECL=133,RECFM=FA)
//SYSPRINT DD   DSN=&&TEMP3,DISP=(MOD,PASS)
//SYSIN    DD   DISP=SHR,DSN=EXP.D310.SMS.SOURCE(SYSIN2)
//*
//*********************************************************************/
//* STEP : Paso7   (SCDS -----> ACDS)                                 */
//*                                                                   */
//* This step activates the one of the back-level SMS configurations. */
```

```
//*                                                                   */
//* The REXX SmsREXX2 is called from this step and it isues the       */
//* following SETSMS command in the SYStem LOG. The REXX will also    */
//* check the response messages against this command.                 */
//*                                                                   */
//* Note: This step is executed only if the previous SCDS dataset     */
//*       is imported successfully from the BKPCDS dataset.          */
//*                                                                   */
//*     COMMAND          : 'Setsms Scds(Sis.D31Ø.Scds)'               */
//*     EXPECTED MESSAGE :                                           */
//*     IGDØØ8I New configuration activated from Scds Sis.D31Ø.Scds   */
//*********************************************************************/
//*
//PASO7    EXEC PGM=IKJEFTØ1,DYNAMNBR=3Ø,REGION=4Ø96K,
// COND=((Ø,NE,PASO1),(Ø,NE,PASO2),(Ø,NE,PASO5))
//SYSEXEC  DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE
//ISPPLIB  DD   DISP=SHR,DSN=ISP.SISPPENU
//ISPMLIB  DD   DISP=SHR,DSN=ISP.SISPMENU
//ISPSLIB  DD   DISP=SHR,DSN=ISP.SISPSENU
//ISPTLIB  DD   DISP=SHR,DSN=ISP.SISPTENU
//ISPPROF  DD   DISP=(NEW,DELETE,DELETE),DSN=&&PROF,UNIT=SYSDA,
// DCB=(ISP.SISPTENU),SPACE=(TRK,(1,1,1))
//LOOKAT3  DD   SYSOUT=*,DCB=(LRECL=81,RECFM=F)
//SYSUDUMP DD   SYSOUT=*
//ISPLOG   DD   DSN=&&TEMP1,DISP=(MOD,PASS)
//SYSTSPRT DD   DSN=&&TEMP2,DISP=(MOD,PASS)
//SYSTSIN  DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSINE)
//*
//*********************************************************************/
//* STEP : Paso8                                                      */
//*                                                                   */
//* Delete the following temporary dataset:                           */
//* Exp.D31Ø.Sms.Temp3                                                */
//*********************************************************************/
//*
//PASO8    EXEC PGM=IEFBR14
//DELTMP   DD   DSN=EXP.D31Ø.SMS.TEMP3,DISP=(OLD,DELETE)
//*
//*********************************************************************/
//* STEP : Paso9                                                      */
//*                                                                   */
//*  Copy the cumulative ISPLOG    datasets (&&TEMP1) to the spool.   */
//*  Copy the cumulative SYSTSPRT  datasets (&&TEMP2) to the spool.   */
//*  Copy the cumulative SYSPRINT  datasets (&&TEMP3) to the spool.   */
//*                                                                   */
//*  Note : This step will be executed only if the step Paso7 ends    */
//*         with a non-zero RC.                                       */
//*********************************************************************/
//*
//PASO9    EXEC PGM=IDCAMS,COND=(Ø,EQ,PASO7)
//IN1      DD   DISP=(OLD,DELETE),DSN=&&TEMP1
```

```
//IN2      DD   DISP=(OLD,DELETE),DSN=&&TEMP2
//IN3      DD   DISP=(OLD,DELETE),DSN=&&TEMP3
//ISPLOGS  DD   SYSOUT=*,DCB=(LRECL=125,RECFM=VA)
//TSOLOGS  DD   SYSOUT=*,DCB=(LRECL=125,RECFM=VA)
//JOBLOGS  DD   SYSOUT=*,DCB=(LRECL=131,RECFM=VA)
//SYSPRINT DD   DUMMY
//SYSIN    DD   DISP=SHR,DSN=EXP.D31Ø.SMS.SOURCE(SYSIN7)
//*
//         PEND
```

REXX: SMSREXX1

```
/*REXX*/
/* ------------------------------------------------------------------ */
/* SmsREXX1                                                           */
/*                                                                    */
/* This REXX is called from the Paso11 of the SMSNEW procedure and    */
/* is intended to issue one serie of SETSMS commands through SDSF.    */
/*                                                                    */
/* REXX PROCEDURES CALLED                                             */
/* ----------------------                                             */
/* SmsREXX3  : It is an external procedure. It formats the Isfout     */
/*             dataset by using an edit macro (SmsMacro).             */
/*                                                                    */
/* Handle1   : It is an internal procedure. In case of not getting an */
/*             expected SMS message against a 'SETSMS' command,       */
/*             SYStem LOG is checked to look for any other SMS        */
/*             messages (most likely error messages).                 */
/*                                                                    */
/* Handle2   : It is an internal procedure. It creates the Sysout     */
/*             file LOOKAT3.                                          */
/*                                                                    */
/* Issue_proc: It is an internal procedure. It gets an SETSMS command */
/*             as a parameter and issues it by calling SDSF rogram.   */
/* ------------------------------------------------------------------ */
'Prof nopref'
/* ------------------------------------------------------------------ */
/* Create a new data stack that will be used for writing information  */
/* into the LOOKAT3 dataset.                                        */
/* ------------------------------------------------------------------ */
'Newstack'
/* ------------------------------------------------------------------ */
/* The variable 'Dsn' is the ISFOUT dataset to be formatted by the    */
/* edit macro.                                                        */
/* ------------------------------------------------------------------ */
Dsn  = Exp.D31Ø.Sms.Sdsf.Temp
Rr   = ' '
MyRC = Ø
/* ------------------------------------------------------------------ */
/* Send the following SMSACT command and get the response:            */
/* Setsms Acds(Sis.D31Ø.Acds)                                         */
```

```
/* This command is intended for setting the ACDS dataset correctly.   */
/* ------------------------------------------------------------------ */
Komut      = '/SETSMS ACDS(SIS.D31Ø.ACDS)'
Call Issue_proc Komut
Sonuc = Record.1

If Index(Sonuc,'IGDØØ9I') ¬=Ø Then
    Do
       Mid1 ='IGDØØ9I'
       Look1Ø= 'ACDS dataset has been set. The related message is: '
       Look11= '-------------------------------------------------'
       Look12= Substr(Sonuc,Index(Sonuc,'IGD'))
    End

    Else                              /* Handle the error condition. */
      Do
        /* ---------------------------------------------------------- */
        /*  Call the Handle1 procedure in case we do not get the      */
        /*  expected message (here IGDØØ9I) against SETSMS command.   */
        /*                                                            */
        /*  Messages other than IGDØØ9I will not appear on the Message*/
        /*  Line. Because of that, it is necessary to check out       */
        /*  other messages - most likely error messages - in the      */
        /*  SYSLOG. Handle1 procedure will issue the 'LOG' command    */
        /*  via SDSF to scan the most recent part of the Syslog for   */
        /*  other IGD* messages.                                      */
        /* ---------------------------------------------------------- */
        Call Handle1
        Look1Ø= 'ACDS dataset has NOT been set. The related message is:'
        Look11= '-----------------------------------------------------'
        Look12= Substr(Sonuc,Index(Sonuc,'IGD'))
        Myrc = Myrc + 21
        /* ---------------------------------------------------------- */
        /*  Call the Handle2 procedure to write all queued messages   */
        /*  to the LOOKAT3 file. Then Exit with the appropriate       */
        /*  Return Code (MyRc).                                       */
        /* ---------------------------------------------------------- */
        Call Handle2
        Exit MyRc
      End
/* ------------------------------------------------------------------ */
/* Send the following SMSACT command and get the response:            */
/* Setsms Saveacds(Sis.D31Ø.Acds.Spare)                               */
/*                                                                    */
/* This command is for creating a backup copy of the active/current   */
/* SMS configuration(Acds) before activating a new one.               */
/* ------------------------------------------------------------------ */
Komut      = '/SETSMS SAVEACDS(SIS.D31Ø.ACDS.SPARE)'
Call Issue_proc Komut
Sonuc = Record.1
```

```
If Index(Sonuc,'IGD014I') ¬=0 Then
    Do
      Mid2 ='IGD014I'
      Look20= 'Active SMS configuration has been backed-up. The msg is:'
      Look21= '-------------------------------------------------------'
      Look22= Substr(Sonuc,Index(Sonuc,'IGD'))
    End

    Else                              /* Handle the error condition. */
      Do
        Call Handle1
        Look20= 'Backup CDS can not be created. The related message is:'
        Look21= '--------------------------'
        Look22= Substr(Sonuc,Index(Sonuc,'IGD'))
        Myrc = Myrc + 31
        Call Handle2
        Exit MyRc
      End

