175

MVS update

April 2001

In this issue

MVS Update

Published by

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

North American office

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

Contributions

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

Editor

Jaime Kaminski

Disclaimer

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

MVS Update on-line

Code from *MVS Update* can be downloaded from our Web site at http://www.xephon.com/mvsupdate.html; you will need to supply a word from the printed issue.

Subscriptions and back-issues

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

Printed in England.

Potential confusion surrounding IBM’s new ‘z’ products

INTRODUCTION

IBM’s new z/Series product line of mainframes is generating more than its fair share of confusion amongst users who are eager to exploit this latest technology, but are not quite sure of the implications in terms of hardware and software configurations. There are three principal areas causing confusion with users.

NOMENCLATURE

Firstly there is the issue of nomenclature. Because of the nature of the development of complex IT products nowadays, manufacturers seem to always have various tags to refer to different aspects of products under development, some of which never see the light of the public marketplace.

There are at least two such tags which crop up surrounding the ‘z’ products, one is ‘Freeway’ and the other is ‘ESAME’ or ‘ESA/ME’ as I have also seen it. ‘Freeway’ is fairly well established now as an alternative to zSeries as a generic description of the new processor hardware although I am not sure if that is how IBM used it internally. ‘ESAME’ is an acronym for Enterprise Systems Architecture Modal Extensions, which certainly does not roll off the tongue. It was apparently an internal name for what is now called z/Architecture.

HARDWARE REQUIREMENTS

The second area of confusion which seems to arise regularly is exactly what type of IBM hardware is required to run various levels of IBM software. Specifically in this regard, what processor can run what level of OS/390 and/or z/OS.

The reason behind this second area of confusion, as I see it, is IBM’s introduction of two different ‘Architectural Level Sets’ (ALS) in the recent past. These are referred to as ALS1 and ALS2. A sublevel of confusion arises here because of the way IBM refers to Generations within the 9672 product line, with say a Generation 4 (G4) model being a 9672-abc where the value of ‘c’ is 5. Hence a 9672-R55 is a G4 machine and a 9672-R84 is a G3.

- ALS1 was introduced with the G2 level of 9672 processors and is also present on P/390, R/390, Multiprise and Integrated Server products. ALS1 is required for OS/390 Release 10 (ie OS/390 Version 2 Release10).
- ALS2 was introduced with the G5 level of 9672 processors, and is also present on zSeries processors and Multiprise 3000 models. ALS2 is required for z/OS Version 1 Release 1.

To reiterate, z/OS will not run on pre-ALS2 hardware, which includes P/390, R/390, MP2000, and Integrated Server systems. The users of these systems fall into different categories, one being small companies with only declining legacy mainframe requirements who will probably be content to progress no further than OS/390 Version 2 Release 10. But another category is Independent Software Vendors (ISVs) and small companies who, while their mainframe use is small, nevertheless desire to maintain currency in the coming years. The MP3000 range is ALS2 compliant so that is one option for these users, but even the MP3000 is overly large for the needs of some users, especially small ISVs.

For these, Flex-ES running on NUMA-Q or Netfinity hardware has been identified as the platform for the future, although IBM will, apparently, not supply these machines directly, but through business partners instead.

64-BIT SUPPORT

The third area of confusion that I want to address is that surrounding the 64-bit support introduced in with the z/Architecture and z/OS announcements. The most common misconception is that IBM haS introduced 64-bit virtual addressing, much as XA extended the old 24-bit (16 megabyte) 370 architecture address space to 31-bit (2 gigabyte). This is definitely not the case yet, rather the 64-bit support is to enable larger real storage configurations on z/Architecure systems.

Even the recent Preview Announcement of z/OS Version 1 Release 2 made no reference to any timetable for 64-bit virtual support.

Z/OS AND OS/390 VERSION 2 RELEASE 10

Related to this is the question of the differences between z/OS Version 1 Release 1 and OS/390 Version 2 Release 10. The answer seems to be not a great deal, which is why the upgrade is relatively straight forward. One difference is that on a zSeries processor, OS/390 Version 2 Release 10 gives the installation the option of running in either 31-bit or 64-bit mode, while z/OS Version 1 Release 1 will run only in 64-bit mode.