/* ---------------------------------------------------------------- */
/* We have successfully completed first two SETSMS commands. Now it   */
/* is time to activate the new SMS configuration.                     */
/*                                                                    */
/* Send the following SMSACT command and get the response:            */
/* Setsms SCDS (Sis.D310.Scds)                                        */
/*                                                                    */
/* On a successful completion of this command, ACDS dataset will be   */
/* updated with the contents of the SCDS dataset.                     */
/* ---------------------------------------------------------------- */
Komut      = '/SETSMS SCDS(SIS.D310.SCDS)'
Call Issue_proc Komut
Sonuc = Record.1

If Index(Sonuc,'IGD008I') ¬=0 Then
    Do
      Look30= 'New SMS configuration activated. The reletad message is:'
      Look31= '-----------------------------------------------'
      Look32= Substr(Sonuc,Index(Sonuc,'IGD'))
    End

    Else                               /* Handle the error condition. */
      Do
        Call Handle1
        Look30= 'New SMS config. NOT activated. The related message is:'
        Look31= '-------------------------------------------------'
        Look32= Substr(Sonuc,Index(Sonuc,'IGD'))
        Myrc = Myrc + 41
        Call Handle2
        Exit MyRc
      End
```

```
/* ---------------------------------------------------------------- */
/* We have successfully completed all three SETSMS commands, So we  */
/* now have a new SMS configuration active. Set the RC to Ø and exit. */
/* ---------------------------------------------------------------- */
MyRc=Ø
Call Handle2
EXIT MyRc   /* End-of-Main-REXX */
/* ---------------------------------------------------------------- */
ISSUE_PROC:
/* ---------------------------------------------------------------- */
/*  Procedure : Issue_Proc  (Inline procedure)                      */
/*  Function  : Issue SMS commands from SDSF command line.          */
/* ---------------------------------------------------------------- */
Arg Kom    /* Get the SMS command to be issued. */
/* ---------------------------------------------------------------- */
/* ( Clear residual values from the stem variable.)                 */
/* ---------------------------------------------------------------- */
DROP Record.
/* ---------------------------------------------------------------- */
/* Allocate SDSF-output dataset (ISFOUT) & Sysin file (ISFIN)       */
/* ---------------------------------------------------------------- */
'Alloc Da('Dsn') Fi(Isfout) Reuse New Catalog Tracks Space(2,1) Release,
 Recfm (F,B,A) Lrecl(133) Blksize(Ø)'
'Alloc Fi(Isfin) Space(1,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse'
/* ---------------------------------------------------------------- */
/*  SET CONSOLE SMSACT                                              */
/*  Extended console name used by SDSF will be 'SMSACT'.            */
/*  'Setsms' commands and the replies are issued under this name in */
/*  the console.                                                    */
/*                                                                  */
/*  SET DELAY 3  (Delay Interval)                                   */
/*  Allows SDSF to wait and display messages in response to slash (/) */
/*  command on the Message Line.                                    */
/*                                                                  */
/*  Messages issued within the delay interval are displayed on the  */
/*  'Message line' of the panel you are ON. The delay interval is the */
/*  maximum amount of time SDSF will wait for messages before       */
/*  displaying them on the message line.                            */
/*                                                                  */
/*  Here it is set to three seconds. You have to adjust this value  */
/*  according to your system. Note that if you keep this value too  */
/*  low, you may receive the message 'NO RESPONSE RECEIVED' on the  */
/*  'Message Line' of the SDSF panel. This situation prevents proper */
/*  execution of the REXX.                                          */
/* ---------------------------------------------------------------- */
Rec_Isfin.1 = 'SET CONSOLE SMSACT'
Rec_Isfin.2 = 'SET DELAY 3'
Rec_Isfin.3 = Kom
Rec_Isfin.Ø = 3
/* ---------------------------------------------------------------- */
/*  Build Isfin file. (Sysin control statements for SDSF program.)  */
```

```
/* ---------------------------------------------------------------- */
'Execio * Diskw Isfin (Finis Stem Rec_Isfin.'
/* ---------------------------------------------------------------- */
/* Set the stem variable Rec_Isfin to its uninitialized state.      */
/* ( Clear residual values from the stem variable.)                 */
/* ---------------------------------------------------------------- */
DROP Rec_Isfin.
/* ---------------------------------------------------------------- */
/* Call the SDSF program.                                           */
/* ---------------------------------------------------------------- */
'SDSF'
RC2 = Rc
'Free File(Isfin)'
'Free File(Isfout)'
/* ---------------------------------------------------------------- */
/* SDSF Return code control.                                        */
/* ---------------------------------------------------------------- */
If Rc2 ¬= Ø Then Do
                  Say 'Error on calling SDSF program... Exiting....'
                  'Delete' Dsn                /* Delete Isfout dset */
                  Exit 51
                End
/* ---------------------------------------------------------------- */
/* Call the REXX SmsREXX3. This REXX will format the SDSF output    */
/* dataset (Isfout) by using an Edit Macro, which is SmsMacro.      */
/*                                                                  */
/* If we need to give an 'Automatic Reply', the REXX SmsREXX3 is not */
/* called. In this case we exit from this procedure (Issue_Proc).   */
/* Because we will not interpret the response of the AutoReply and we */
/* will be assuming that 'Automatic Reply' went OK.                 */
/* ---------------------------------------------------------------- */
If Substr(Kom,1,2) = '/R' Then
      Do
        'Delete' Dsn                          /* Delete Isfout DSET */
        Return
      End
'Ispstart Cmd(SmsREXX3 'Dsn' 'Mid1' 'Mid2' 'Kom') Batscrw(132),
                  Batscrd(27) Bredimax(2) Bdispmax(1Ø)'
/* ---------------------------------------------------------------- */
/* It is time to read the just-formatted-ISFOUT dataset.            */
/* Having edited the ISFOUT dataset by the edit macro (called from  */
/* SmsREXX3), we must now have only some IGD* messages left in the  */
/* Isfout dataset.                                                  */
/* ---------------------------------------------------------------- */
'Alloc Da('Dsn') Fi(Isfout) Shr Reuse'
'Execio * Diskr Isfout (Stem Record. Finis'
/* ---------------------------------------------------------------- */
/* Having read the ISFOUT dataset, we now can free and delete it.   */
/* ---------------------------------------------------------------- */
'Free File(Isfout)'
```

```
'Delete' Dsn
RETURN         /* End-of-Issue_Proc */

HANDLE1:
/* ---------------------------------------------------------------- */
/* Procedure : Handle1     (Inline procedure)                       */
/* This procedure is used to check the Syslog for any SMS error     */
/* messages. To do that, from SDSF a 'LOG' command is issued and the  */
/* most recent part of the Syslog output is scanned.                */
/*                                                                  */
/* We will most likely get the following two messages due to the    */
/* non-existence of the SMS Control datasets:                       */
/*                                                                  */
/* IGD056I {SCDS3ACDS3COMMDS} DSname NOT FOUND                      */
/* IGD040D UNABLE TO COMPLETE CONFIGURATION REQUEST: text - REPLY 'U' */
/*         TO RETRY OR 'C' TO PURGE REQUEST                         */
/*                                                                  */
/* The second message is an outstanding WTOR message that is asking   */
/* for a reply. In the REXX, we reply 'C' to this message.          */
/* ---------------------------------------------------------------- */
Komut      = 'LOG'
Call Issue_proc Komut
Sonuc  = Record.1   /* IGD056I message. */
Sonuc2 = Record.2   /* IGD040D message. */
/* ---------------------------------------------------------------- */
/* Check the 'IGD056I' message.                                     */
/* ---------------------------------------------------------------- */
If Index(Sonuc,'IGD056I') ¬=0 Then
                Look33= '*** Please verify that CDS DSET exists.'
          Else  Look33= '*** Other type of error. Please check SYSLOG.'

/* ---------------------------------------------------------------- */
/* Check the 'IGD040D' message.                                     */
/* ---------------------------------------------------------------- */
If Index(Sonuc2,'IGD040D') ª=0 Then Do
                Myrc = 1
                Look34= '*** Automatic reply to IGD040D in progress.'

             /* ---------------------------------------------------*/
             /* Find the message number of the WTOR and generate the  */
             /* the reply message text, then reply 'C' to that     */
             /* message to cancel the outstanding SMS request.     */
             /* ---------------------------------------------------*/
                 FIR=Substr(Sonuc2,(Index(Sonuc2,'IGD040D')-3),2)
                 Komut      = '/R 'fir',C'
                 Call Issue_proc Komut
                 Look35= '*** Automatic reply given, check Syslog.'
                 End
          Else   Look34= '*** Other type of error. Please check SYSLOG.'
RETURN         /* End-of-Handle1 procedure */
```

```
HANDLE2:
/* ---------------------------------------------------------------- */
/* Procedure : Handle2     (Inline procedure)                       */
/* Function  :                                                      */
/* -----------                                                      */
/* Write all queued messages, together with SETSMS command replies  */
/* to the LOOKAT3 file. Then delete the data stack.                 */
/* ---------------------------------------------------------------- */
Queue Look1Ø
Queue Look11
Queue Look12
Queue ' '
If Look2Ø ¬='LOOK2Ø" Then Queue Look2Ø
If Look21 ¬='LOOK21" Then Queue Look21
If Look22 ¬='LOOK22" Then Queue Look22
Queue ' '
If Look3Ø ¬='LOOK3Ø" Then Queue Look3Ø
If Look31 ¬='LOOK31" Then Queue Look31
If Look32 ¬='LOOK32" Then Queue Look32
Queue ' '
If Look33 ¬='LOOK33" Then Queue Look33
Queue ' '
If Look34 ¬='LOOK34" Then Queue Look34
Queue ' '
If Look35 ¬='LOOK35" Then Queue Look35
/* ---------------------------------------------------------------- */
/* Queue a null line at the bottom of the stack to indicate the end */
/* of the information. Write the data stack to LOOKAT3. Then delete */
/* the data stack.                                                  */
/* ---------------------------------------------------------------- */
Queue ''
'Execio * Diskw Lookat3 (Finis'
'Delstack'
RETURN         /* End-of-Handle2 procedure */
```

SMSREXX2

```
/*REXX*/
/* ---------------------------------------------------------------- */
/* SMSREXX2                                                         */
/*                                                                  */
/* This REXX is called by the procedure SMSREST, which is 'FALL-BACK  */
/* TO PREVIOUS SMS CONFIGURATIONS' process.                         */
/*                                                                  */
/* When one of the previous version of SCDS dataset is restored by  */
/* the SMSREST procedure, the REXX SmsREXX2 activates that SCDS by  */
/* issuing the command 'SETSMS SCDS()' from the SYStem LOG.         */
/*                                                                  */
/* REXX PROCEDURES CALLED                                           */
/* ----------------------                                           */
```

```
/* SmsREXX3                                                          */
/* Handle1                                                           */
/* Handle2                                                           */
/* Issue_proc                                                        */
/* ----------------------------------------------------------------- */
'Prof nopref'
'Newstack'
Dsn  = Exp.D31Ø.Sms.Sdsf.Temp
Rr   = ' '
MyRC = Ø
/* ----------------------------------------------------------------- */
/* Send the following SMSACT command and get the response:           */
/* Setsms Scds(Sis.D31Ø.Scds)                                        */
/* ----------------------------------------------------------------- */
Komut      = '/SETSMS SCDS(SIS.D31Ø.SCDS)'
Call Issue_proc Komut
Sonuc = Record.1
If Index(Sonuc,'IGDØØ8I') ª=Ø Then
    Do
      Look1Ø= 'Previous SMS config. recovered. The related message is:'
      Look11= '-------------------------------------------------------'
      Look12= Substr(Sonuc,Index(Sonuc,'IGD'))
    End

    Else                        /* Handle the error condition. */
      Do
        Call Handle1
        Look1Ø= 'New SMS config. NOT activated. The related message is:'
        Look11= '-------------------------------------------------------'
        Look12= Substr(Sonuc,Index(Sonuc,'IGD'))
        Myrc = Myrc + 41
        Call Handle2
        Exit MyRc
      End
/* ----------------------------------------------------------------- */
/* We have successfully completed the SETSMS command. Now we have a  */
/* back level SMS configuration active. Set the RC to Ø and exit.    */
/* ----------------------------------------------------------------- */
MyRc=Ø
Call Handle2
EXIT MyRc   /* End-of-main-REXX */
/* ----------------------------------------------------------------- */

ISSUE_PROC:
/* ----------------------------------------------------------------- */
/* In this procedure, SMS commands are issued from the SDSF command  */
/* line.                                                             */
/* ----------------------------------------------------------------- */
Arg Kom
DROP Record.
```

```
/* ---------------------------------------------------------------- */
/* Allocate SDSF-output dataset (ISFOUT) and Sysin file (ISFIN)     */
/* ---------------------------------------------------------------- */
'Alloc Da('Dsn') Fi(Isfout) Reuse New Catalog Tracks Space(2,1) Release,
 Recfm (F,B,A) Lrecl(133) Blksize(Ø)'
'Alloc Fi(Isfin) Space(1,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse'

Rec_Isfin.1 = 'SET CONSOLE SMSACT'
Rec_Isfin.2 = 'SET DELAY 3'
Rec_Isfin.3 = Kom
Rec_Isfin.Ø = 3
'Execio * Diskw Isfin (Finis Stem Rec_Isfin.'
DROP Rec_Isfin.
'SDSF'
RC2 = Rc
'Free File(Isfin)'
'Free File(Isfout)'
If Rc2 ¬= Ø Then Do
                  Say 'Error on calling SDSF program... Exiting....'
                  'Delete' Dsn                /* Delete Isfout dset */
                  Exit 51
                End
If Substr(Kom,1,2) = '/R' Then
      Do
        'Delete' Dsn
        Return
      End
'Ispstart Cmd(SmsREXX3 'Dsn' 'Mid1' 'Mid2' 'Kom') Batscrw(132),
                  Batscrd(27) Bredimax(2) Bdispmax(1Ø)'
'Alloc Da('Dsn') Fi(Isfout) Shr Reuse'
'Execio * Diskr Isfout (Stem Record. Finis'
'Free File(Isfout)'
'Delete' Dsn
RETURN              /* End-of-issue_Proc */

HANDLE1:
Komut      = 'LOG'
Call Issue_proc Komut
Sonuc  = Record.1   /* IGDØ56I message. */
Sonuc2 = Record.2   /* IGDØ4ØD message. */
/* ---------------------------------------------------------------- */
/* Check the 'IGDØ56I' message.                                     */
/* ---------------------------------------------------------------- */
If Index(Sonuc,'IGDØ56I') ¬=Ø Then
                Look13= '*** Please verify that CDS DSET exists.'
          Else  Look13= '*** Other type of error. Please check SYSLOG.'

/* ---------------------------------------------------------------- */
/* Check the 'IGDØ4ØD' message.                                     */
/* ---------------------------------------------------------------- */
```

```
If Index(Sonuc2,'IGD040D') ¬=0 Then Do
                 Myrc = 1
                 Look14= '*** Automatic reply to IGD040D in progress..'
                 FIR=Substr(Sonuc2,(Index(Sonuc2,'IGD040D')-3),2)
                 Komut      = '/R 'fir',C'
                 Call Issue_proc Komut
                 Look15= '*** Automatic reply given, check Syslog.'
                 End
          Else   Look14= '*** Other type of error. Please check SYSLOG.'
RETURN         /* End-of-Handle1 procedure */

HANDLE2:
If Look10 ¬='LOOK10" Then Queue Look10
If Look11 ¬='LOOK11" Then Queue Look11
If Look12 ¬='LOOK12" Then Queue Look12
Queue ' '
If Look13 ¬='LOOK13" Then Queue Look13
Queue ' '
If Look14 ¬='LOOK14" Then Queue Look14
Queue ' '
If Look15 ¬='LOOK15" Then Queue Look15
Queue ''
'Execio * Diskw Lookat3 (Finis'
'Delstack'
RETURN         /* End-of-Handle2 procedure */
```

SMSREXX3

```
/*REXX*/
/* ------------------------------------------------------------------- */
/* SMSREXX3 (external procedure)                                       */
/*                                                                     */
/* This REXX is used to call an edit macro (SmsMacro) to select only  */
/* SMS messages in the Isfout dataset which is created in SmsREXX1   */
/* or SmsREXX2. Isfout contains either most recent part of the Syslog */
/* or responses associated with 'SETSMS' commands issued from the     */
/* SDSF.                                                               */
/*                                                                     */
/* PARAMETERS                                                          */
/* ----------                                                          */
/* These parameters have to be passed to the edit macro by using ISPF */
/* VGET and VPUT services.                                             */
/*                                                                     */
/* DSET    : Dataset to be updated by edit macro.                      */
/* MSGID1  : It can be either 'IGD009I' or the string 'MID1'.          */
/* MSGID2  : It can be either 'IGD014I' or the string 'MID2'.          */
/* COMMAND : Command of whose output in the Spool will be interpreted */
/*           by the edit macro.                                        */
/*                                                                     */
/* ADDITIONAL NOTES ON THIS REXX:                                      */
```

```
/* There is a possibility that when we issue the 'LOG' command from   */
/* Sdsf, we can encounter those two messages. In some way, we should  */
/* not care about messages of the 'successfully issued SMS commands'  */
/* and exclude them from the ISFOUT dataset. This way we will leave   */
/* only unprocessed IGD* messages.                                    */
/*                                                                    */
/* For this reason, we remove those messages from the Isfout dataset  */
/* by using the edit macro (SmsMacro) which is called from this REXX. */
/*                                                                    */
/* eg If the first SMS command is issued successfully, to be able     */
/* to pinpoint properly the result of the next SMS command, we are    */
/* excluding the expected response message 'IGDØØ9I' of the first     */
/* command from the Isfout dataset ('LOG' command output).            */
/* ------------------------------------------------------------------ */
Arg Dset Msgid1 Msgid2 Command