Having said that, z/OS Version 1 Release 1 can run in 31-bit mode on a non-zSeries ALS2 system, and it can apparently even be forced to run in 31-bit mode on a zSeries if absolutely necessary, but assistance from IBM is required for this option and it is envisaged that this would be only in situations where proven problems exist in 64-bit mode.

On the point of upgrading, yet another source of confusion is the so-called Product Upgrade Package (PUP) upgrade option to get to z/OS Version 1 Release 1. The PUP is solely intended for customers already at OS/390 Version 2 Release 10 to quickly and easily upgrade to z/OS Version 1 Release 1, it has no other use and does not replace any of the existing operating system upgrade methods.

CONCLUSION

Since IBM is nowadays in a more or less six monthly release schedule for new operating system functionality, it is becoming increasingly difficult to keep abreast, especially when new terminology accompanies so many of these releases. I hope that this short discussion of some of the more common areas of confusion that I have witnessed will help to clarify matters.

Patrick Mullen
Consultant (Canada)

Spool offload facility

The Spool Offload facility (SOF) is a panel-driven ISPF application written in Assembler. SOF uses dynamic allocation and the Catalog Search Interface, program name IGGCSI00 (in SYS1.LINKLIB). This technology is documented in:

1 *OS/390 MVS Programming: Authorized Assembler Services Guide*, GC28-1763-08, Chapter 25 – *Dynamic Allocation*, and Chapter 26 – *Requesting Dynamic Allocation Functions*.

2 *DFSMS/MVS Version 1 Release 5, Managing Catalogs*, SC26-4914-04, Appendix D.

As the informational messages are displayed on the TSO/ ISPF screen, the following JCL statement should be included into the TSO logon JCL:

```
//PRINTOUT  DD TERM=TS,SYSOUT=*
```

The offloaded spool datasets are allocated by the transmit process. The offload receive panel displays the list of allocated datasets and the requested dataset can be selected. The fast selection of the required data is enabled by implementing a simple naming standard for the offloaded spool datasets. The naming standard for the offloaded spool datasets is:

```
SYS2.OFFLOAD1.Dddmmmyy.ThhHmmss.TY
```

Where:

- SYS2.OFFLOAD1 – is the prefix of each spool offload dataset.
- dd – day number
- mmm – month name
- yy – first two digits of the year
- hh – time (hour)
- mm – time (minutes)
- ss – time (seconds)
- TY – classes offloaded to tape.

For example SYS2.OFFLOAD1.D01DEC20.T20H3154.TYmeans that the spool was offloaded on 1 December 2000, at 20:31:54, the offloaded classes were T and Y.

SYS2.OFFLOAD1.D12DEC20.T20H0950 means that the spool was offloaded on 12 December 2000, at 20: 09:50, the offloaded class was Y.

This naming standard can be modified according to your individual requirements. SOF has two basic functions:

1 Spool offload transmit – offload specific classes from JES2 Spool to cartridges(s).

2 Spool offload receive – upload contents of the cart(s) to JES2 Spool.

The classes (to be spooled off to cartridges) should be selected manually from the panel. The spool offload services menu ISPF panel is called using the following sequence:

```
OFF,'PANEL(PNLOFFLØ)'
```

The following panel is displayed:

```
PNLOFFLØ -------- SOFF - Spool Offload JSE Services Menu ----------------
OPTION ===>

Specify the spool offload function to be performed and press <ENTER>
R    Receive - Upload contents of the cart(s) to JES2 Spool.
T    Transmit - Offload specific classes from JES2 Spool to cart(s).
X    EXIT - Exit from Spool Offload Menu.
```

Two functions are available: receive and transmit. When selecting R or T the following sequence is executed:

```
/**** Transmit to offload to the cart(s)    **********/
T,'PGM(SUOFTRAN)'
/**** Receive to upload from the cart(s)    **********/
R,'PGM(SUOFRECE)'
```

1 The transmit function will offload the specific classes of JES2 spool to cart(s). When selecting T (transmit) the following panel is displayed:

```
--------------- Transmit Function of Spool Offload Services -----------

Transmit function will offload the specific classes of JES2 spool to
cart(s). Specify the classes to be spooled off to carts and press <ENTER>
Class(es)   :           Only T and/or Y classes allowed
```

The current application allows to specify only T and Y classes, but this can be easily changed. After selecting the classes and pressing the enter key the following information is displayed on the screen:

```
CLASSES=T
THE FOLLOWING JCL WAS SUBMITTED:
 8:Ø9:41
//HZTOFFTR JOB (HZT,SW),(SPOOL),CLASS=F,
//   MSGCLASS=T,NOTIFY=FSSØ3
// COMMAND  '$P OFFLOAD1      '
// COMMAND  '$TOFF1.ST,WS=(Q/)'
// COMMAND  '$TOFF1.ST,DISP=DELETE,Q=T         '
// COMMAND  '$TOFFLOAD1,DSN=SYS2.OFFLOAD1.D14DEC2Ø.TØ8HØ941.T      '
// COMMAND  '$TOFFLOAD1,UNIT=CART,LABEL=SL                         '
// COMMAND  '$SOFFLOAD1,TYPE=TRANSMIT                              '
//IEFBR14  EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//* SPOOL OFFLOAD TRANSMIT TO CARTS
//*  SYS2.OFFLOAD1.D14DEC2Ø.TØ8HØ941.T
/*EOF
```

The Spool Offload Transmit can be traced on the operator’s console.

2 The Receive Function will upload the contents of the selected tape data. When selecting R (receive) the following panel is displayed:

```
--------------------- Receive Function of Spool Offload Services ------

 Receive Function will upload the contents of the selected tape data.
Select the jobname (or ALL) and the Output Class(es) and press <ENTER>
Jobname or ALL :
Output Class   :         Only T and/or Y classes allowed
```

The requested jobname (or ALL) and the output class(es) should be selected manually from the panel. Then a ‘spool datasets names list’ panel will appear:

```
Spool Datasets Names List
The number of Spool Datasets found  27   .
To Select insert the select code S and press <ENTER>
 Sel ............Data Set Name....................
First    Total
 code
cart     carts

     SYS2.OFFLOAD1.DØ1FEB2Ø.T2ØH54Ø4.TY            DØ4Ø79     2
     SYS2.OFFLOAD1.D02FEB20.T20H2721.TY            D05455     2
     SYS2.OFFLOAD1.D05FEB20.T20H4446.TY            D05043     3
     SYS2.OFFLOAD1.D06FEB20.T20H3452.TY            D05209     3
```

After selecting the requested spool offload dataset (by checking date, time, and class) the following information is displayed on the screen:

```
JOBNAME=FSS03
OUTCLASS T
DSN= SYS2.OFFLOAD1.D02FEB20.T20H2721.TY
THE FOLLOWING JCL WAS SUBMITTED:
//HZTOFFRE JOB (HZT,SW),(SPOOL),CLASS=F,
//   MSGCLASS=T,NOTIFY=FSS03
// COMMAND  '$DOFF1.SR                                              '
// COMMAND  '$TOFF1.SR,WS=(JOB,Q/),JOBNAME=FSS03                    '
// COMMAND  '$TOFF1.SR,Q=T                                          '
// COMMAND  '$TOFFLOAD1,DSN= SYS2.OFFLOAD1.D02FEB20.T20H2721.TY     '
// COMMAND  '$SOFFLOAD1,TYPE=RECEIVE                                '
//IEFBR14  EXEC PGM=IEFBR14
//SYSPRINT DD SYSOUT=*
//* SPOOL OFFLOAD RECEIVE FROM CARTS
//* SYS2.OFFLOAD1.D02FEB20.T20H2721.TY
/*EOF
```

The ‘spool offload receive’ can be traced on the operator’s console. The following source code is used:

1 Assembler routines: SUOFTRAN, SUOFRECE, and DYINTRDR.

2 ISPF panels: PNLOFFL0, PNLOFRE1, PNLOFRE2, and PNLOFTR1.

3 ISPF message: JMSD005.

The standard compile and link procedure should be used. The Linkage Editor step should contain two additional libraries in the //SYSLIB DD statement:

```
     SYS1.ISP.SISPLOAD (for ISPF functions)
     SYS1.LINKLIB      (for IGGCSI00)
```

The output from the Linkage Editor should go to your ISPF library concatenated to the ISPLLIB sequence. The SUTIME procedure was described in *MVS Update*, Issue 102, March 1995, page 70.