'Ispexec Vput (Msgid1,Msgid2,Command) Profile'

/* ------------------------------------------------------------------ */
/* Call the edit macro and extract the IGD* messages.                 */
/* ------------------------------------------------------------------ */
'Ispexec Edit Dataset(''Dset'') Macro(SmsMacro)'
 If Rc ¬= Ø  Then Do
                   Say 'Error on edit macro SmsMacro. Please check it.'
                   Return
                 End
EXIT Ø
```

REXX: SMSREXX4

```
/*REXX*/
/* ------------------------------------------------------------------ */
/* SmsREXX4                                                           */
/*                                                                    */
/* Interpret the Translate or Validate listing dataset and set the    */
/* Return code.                                                       */
/*                                                                    */
/* Parameters:                                                        */
/* -----------                                                        */
/* Tra_or_val : It can be a value of 'TRA' or 'VAL'.   (Input)        */
/* ------------------------------------------------------------------ */
Arg Tra_or_val
'Prof nopref'
Flag = '1'    /* If it is still '1' at the end, the process is OK.    */
Ret  = Ø      /* Default Return code.                                 */

MsgX = ' translation is successful.'
MsgY = ' Validation  is successful.'
MsgV = 'All ACS Objects have been successfully created and stored in '
MsgW = 'the specified SCDS. Validation of the ACS Routines against
```

```
MsgW = 'storage constructs is successful.'
/* ---------------------------------------------------------------- */
/* Create a new data stack that will be used for writing information  */
/* to the LOOKATx datasets. (Lookat1 for Tranlate, Lookat2 Validate) */
/* ---------------------------------------------------------------- */
'NewStack'
/* ---------------------------------------------------------------- */
/* Check out the listing if translation or validation process is OK.  */
/* ---------------------------------------------------------------- */
If Tra_or_Val = 'TRANS' Then Do
                             Call Read Exp.D310.Sms.Dc.Listing
                             Call Read Exp.D310.Sms.Mc.Listing
                             Call Read Exp.D310.Sms.Sc.Listing
                             Call Read Exp.D310.Sms.Sg.Listing
                             End
If Tra_or_Val = 'VALID' Then Call Read Exp.D310.Sms.Validate.Listing
/* ---------------------------------------------------------------- */
/* Write all Trans./Valid. result messages to the LOOKATx datasets   */
/* and set the RC.                                                   */
/* ---------------------------------------------------------------- */
If Flag='1' Then If Class ¬='VA' Then Queue MsgV
                                 Else Queue MsgW
            Else If Class ¬='VA' Then Ret= 11
                                 Else Ret= 12
/* ---------------------------------------------------------------- */
/* Queue a null line at the bottom of the stack to indicate the end  */
/* of the information. Write the data stack to the dataset. Then     */
/* delete the data stack.                                            */
/* ---------------------------------------------------------------- */
Queue ''
If Tra_or_Val = 'TRANS' Then 'Execio * DiskW 'Lookat1' (Finis'
                        Else 'Execio * DiskW 'Lookat2' (Finis'
'DelStack'
EXIT Ret  /* End of main REXX */

READ:
/* ---------------------------------------------------------------- */
/*  Procedure : Read (Internal procedure)                            */
/*  Function  : Read the related record of the Translate or Validate  */
/*              Listing.                                             */
/* Parameters:                                                       */
/* -----------                                                       */
/* DSN        : Translation listing or Validation listing.  (Input)   */
/* ---------------------------------------------------------------- */
Arg Dsn

Class = Substr(Dsn,14,2)
If Class = 'VA' Then Txt = 'VALIDATION RESULT:'
                Else Txt = 'TRANSLATION RETURN CODE:'

'Alloc Da('Dsn') Fi(File) Shr Reuse'
```

```
'Execio * Diskr File (Stem Marta. Finis'
'Free  Fi(File)'
/* ---------------------------------------------------------------- */
/* Find the line which contains the return code of the translation  */
/* or the Validation Result. Martapv variable will contain that line. */
/* In addition, any SMS error message in Translation or Validation  */
/* listing will be extracted and put into the LOOKATx spool outputs.  */
/*                                                                  */
/* The string to be searched in the listings:                       */
/* ========================================                         */
/* For Translation listing: 'TRANSLATION RETURN CODE:  0000'        */
/* For Validation  listing: 'VALIDATION RESULT: VALIDATION SUCCESSFUL */
/* ---------------------------------------------------------------- */
Do i = 1 to Marta.0
   If Index(Marta.i,Txt)   ¬= 0 Then Martapv = Delstr(Marta.i,44)
   If Index(Marta.i,'IGD') ¬= 0 Then
                                  Do
                                    MsgZ=
Substr(Marta.i,Index(Marta.i,'IGD'))
                                    Push MsgZ
                                  End
End

If Class ¬= 'VA' Then Do
                         Msg  = Insert(Class,MsgX,0)
                         Srch = '0000'
                         Chk = 'Check the TRANSLATE output.'
                      End
                 Else Do
                         Msg  = Insert('  ',MsgY,0)
                         Srch = 'VALIDATION SUCCESSFUL'
                         Chk = 'Check the VALIDATE  output.'
                      End
If Index(Martapv,Srch) = 0 Then Do
                                 Msg = Insert('NOT ',Msg,18)
                                 Flag = '0'
                                 Msg=Insert(Chk,Msg,34)
                                End
Queue Msg
RETURN /* End-of-procedure-READ */
```

SMSREXX5

```
/*REXX*/
/* ---------------------------------------------------------------- */
/* SMSREXX5                                                         */
/*                                                                  */
/* Save the current SMS configuration as a new generation in the    */
/* BKPCDS GDG dataset.                                              */
/* ---------------------------------------------------------------- */
```

```
'Prof nopref'
Dat = Date('J')              /* Today's date (in Julian format)  */

Say ' '
Say '*****************************************************************'
Say '**** SAVING THE ACS ROUTINES AND THE ACTIVE SCDS INTO BKPCDS  ****'
Say '*****************************************************************'
Say ' '

/* ------------------------------------------------------------------ */
/*  Extract the name and the creation date of the latest generation   */
/*  of the BKPCDS dataset.                                            */
/* ------------------------------------------------------------------ */
Say 'The latest generation of the BKPCDS dataset was:'
Call Find_Latest
Prt= Delstr(Result,3,1)
Say 'Back-up date was = ' DATE('N',Prt,'J')
Say ''
Say ' ---------------------------------------'

If Prt = Dat Then Comp_sysin = ' COPY OUTDD=OUT2,INDD=((IN,R))'
             Else Comp_sysin = ' COPY OUTDD=OUT1,INDD=((IN,R))'

'Alloc Fi(Sysout) Space(2,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse'
'Alloc Fi(Sysprint) Space(2,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø)
Reuse'
'Alloc Fi(Sysin)  Space(1,1) Track Lrecl(8Ø) Recfm(f) Blksize(8Ø) Reuse'
'Newstack'     /* Create a new data stack for Sysin file.      */
Queue Comp_sysin
Queue ''
'Execio 1 Diskw Sysin (Finis'
'Delstack'     /* Delete the data stack for Sysin file.      */
/* ---------------------------------------*/
/* Call Iebcopy utility.                  */
/* ---------------------------------------*/
'Call 'Sys1.Linklib(Iebcopy)''
RcØ = Rc
If RcØ = Ø Then  /* IF-1 */
   DO
    Ddinfo1= Listdsi('Out1' 'File')
    Dset1  = SYSDSNAME
    Ddinfo2= Listdsi('Out2' 'File')
    Dset2  = SYSDSNAME
    If Prt = Dat Then
        DO
         Say 'NOTE:'
         Say 'The new generation of the GDG dataset ('Dset1') '
         Say 'was not created, since the latest generation was '
         Say 'today! Can only be created one generation a day to '
         Say 'prevent the BKPCDS dataset from rolling out all its past '
         Say 'generations. On the other hand, current ACS routines and '
```

```
          Say 'the current SCDS have been written into the latest '
          Say ''generation of BKPCDS, replacing the previous copy.'
          Say ' ------------------------------------------------------'
          Say ' '
          /* ---------------------------------------------------------- */
          /* Delete the new generation of BKPCDS, since it will not be */
          /* used. (It was allocated before in the step Paso14 of the  */
          /* SMSNEW.)                                                   */
          /* ---------------------------------------------------------- */
          'Delete' Dset1
          If Rc <> Ø Then Say 'Please delete dataset' Dset1 'manually...'
          Ret = 61

        END
        ELSE
        DO
    Say 'ACS routnes and SCDS dataset has been successfully backed-up.'
    Say '(to the generation' Dset1".)'
    Say '-------------------------------------------------------------'
    Say ''
    'Free File(Out2)'
        Ret = Ø
        END
   END  /* IF-1 */

'Free File(Out1)'
'Free File(In)'
'Free File(Sysin)'
Say 'The latest generation of the BKPCDS dataset is NOW:'
Call Find_Latest
PrT= Delstr(Result,3,1)
Say 'Back-up date is = ' DATE('N',Prt,'J')
Exit Ret            /* End-of-REXX SmsREXX5    */

FIND_LATEST:
/* ---------------------------------------------------------------- */
/*  Procedure : Find_Latest (Inline procedure)                      */
/*  Function  : Find the latest version of the BKPCDS GDG dataset.  */
/* ---------------------------------------------------------------- */
X = Outtrap('Var1.')
'Listc Ent('Exp.D31Ø.Sms.Bkpcds') GDG All'
          /*-------------------------------------------------------*/
          /* The last line of the command output will have the latest */
          /* GDG generation of the BKPCDS dataset.                 */
          /*-------------------------------------------------------*/
          i = Var1.Ø
          Absolute_name = Substr(Var1.i,17)
         Say  Absolute_name
          /*-------------------------------*/
          /* Now, find the creation date.  */
          /*-------------------------------*/
```

```
          Y = Outtrap('Var2.')
          'Listc Ent(''Absolute_name'') History'
          Do k = Var2.Ø to 1 by -1
             P= Index(Var2.k,'CREATION')
             If P <> Ø Then Leave
          End
RETURN Substr(Var2.k,p+18)       /* End_of_proc_Find_Latest */
```

SMSMACRO

```
Address Ispexec
'Isredit Macro'
/* ---------------------------------------------------------------- */
/* Edit macro name : SmsMacro                                       */
/* Called from     : SmsREXX3  (REXX program)                       */
/* Function        : To make necessary changes on the ISFOUT        */
/*                   datset which is 'Exp.D31Ø.Sms.Sdsf.Temp'.      */
/* Variable Msgid1 can be either 'IGDØØ9I' or the string 'MID1'.    */
/* Variable Msgid2 can be either 'IGDØ14I' or the string 'MID2'.    */
/* ---------------------------------------------------------------- */
'Vget (Msgid1,Msgid2,Command) Profile'

If Command = 'LOG' Then
/* ---------------------------------------------------------------- */
/* NOTE:                                                            */
/* Since we will have almost all SMS messages in the most recent    */
/* part of the SYSLOG, we are trying to pinpoint the messages other */
/* than IGDØØ9I and IGDØ14I. For this reason, with the help of the  */
/* edit macro, we are deleting these messages that had already been */
/* processed in the main REXX. Then we can easily localize other    */
/* SMS messages.                                                    */
/*                                                                  */
/* To suppress the reply messages that were already processed by the */
/* SmsREXX1, the macro will use the following algorithm, so that we */
/* will be dealing with other unprocessed IGD* messages.            */
/*                                                                  */
/* If Msgid1 ¬= 'MID1' (ie if it is 'IGDØØ9I')                      */
/*             Then Exclude the line that contains this message-id. */
/*                                                                  */
/* If Msgid2 ¬= 'MID2' (ie if it is 'IGDØ14I')                      */
/*             Then Exclude the line that contains this message-id. */
/* ---------------------------------------------------------------- */
     Do
      If Msgid1 ¬= 'MID1' Then 'Isredit Exclude'  '''Msgid1''' 'ALL'
      If Msgid2 ¬= 'MID2' Then 'Isredit Exclude'  '''Msgid2''' 'ALL'
      'Isredit Delete X ALL'
      'Isredit Exclude ALL'
      /* ------------------------------------------------------- */
      /* First message is most likely 'IGDØ56I' message. The other */
      /* one is most likely outstanding 'IGDØ4ØD' message, that    */
      /* requires a reply.                                         */
```

```
      /* ------------------------------------------------------------ */
      'Isredit Find 'IGD' 58 ALL' /*  Messages begin in this column. */
      'Isredit Find 'IGD' 61 ALL' /* WTOR msgs begin in this column. */
     End
     /* ------------------------------------------------------------ */
     /* If the command is not 'LOG' then this means that we are     */
     /* looking for the IGD* messages on the SDSF 'Message Line'    */
     /* that starts from column 22.                                 */
     /* ------------------------------------------------------------ */
     Else Do
           'Isredit Exclude ALL'
           'Isredit Find 'IGD' 22 FIRST'
          End
/* ---------------------------------------------------------------- */
/* Delete all excluded lines.                                       */
/* ---------------------------------------------------------------- */
'Isredit Delete ALL X'
/* ---------------------------------------------------------------- */
/* Save all changes made on the Isfout dataset and exit.            */
/* ---------------------------------------------------------------- */
'Isredit Save'
'Isredit Cancel'
Return
```

SYSINA

```
PROFILE NOPREF
DEL EXP.D310.SMS.TEST.RESULT1
DEL EXP.D310.SMS.TEST.RESULT2
ISPSTART CMD(ACBQBAIA +
Scdsname(Sis.D310.Acds) +
Testbed(Exp.D310.Sms.Test.Library) Member(*) +
Dc(Y) Sc(Y) Mc(Y) Sg(Y) +
Listname(Exp.D310.Sms.Test.Result1)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)

ISPSTART CMD(ACBQBAIA +
Scdsname(Sis.D310.Scds) +
Testbed(Exp.D310.Sms.Test.Library) Member(*) +
Dc(Y) Sc(Y) Mc(Y) Sg(Y) +
Listname(Exp.D310.Sms.Test.Result2)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
```

SYSIN: SYSINB

```
EXECUTIL SEARCHDD(YES)
%SMSREXX1
```

SYSIN: SYSINC

```
DELETE SIS.D31Ø.SCDS CLUSTER PURGE
SET MAXCC=Ø
IMPORT                                  -
     OUTDATASET(SIS.D31Ø.SCDS)          -
     INFILE(SOURCE)                     -
     INTOEMPTY                          -
     OBJECTS(                           -
           (SIS.D31Ø.SCDS)              -
           (SIS.D31Ø.SCDS.DATA))
```

SYSIN: SYSIND

```
REPRO INFILE(I1) OUTFILE(O1)
REPRO INFILE(I2) OUTFILE(O2)
REPRO INFILE(I3) OUTFILE(O3)
REPRO INFILE(I4) OUTFILE(O4)
REPRO INFILE(I5) OUTFILE(O5)
REPRO INFILE(I6) OUTFILE(O6)
```

SYSIN: SYSINE

```
EXECUTIL SEARCHDD(YES)
%SMSREXX2
```

SYSIN: SYSINF

```
EX 'EXP.D31Ø.SMS.SOURCE(SMSREXX5)'SYSIN: SYSING
REPRO INFILE(IN) OUTFILE(LOOKAT4) SKIP(3)
```

SYSIN: SYSINH

```
REPRO INFILE(INPUT) OUTFILE(LOOKAT1)
```

SYSIN: SYSIN0

```
REPRO INFILE(DCI) OUTFILE(DCO)
REPRO INFILE(MCI) OUTFILE(MCO)
REPRO INFILE(SCI) OUTFILE(SCO)
REPRO INFILE(SGI) OUTFILE(SGO)
```

SYSIN: SYSIN1

```
REPRO INFILE(INPUT) OUTFILE(TRANSLAT)
```

SYSIN: SYSIN2

```
REPRO INFILE(INPUT) OUTFILE(VALIDATE)
```

SYSIN: SYSIN3

```
REPRO INFILE(INPUT) OUTFILE(TEST)
```

SYSIN: SYSIN4

```
EXPORT SIS.D310.SCDS OUTFILE(OUT) CIMODE TEMPORARY
```

SYSIN: SYSIN5

```
REPRO INFILE(I1) OUTFILE(O1)
REPRO INFILE(I2) OUTFILE(O2)
REPRO INFILE(I3) OUTFILE(O3)
REPRO INFILE(I4) OUTFILE(O4)
REPRO INFILE(I5) OUTFILE(O5)
REPRO INFILE(I6) OUTFILE(O6)
REPRO INFILE(I7) OUTFILE(O7)
```

SYSIN: SYSIN6

```
COPY I=IN,O=OUT
```

SYSIN: SYSIN7

```
REPRO INFILE(IN1) OUTFILE(ISPLOGS)
REPRO INFILE(IN2) OUTFILE(TSOLOGS)
REPRO INFILE(IN3) OUTFILE(JOBLOGS)
```

SYSIN: SYSIN8

```
PROFILE NOPREF
DEL EXP.D310.SMS.DC.LISTING
DEL EXP.D310.SMS.MC.LISTING
DEL EXP.D310.SMS.SG.LISTING
DEL EXP.D310.SMS.SC.LISTING
ISPSTART CMD(ACBQBAO1 +
Acssrc(Exp.D310.Sms.Acs) Member(DATACLAS) +
Scdsname(Sis.D310.Scds) Listname(Exp.D310.Sms.Dc.Listing)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)

ISPSTART CMD(ACBQBAO1 +
Acssrc(Exp.D310.Sms.Acs) Member(STORCLAS) +
```

```
Scdsname(Sis.D310.Scds) Listname(Exp.D310.Sms.Sc.Listing)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)
ISPSTART CMD(ACBQBAO1 +
Acssrc(Exp.D310.Sms.Acs) Member(MGMTCLAS) +
Scdsname(Sis.D310.Scds) Listname(Exp.D310.Sms.Mc.Listing)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)