A copy of the date conversion subroutine SUYYDD2K has not been supplied because most shops will have different requirements. Any date handling procedure can be used to perform the conversion of the date from TIME macro format to a specific format. Please e-mail me if you need a copy of SUYYDD2K (*stevek@jse.co.za*).

```
RØ       EQU   Ø
....
R15      EQU   15
SUOFTRAN CSECT
         USING *,R1Ø,R11           ESTABLISH ADDRESSABILITY
         STM   R14,R12,12(R13)     SAVE3 REGISTERS
         LR    R1Ø,R15             SET FIRST BASE REGISTER
         LA    R11,2Ø48(R1Ø)       SET SECOND BASE REGISTER
         LA    R11,2Ø48(R11)       AND INCREMENT TO PROPER VALUE
         LR    R12,R13             STORE PREVIOUS SA ADDRESS
GETMA1   GETMAIN R,LV=4ØØ
         LR    R9,R1 (R9) = ADDR. OF THE ALLOCATED VIRTUAL STORAGE AREA
         LTR   15,15
         BZ    OKGETMA1
         LA    R15,4
         B     ENDRET
OKGETMA1 EQU   *
         USING VARIDSEC,R9
         LA    R13,SAVE3           LOAD ADDRESS OF THIS SAVE3 AREA
         ST    R12,SAVE3+4         CHAIN BACKWARDS
         ST    R13,8(R12)          CHAIN FORWARD
         OPEN  (PRINTDCB,(OUTPUT))
         MVC   FILE(8),=C'SUPEFILE'
*            DEFINE VARIABLE FILE FOR DIALOG SERVICE
*            RECN
         CALL  ISPLINK,(VDEFINE,NRECN,RECN,FIXED,LRECN),VL
*            MEMB
         CALL  ISPLINK,(VDEFINE,NMEMB,MEMB,CHAR,LMEMB),VL
*            VOLSER
         CALL  ISPLINK,(VDEFINE,NVOLSER,VOLSER,CHAR,LVOLSER),VL
*            CATLG
         CALL  ISPLINK,(VDEFINE,NCATLG,CATLG,CHAR,LCATLG),VL
*            DSN
         CALL  ISPLINK,(VDEFINE,NDSN,DSN,CHAR,LDSN),VL
*            DSNM
         CALL  ISPLINK,(VDEFINE,NDSNM,DSNM,CHAR,LDSNM),VL
*            SDATE
         CALL  ISPLINK,(VDEFINE,NSDATE,SDATE,CHAR,LSDATE),VL
*            STIME
         CALL  ISPLINK,(VDEFINE,NSTIME,STIME,CHAR,LSTIME),VL
*            SEL
         CALL  ISPLINK,(VDEFINE,NSEL,SEL,CHAR,LSEL),VL
*            RETURN CODE
         CALL  ISPLINK,(VDEFINE,NRC,RC,FIXED,LRC),VL
*            REPLY
```

```
         CALL  ISPLINK,(VDEFINE,NR,R,CHAR,LR),VL
DIFIPANE LA    R15,Ø
         ST    R15,RC
         ST    R15,RECN
         MVC   MEMB(8),BLANK
         MVC   VOLSER(6),BLANK
         MVC   CATLG(1),BLANK
         MVC   R,BLANK
         MVI   SEL,C' '
         CALL  ISPLINK,(DISPLAY,PNLOFTR1),VL
         LTR   R15,R15
         BNZ   ENDPRO3
         MVC   PRINT,BLANK
         MVC   PRINT+1(8),=C'CLASSES='
         MVC   PRINT+9(8),MEMB
         PUT   PRINTDCB,PRINT
*            PREPARE THE JOB
*            SUBMIT THE JOB FOR SPOOL
*            ALLOCATE INTERNAL READER DATA SET