ISPSTART CMD(ACBQBAO1 +
Acssrc(Exp.D310.Sms.Acs) Member(STRGROUP) +
Scdsname(Sis.D310.Scds) Listname(Exp.D310.Sms.Sg.Listing)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)

EX 'EXP.D310.SMS.SOURCE(SMSREXX4)' 'Trans'SYSIN: SYSIN9

PROFILE NOPREF
DEL Exp.D310.Sms.Validate.Listing
ISPSTART CMD(ACBQBAO2 SCDSNAME(Sis.D310.Scds) TYPE(*) +
Listname(Exp.D310.Sms.Validate.Listing)) +
NEWAPPL(DGT) BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(99999999)

EX 'EXP.D310.SMS.SOURCE(SMSREXX4)' 'Valid'JCL: SMSCNTL1

//INEX0041  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl1                                               */
//* This job allocates the following datasets. It has to be run just  */
//* once before implementing the procedures SMSNEW and SMSREST.       */
//*                                                                   */
//* 1- Exp.D310.Sms.Bkpcds       : Holds historic SMS configurations. */
//* 2- Exp.D310.Sms.Test.Library : Holds test cases.                  */
//* 3- Exp.D310.Sms.Test.Title   : Holds title members.               */
//*********************************************************************
//*
//DEFINE1   EXEC PGM=IEFBR14
//DSET1     DD   DSN=EXP.D310.SMS.TEST.LIBRARY,DISP=(,CATLG),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1)),DCB=(LRECL=80,BLKSIZE=0,RECFM=FB)
//*
//DSET2     DD   DSN=EXP.D310.SMS.TEST.TITLE,DISP=(,CATLG),UNIT=SYSDA,
// SPACE=(TRK,(1,1,1)),DCB=(LRECL=133,BLKSIZE=0,RECFM=FBA)
//*
//DEFINE2   EXEC PGM=IDCAMS
//SYSPRINT  DD   SYSOUT=*
//SYSIN     DD  *
      DEF  GDG (NAME(EXP.D310.SMS.BKPCDS)  -
                NOEMPTY                     -
                SCRATCH                     -
                LIMIT(5))
```

JCL: SMSCNTL2

```
//INEX0043  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
```

```
//* JCL NAME : SmsCntl2                                                 */
//* This job recovers the SCDS dataset from the BKPCDS dataset.        */
//*                                                                     */
//* You have to manually change the absolute generation number on the */
//* BKPCDS DD card. Because it is not possible to join member name     */
//* and relative version number on this DD card. For this reason, you */
//* must find out the absolute generation name and put it on the       */
//* the BKPCDS DD card.                                                 */
//*                                                                     */
//* VOLUMES parameter specifies the volume on which the SCDS is to     */
//* reside. If you want the receiving volume to be the original        */
//* volume, this parameter can be skipped.                              */
//*********************************************************************/
//STEP     EXEC  PGM=IDCAMS
//BKPCDS   DD    DISP=SHR,DSN=EXP.D310.SMS.BKPCDS.G0071V00(SCDS)
//SYSPRINT DD    SYSOUT=*
//SYSIN    DD    *
  DELETE SIS.D310.SCDS CLUSTER PURGE
  IMPORT                                  -
      OUTDATASET(SIS.D310.SCDS)           -
      INFILE(BKPCDS)                      -
      INTOEMPTY                           -
      OBJECTS(                            -
            (SIS.D310.SCDS                -
             VOLUMES(DRS303))             -
            (SIS.D310.SCDS.DATA))
```

JCL: SMSCNTL3

```
//INEX0044  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl3                                                */
//* This job allocates an SCDS dataset.                                */
//* NOTE 1- Specifying 'Shareoptions(2,3)' lets one update-mode user   */
//*         operate simultaneously with other read-mode users between */
//*         regions.                                                   */
//*      2- REUSE option can be used to avoid running into space       */
//*         problems (SMS Reason Code 6068) as a result of subsequent */
//*         SCDS updates, or Import/Export functions.                  */
//*********************************************************************
//STEP     EXEC PGM=IDCAMS
//SYSUDUMP DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DEFINE CLUSTER(NAME(SIS.D310.SCDS)    -
 LINEAR                                -
 VOL(DRS303)                           -
 CYL(5)                                -
 SHAREOPTIONS(2,3))                    -
 DATA(NAME(SIS.D310.SCDS.DATA))
```

JCL: SMSCNTL4

```
//INEX0045  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl4                                               */
//* This job allocates an ACDS dataset.                               */
//* NOTE: This dataset must be shared between systems that are        */
//*       managing a shared DASD configuration in a SMS environment.  */
//*       For this reason, 'Shareoptions(3,3)' is specified.          */
//*       This allows full authority to read from and write to an     */
//*       ACDS from any system.                                       */
//*********************************************************************
//STEP     EXEC PGM=IDCAMS
//SYSUDUMP DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DELETE SIS.D310.ACDS CLUSTER PURGE
 DEFINE CLUSTER(NAME(SIS.D310.ACDS)    -
 LINEAR                                -
 VOL(DRS304)                           -
 CYL(5)                                -
 SHAREOPTIONS(3,3))                    -
 DATA(NAME(SIS.D310.ACDS.DATA))
```

JCL: SMSCNTL5

```
//INEX0042  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl5                                               */
//* This job allocates a spare ACDS.                                  */
//*********************************************************************
//STEP     EXEC PGM=IDCAMS
//SYSUDUMP DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
 DEFINE CLUSTER(NAME(SIS.D310.ACDS.SPARE) -
 LINEAR                                -
 VOL(DRS303)                           -
 CYL(5)                                -
 SHAREOPTIONS(3,3))                    -
 DATA(NAME(SIS.D310.ACDS.SPARE.DATA))
```

CL: SMSCNTL6

```
//INEX0047  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl6                                               */
//* This job allocates an COMMDS dataset.                             */
//* NOTE: This dataset must be shared between systems that are        */
//*       managing a shared DASD configuration in a SMS environment.  */
```

```
//*       For this reason, 'Shareoptions(3,3)' is specified.          */
//*********************************************************************
//STEP     EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=*
//SYSUDUMP DD    SYSOUT=*
//SYSIN    DD    *
      DEFINE CLUSTER(NAME(SIS.D310.COMMDS)           -
             LINEAR                                  -
             VOL(DRS303)                             -
             TRK(1 1)                                -
             SHAREOPTIONS(3,3))                      -
             DATA(NAME(SIS.D310.COMMDS.DATA))
/*
```

JCL: SMSCNTL7

```
//INEX0046  JOB  MSGCLASS=X,MSGLEVEL=(1,1),CLASS=C
//*********************************************************************
//* JCL NAME : SmsCntl7                                               */
//* This job allocates a spare COMMDS dataset.                        */
//*********************************************************************
//STEP     EXEC  PGM=IDCAMS
//SYSPRINT DD    SYSOUT=*
//SYSUDUMP DD    SYSOUT=*
//SYSIN    DD    *
      DELETE SIS.D310.COMMDS.SPARE CLUSTER PURGE
      DEFINE CLUSTER(NAME(SIS.D310.COMMDS.SPARE)     -
             LINEAR                                  -
             VOL(DRS304)                             -
             TRK(1 1)                                -
             SHAREOPTIONS(3,3))                      -
             DATA(NAME(SIS.D310.COMMDS.SPARE.DATA))
/*
```

TITLE: TITLE1

```
  ************************************************************
  =======>  TEST BY USING PREVIOUS SMS CONFIGURATION  <=======
  ************************************************************
```

TITLE: TITLE2

```
  ******************************************************
  =======>  TEST BY USING NEW SMS CONFIGURATION  <=======
  ******************************************************
```

Using COBOL in z/OS

INTRODUCTION

We recently moved from OS/390 Version 2 Release 10 to z/OS Version 1 Release 2. Initially we were a little concerned about the potential impact that this could have on our numerous COBOL applications. However, we found that there were no significant problems with migration at all, and the benefits were impressive.

BENEFITS

During migration to z/OS there was no need to recompile or re-link the COBOL applications. The COBOL applications run unchanged on z/OS just like they would on OS/390 Version 2 Release 10.

However, an important benefit of migration is that COBOL will run in 64-bit z/OS. Although COBOL does not support 64-bit addresses in COBOL programs, users will get some important benefits of 64-bit z/OS just by moving to it. With a 64-bit addressable real memory backing the virtual memory, there is less paging and swapping and therefore better system performance. Even when your z/OS system is running in 64-bit mode, you can still run existing AMODE=24 and AMODE=31 applications without having to relink or recompile them. Furthermore, if your shop runs DB2 you can exploit 64-bit addressing for SQL statements in COBOL programs without any changes to the COBOL programs themselves. This is a real bonus.

THE FUTURE

IBM has mentioned no plans for 64-bit addressing support in COBOL. The BINDER already supports 64-bit Assembler programs and will support 64-bit C/C++ programs in the future. If our experience of improvements in system performance without any changes to our applications just by moving to z/OS is anything to go by then there is probably no incentive for IBM to provide 64-bit addressing support in COBOL in the foreseeable future.

Systems programmer (UK)

Converting files and translating special characters – part 2

In the previous edition of MVS Update *we provided an overview of the functionality of a utility to convert files and translate special characters. The source code is provided below.*

SOURCE