         CALL  DYINTRDR,(DDNAMEAD),VL
*            OK ALLOCATION
*            INTERNAL READER DATASET ALLOCATED SUCCESSFULLY
         LA    R1,DDWORKNA
         L     R3,DDNAMEAD
         MVC   Ø(8,R1),Ø(R3)
*            ALLOCATION OF THE INTERNAL READER DATA SET COMPLETED
*            OPEN OWN DCB'S: SUBMIDCB
         LA    R2,SUBMIDCB         SUBMIT DCB
         USING IHADCB,R2
         MVC   DCBDDNAM(8),DDWORKNA
         DROP  R2
         OPEN  (SUBMIDCB,(OUTPUT))
         MVC   PRINT,BLANK
         MVC   PRINT+1(33),=C'THE FOLLOWING JCL WAS SUBMITTED: '
         PUT   PRINTDCB,PRINT
         TIME  BIN
         ST    R1,NUMBER
         CALL  SUYYDD2K,(NUMBER,STAMP),VL
         TIME  BIN
         ST    RØ,NUMBER
         CALL  SUTIME,(NUMBER,TIMESTAM),VL
         MVC   PRINT,BLANK
         MVC   PRINT+1(8),TIMESTAM
         PUT   PRINTDCB,PRINT
         CLC   TIMESTAM(1),=C' '   TEST IF BLANK
         BNE   NOBLANK1
         MVC   TIMESTAM(1),=C'Ø'   FILL WITH Ø
NOBLANK1 CLC   TIMESTAM+1(1),=C' ' TEST IF BLANK
         BNE   NOBLANK2
         MVC   TIMESTAM+1(1),=C'Ø' FILL WITH Ø
```

```
NOBLANK2 MVC   DSNAME(44),BLANK
         MVC   DSNAME(15),=C'SYS2.OFFLOAD1.D'
         MVC   DSNAME+15(7),DAYNO
         MVC   DSNAME+22(2),=C'.T'
         MVC   DSNAME+24(7),TIMESTAM
         MVC   DSNAME+26(1),=C'H'
         MVC   DSNAME+29(2),TIMESTAM+6
         MVC   DSNAME+31(1),=C'.'
         MVC   DSNAME+32(8),MEMB
         MVC   COMMENT1+5(44),DSNAME
*            PUT SINGLE QUOTES AND SUBMIT
         MVC   COMMAND1+12(1),SINGQUOT
         MVC   COMMAND1+3Ø(1),SINGQUOT
         MVC   COMMAND2+12(1),SINGQUOT
         MVC   COMMAND2+3Ø(1),SINGQUOT
         MVC   COMMAND3+12(1),SINGQUOT
         MVC   COMMAND3+37(2),MEMB
         MVC   COMMAND3+49(1),SINGQUOT
         MVC   COMMAND4+12(1),SINGQUOT
         MVC   COMMAND4+28(4Ø),DSNAME
         MVC   COMMAND4+68(1),SINGQUOT
         MVC   COMMAND5+12(1),SINGQUOT
         MVC   COMMAND5+68(1),SINGQUOT
         MVC   COMMAND6+12(1),SINGQUOT
         MVC   COMMAND6+68(1),SINGQUOT
*            SUBMIT STEP1
         LA    R2,JOBLENGT  (R2) = NUMBER OF JCL 7Ø BYTES LONG CARDS
         LA    R3,JCLREC    (R3) = ADDRESS OF JCLREC
         LA    R4,JOBCARD1  (R4) = ADDRESS OF JOBCARD1
LSUBST1  MVC   PRINT,BLANK
         MVC   JCLREC(8Ø),BLANK
         MVC   Ø(7Ø,R3),Ø(R4)
         MVC   PRINT+1(8Ø),JCLREC
         PUT   PRINTDCB,PRINT
         PUT   SUBMIDCB,JCLREC
         A     R4,=F'7Ø'           INCREASE COUNTER VALUE
         BCT   R2,LSUBST1
         MVC   JCLREC(8Ø),BLANK
         MVC   JCLREC(5),EOF
         PUT   SUBMIDCB,JCLREC
         MVC   PRINT,BLANK