```
* simple structured macros
         MACRO
         IF    &COND,&KWD
.* IF (cond),THEN
.* code: [N](E, P, Z, H, L, M, O)
         GBLA  &IFH
         GBLC  &IFLBL(1Ø)
         LCLA  &K
         LCLC  &C,&LBL
&IFH     SETA  &IFH+1
&C       SETC  '&COND(1)'
&LBL     SETC  'IF&SYSNDX'
&IFLBL(&IFH) SETC '&LBL'
         AIF   ('&C'(1,1) EQ 'N').A1
         JN&C  &LBL
         AGO   .A2
.A1      ANOP
&K       SETA  K'&C-1
&C       SETC  '&C'(2,&K)
         J&C   &LBL
.A2      MEND

         MACRO
         EIF
         GBLA  &IFH
         GBLC  &IFLBL(1Ø)
         LCLC  &LBL
&LBL     SETC  '&IFLBL(&IFH)'
&LBL     DS    ØH
&IFH     SETA  &IFH-1
         MEND

         MACRO
         ELSE
         GBLA  &IFH
         GBLC  &IFLBL(1Ø)
```

```
         LCLC  &LBL1,&LBL2
&LBL1    SETC  '&IFLBL(&IFH)'
&LBL2    SETC  'IF&SYSNDX'
&IFLBL(&IFH) SETC '&LBL2'
         J     &LBL2
&LBL1    DS    ØH
         MEND

XEDCICNV TITLE 'Extended EDCICONV'
         PRINT NOGEN
         SPACE 3
**
* XEDCICNV function:
* The input dataset (DD-Name: SYSUT1) is copied to
* the output dataset (DD-Name: SYSUT2).
* Any missing attributes for the output file definition
* will be copied from the input file definition.
*
* Codepage 273 special characters ({ = X'43', etc) are
* converted to codepage 1047 equivalents.
*
* The following record format combinations are supported:
*  V(B) F(B) U.
*
* Trailing blanks will be removed from output RECFM=V,U
* records (the minimum record length is 1).
*
* The following file formats are supported:
*  PS (PO with explicit member), HFS (input).
**
         SPACE 1
**
* DD-Statements:
*  SYSUT1 - input
*  SYSUT2 - output
**
         SPACE 1
**
* EXEC-Parameter (optional):
*  /NT = no code translation
*  /T = code translation (default)
**
         SPACE 1
**
* Return-code:
*  Ø - OK
*  4 - OK, input file empty
*  8 - NOK, record length error
* 12 - NOK, SYSUT1 open error
* 16 - NOK, SYSUT2 open error
* 2Ø - NOK, buffer overflow
```

```
**
         SPACE 3
XEDCICNV CSECT
XEDCICNV AMODE 31
XEDCICNV RMODE 24
         SPACE 1
         BAKR  R14,Ø                    Save registers
         BASR  R12,Ø
         USING *,R12
* Get EXEC-Parameter (if specified)
         L     R2,Ø(R1)
         LH    R1,Ø(R2)
         LA    R2,2(R2)
         CH    R1,=H'2'
         IF    (H),THEN
         CLC   =C'/NT',Ø(R2)
         IF    (E),THEN
         MVI   NTRFLAG,1
         EIF
         EIF
         SPACE 1
         RDJFCB (SYSUT1,(INPUT))
         LA    R5,JFCB                  A(JFCB)
         USING JFCB_DSC,R5
         SPACE 1
         MVI   HFSIN,Ø
         CLC   =C'..',JFCBDSNM          Path?
         IF    (E),THEN                 HFS
         MVC   JFCLRECL,=H'32760'       maxLRECL
         MVI   JFCRECFM,JFCUND          RECFM=U
         MVI   HFSIN,1
         EIF
         SPACE 1
         OPEN  (SYSUT1,(INPUT)),TYPE=J
INDCB    USING IHADCB,SYSUT1
         TM    INDCB.DCBOFLGS,X'1Ø'     Input-OPEN OK?
         IF    (Z),THEN                 :no
         WTO   'SYSUT1 OPEN error',ROUTCDE=(11)
         MVC   RC,=H'12'
         J     EOJ                      OPEN error
         EIF
         SPACE 1
OUTDCB   USING IHADCB,SYSUT2
         SPACE 1
         RDJFCB (SYSUT2,(OUTPUT))
         LA    R5,JFCB                  A(JFCB)
         USING JFCB_DSC,R5
         SPACE 1
         MVC   DSN,JFCBDSNM             Dataset name
         MVC   VOL,JFCBVOLS             First volume ID
         OBTAIN CAMLIST
```

```
* OBTAIN error?
         LTR   R15,R15                   OBTAIN error?
         IF    (NZ),THEN                 :yes
         WTO   'SYSUT2 OBTAIN error',ROUTCDE=(11)
         MVC   RC,=H'12'
         J     EOJ                       OBTAIN error
         EIF
* merge DCB parameters
* priority: JCL, DSCB, input dataset
         LA    R2,DSN
         USING DSCB1,R2
         SPACE 1
         TM    JFCRECFM,X'FF'            JCL-RECFM?
         IF    (Z),THEN                  :no
         MVC   JFCRECFM,DS1RECFM         DSCB-RECFM
         TM    JFCRECFM,X'FF'            JCL-RECFM?
         IF    (Z),THEN                  :no
         MVC   JFCRECFM,INDCB.DCBRECFM
         EIF
         EIF
         SPACE 1
         CLC   JFCBLKSI,=H'Ø'            JCL-BLKSI?
         IF    (E),THEN                  :no
         MVC  JFCBLKSI,DS1BLKL           DSCB-BLKSI
         CLC   JFCBLKSI,=H'Ø'            JCL-BLKSI?
         IF    (E),THEN                  :no
         MVC   JFCBLKSI,INDCB.DCBBLKSI
         EIF
         EIF
         SPACE 1
         CLC   JFCLRECL,=H'Ø'            JCL-LRECL?
         IF    (E),THEN                  :no
         MVC   JFCLRECL,DS1LRECL         DSCB-LRECL
         CLC   JFCLRECL,=H'Ø'            JCL-LRECL?
         IF    (E),THEN                  :no
         MVC   JFCLRECL,INDCB.DCBLRECL
         EIF
         EIF
         SPACE 1
         OPEN  (SYSUT2,(OUTPUT)),TYPE=J  OPEN using JFCB
         TM    OUTDCB.DCBOFLGS,X'1Ø'     Output-OPEN OK?
         IF    (Z),THEN                  :no
         WTO   'SYSUT2 OPEN error',ROUTCDE=(11)
         MVC   RC,=H'16'
         J     EOJ                       OPEN error
         EIF
         MVC   MAXLRECL,OUTDCB.DCBLRECL
         SPACE 1
         CLI   HFSIN,1
         IF    (E),THEN
         STORAGE OBTAIN,LENGTH=BUFSIZE
```

```
         ST    R1,ABUF
         ST    R1,PTRBUF
         LR    R6,R1
         MVC   INDCB.DCBEODA,=AL3(HFSEOD)
         L     R8,=A(BUFSIZE)
HFSLOOP  GET   SYSUT1
* R1: A(REC)
         LR    R14,R1
         LH    R15,INDCB.DCBLRECL
         SR    R8,R15                    Residual length
         IF    (M),THEN                  Overflow
         MVC   RC,=H'2Ø'
         J     EOD
         EIF
         LR    R7,R15
         MVCL  R6,R14
         J     HFSLOOP
         SPACE 1
HFSEOD   LR    R9,R6
         EIF
         SPACE 1
         MVC   RC,=H'Ø'               reset RC
         SPACE 1
GETLOOP  DS    ØH
         CLI   HFSIN,Ø
         IF    (E),THEN
         GET   SYSUT1
         LR    R6,R1
         LH    R5,INDCB.DCBLRECL
         SPACE 1
         SPACE 1
         TM    INDCB.DCBRECFM,DCBRECF
         IF    (Z),THEN              RECFM=V?
         SH    R5,=H'4'
         LA    R6,4(R6)              Data
         EIF
         ELSE
         L     R6,PTRBUF
* R6: A(bufstart)
* R9: A(bufend)
         L     RØ,=X'ØØØØØØ15'       NL (NewLine) = record end
         LR    R8,R9
SCANLOOP SRST  R8,R6
         JH    EOD
         JO    SCANLOOP
* else found
* R6: string start, R8: string end
         LR    R5,R8
         SR    R5,R6                 Length
         LA    R1,1(R8)              Next string start
         ST    R1,PTRBUF
```

```
         EIF
         SPACE 1
* move input record to output buffer
* R5: data length
* R6: start of data
         MVI   OUTDATA,C' '    Assign blank character for empty record
         LR    RØ,R6
         LR    R1,R5
         LA    R14,OUTDATA
         LR    R6,R14
         LR    R15,R5
         MVCL  R14,RØ
         CLI   NTRFLAG,Ø
         IF    (E),THEN        No Translate not set
* translate output record
          SR   R2,R2           Start index
          LA   R4,256          Increment
* R5: limit
          LR   R1,R4           Save increment
          BCTR R1,Ø            Length code
TRLOOP    LA   R3,Ø(R2,R6)
          EX   R1,TREX
         BXLE  R2,R4,TRLOOP
         EIF
         SPACE 1
         TM   OUTDCB.DCBRECFM,DCBRECV   RECFM=V,U?
         IF   (Z),THEN         :no (=RECFM(F))
         LH   R1,OUTDCB.DCBLRECL
         SR   R1,R5
         IF   (P),THEN          Right-pad with blanks
         LR   RØ,R6
         AR   RØ,R5
         L    R15,=X'40000000'
         MVCL RØ,R14
         LH   R5,OUTDCB.DCBLRECL
         EIF
         LR   RØ,R6
         ELSE  ,                RECFM=V,U
         LA   R1,Ø(R5,R6)       End + 1
* remove trailing blanks
DEBLANK  BCTR R1,Ø
         CLI  Ø(R1),C' '
         IF   (E),THEN
         BCT  R5,DEBLANK
         EIF
         LTR  R5,R5
         IF   (Z),THEN
         LA   R5,1              Set minimum length
         EIF
         TM   OUTDCB.DCBRECFM,DCBRECU   RECFM=U?
         IF   (NO),THEN         :no (=RECFM(V))
```

```
         LA   R5,4(R5)
         STH  R5,RDW
         LA   RØ,OUTREC
         ELSE
         STH  R5,OUTDCB.DCBLRECL
         LA   RØ,OUTDATA
         EIF
         EIF
         SPACE 1
* test record length
         CH   R5,MAXLRECL
         IF   (H),THEN           Length error, display message
         CVD  R5,PL8
         UNPK LENIN,PL8
         OC   LENIN,=5C'Ø'
         LH   RØ,MAXLRECL
         CVD  RØ,PL8
         UNPK LENOUT,PL8
         OC   LENOUT,=5C'Ø'
         WTO  TEXT=MSG,ROUTCDE=(11)
         MVC  RC,=H'8'
         J     EOD
         EIF
         SPACE 1
         PUT   SYSUT2,(Ø)
         AP    NREC,=P'1'        Increment record count
         J     GETLOOP
         SPACE 1
EOD      CLOSE (SYSUT1,,SYSUT2)
         ICM   R2,15,ABUF        Test whether buffer allocated
         IF    (NZ),THEN         Yes: free
         STORAGE RELEASE,LENGTH=BUFSIZE,ADDR=(R2)
         EIF
         LH    R1,RC
         LTR   R1,R1
         IF    (Z),THEN          No errors
         CP    NREC,=P'Ø'
         IF    (E),THEN
         MVC   RC,=H'4'          :warning, no records processed
         EIF
         EIF
EOJ      LH    R15,RC
         PR    ,                 Program return
         SPACE
TREX     TR    Ø(Ø,R3),TRTAB
         SPACE

* symbolic register equates
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
          TITLE 'Data areas'
KB        EQU   1Ø24
BUFSIZE   EQU   1*KB*KB          Buffer size (=1 MB)
ABUF      DC    A(Ø)             Allocated address of HFS file buffer
PTRBUF    DS    A                Pointer to current entry
          SPACE
MAXLRECL  DS    H
RC        DS    H
PL8       DS    ØD,PL8           Workarea
NREC      DC    PL4'Ø'           Number of input records
HFSIN     DC    X'Ø'             1: HFS input
NTRFLAG   DC    X'Ø'             1: no translation
          SPACE
MSG       DS    ØF
MSGLEN    DC    AL2(MSGEND-MSGTEXT)
MSGTEXT   DC    C'RECLEN ERROR. LENIN:'
LENIN     DS    CL5
          DC    C' MAX. LENOUT:'
LENOUT    DS    CL5
MSGEND    EQU   *
          SPACE
* Codepage translation table (simplified 273 (German) -> 1047)
TRTAB     DC    256AL1(*-TRTAB)             X'ØØ'-X'ff'
          ORG   TRTAB+X'Ø5' TAB
          DC    C' '
          ORG   TRTAB+C'{'
          DC    X'cØ' ex 43
          ORG   TRTAB+C'}'
          DC    X'dØ' ex dc
          ORG   TRTAB+C'['
          DC    X'ad' ex 63
          ORG   TRTAB+C']'
          DC    X'bd' ex fc
          ORG   TRTAB+C'!'
          DC    X'5a' ex 4f
          ORG   TRTAB+C'|'
          DC    X'4f' ex bb
          ORG   TRTAB+C'¬'
```

```
         DC    X'5f' ex 5f
         ORG   TRTAB+C'-'
         DC    X'a1' ex 59
         ORG   TRTAB+C'\'
         DC    X'eØ' ex ec
         ORG
         SPACE 1
SYSUT1   DCB   DDNAME=SYSUT1,DSORG=PS,DEVD=DA,EODAD=EOD,MACRF=GL,      X
               EXLST=RDJFCB
SYSUT2   DCB   DDNAME=SYSUT2,DSORG=PS,DEVD=DA,MACRF=PM,                X
               EXLST=RDJFCB
         SPACE 1
         LTORG
         SPACE 1
         DS    ØF
RDJFCB   DC    X'87',AL3(JFCB)
JFCB     DS    CL176
         SPACE 1
CAMLIST  CAMLST SEARCH,DSN,VOL,WKA
         SPACE 1
VOL      DS    CL6
         SPACE 1
DSN      DS    CL44
WKA      DC    XL14Ø'Ø'        initialize work-area
         SPACE 1
BUFL     EQU   3276Ø
         DS    ØF
OUTREC   DS    CL(BUFL+4)
         ORG   OUTREC
RDW      DS    H,H
OUTDATA  DS    CL(BUFL)
         DS    CL256           filler for TR instruction
         ORG

         TITLE 'DSECTs'
         DCBD  DSORG=PS,DEVD=DA
         SPACE 1
DSCB1    DSECT
         DS    CL44
DS1FMTID DS    C
         ORG   DSCB1+84
DS1RECFM DS    XL1                record format
         ORG   DSCB1+86
DS1BLKL  DS    HL2                block length
DS1LRECL DS    HL2                record length
         SPACE 1
JFCB_DSC DSECT ,                  JFCB-mapping
         IEFJFCBN LIST=NO
         END
```

IBM has announced the z800, the latest member of the zSeries family. It is the world's first Linux mainframe and it highlights IBM's growing commitment to the open source operating system. The bundle combines hardware, virtualization software, and maintenance. The z800 is aimed at server consolidation and infrastructure applications such as firewalls, Web serving, file and print serving and mail serving. It will support the Redhat, SuSE, and Turbolinux distributions of Linux.

Specific elements include one to four z800 Integrated Facility for Linux engines, memory, channels, and OSA Express connectivity. It also supports HiperSockets, for high-speed TCP/IP communication among virtual machines, and logical partitions within the same zSeries. Furthermore it includes z/VM Version 4, three years of hardware support and three years of z/VM subscription and support, and optional services from IBM Global Services for Linux on zSeries.

A new optional service is IBM Network Integration and Deployment Services for zSeries Fiber Cabling, which provides personalized cabling to support current and future z800 systems. The only prerequisite is that sites using parallel-attached devices need to get a parallel channel converter box in order to continue using them with the zSeries Offering for Linux.

For further information contact your local IBM representative.

http://www.ibm.com/servers

Serena has announced general availability of Version 3.1 of its StarTool IOO I/O optimizer for z/OS and OS/390, which automates I/O tuning functions. New in this release is statistical reporting, which will enable users to determine how an application is performing, identify the resources used, and compare report data to optimize efficiency and ensure that application tuning requirements are met.

Other enhancements include z/OS support for users running the product in z/OS 64-bit mode, as well as standardization of the SMF Record Header format to help speed the delivery of applications across the enterprise.

The integrated optimization system tunes the major components of z/OS and OS/390's I/O processing to improve batch and on-line throughput. The product is based on a proprietary analysis and intelligence gathering process during the life of an I/O operation.

For further information contact:

Serena Software, Inc, 500 Airport Boulevard, 2nd Floor, Burlingame, California 94010 1904, USA.
Tel: (650) 696 1800
Fax: (650) 696 1849

Serena Software UK Limited, Nash House, Repton Place, White Lion Road, Amersham, Buckinghamshire HP7 9LP, UK.
Tel: (01494) 766777
Fax: (01494) 766888

http://www.serena.com