         MVC   PRINT+1(8Ø),JCLREC
         PUT   PRINTDCB,PRINT
         CLOSE SUBMIDCB
*            DELETE VARIABLES DEFINITIONS
ENDPRO3  CALL  ISPLINK,(VDELETE,NRECN),VL
         CALL  ISPLINK,(VDELETE,NMEMB),VL
         CALL  ISPLINK,(VDELETE,NVOLSER),VL
         CALL  ISPLINK,(VDELETE,NCATLG),VL
         CALL  ISPLINK,(VDELETE,NDSN),VL
         CALL  ISPLINK,(VDELETE,NDSNM),VL
         CALL  ISPLINK,(VDELETE,NSDATE),VL
         CALL  ISPLINK,(VDELETE,NSTIME),VL
         CALL  ISPLINK,(VDELETE,NRC),VL
         CALL  ISPLINK,(VDELETE,NR),VL
         CALL  ISPLINK,(VDELETE,NSEL),VL
         CLOSE (PRINTDCB)
FREEMA1  FREEMAIN  R,LV=4ØØ,A=(R9)
ENDRET   EQU   *
         L     R13,4(R13)
         LA    R7,Ø
         LR    R15,R7
         RETURN (14,12),RC=(15)
         EJECT
*           ISPF SERVICES CONSTANTS
         DS    ØD
DISPLAY  DC    CL8'DISPLAY '
SELECT   DC    CL8'SELECT  '
VDEFINE  DC    CL8'VDEFINE '
VGET     DC    CL8'VGET    '
VPUT     DC    CL8'VPUT    '
VDELETE  DC    CL8'VDELETE '
*           PARAMETERS
CHAR     DC    CL8'CHAR    '
FIXED    DC    CL8'FIXED   '
KEYS     DC    CL8'KEYS    '
NAMES    DC    CL8'NAMES   '
*           PANELS
PNLOFTR1 DC    CL8'PNLOFTR1'       MAIN SELECTION PANEL
*           TABLES
FILE     DC    CL8'        '       TABLE NAME
*         LENGTH PARAMETER IN 'CALL ISPLINK VDEFINE' MUST BE FULL WORD
LRECN    DC    F'4'    LENGTH OF THE RECN     FIELD
LMEMB    DC    F'8'    LENGTH OF THE MEMB     FIELD
LVOLSER  DC    F'6'    LENGTH OF THE VOLSER   FIELD
LCATLG   DC    F'3'    LENGTH OF THE CATLG    FIELD
LDSN     DC    F'44'   LENGTH OF THE DSN      FIELD
LDSNM    DC    F'54'   LENGTH OF THE DSNM     FIELD
LSDATE   DC    F'9'    LENGTH OF THE SDATE    FIELD
LSTIME   DC    F'5'    LENGTH OF THE STIME    FIELD
LSEL     DC    F'1'    LENGTH OF THE SEL      FIELD
LR       DC    F'1'    LENGTH OF THE REPLY FIELD
LRC      DC    F'4'    LENGTH OF THE RETURN CODE FIELD
*         NAME LISTS FOR VGET/VPUT SERVICE
         DS    ØD
NALNONKE DC    CL4Ø'(MEMB DSN DSNM VOLSER CATLG SDATE STIME)'
NALKEY   DC    CL6'(RECN)'
*          CONSTANTS FOR VARIABLES DEFINITION
         DS    ØD
NRECN    DC    CL6'(RECN)'
```

```
NMEMB    DC    CL6'(MEMB)'
NVOLSER  DC    CL8'(VOLSER)'
NCATLG   DC    CL7'(CATLG)'
NDSN     DC    CL5'(DSN)'
NDSNM    DC    CL6'(DSNM)'
NSDATE   DC    CL7'(SDATE)'
NSTIME   DC    CL7'(STIME)'
NSEL     DC    CL5'(SEL)'
NR       DC    CL3'(R)'
NRC      DC    CL4'(RC)'
         DS    ØD
TDSN     DS    ØCL164
RECN     DS    F                   RECORD NUMBER
DSN      DS    CL44                DATASET NAME
VOLSER   DS    CL6                 VOLSER
CAT      DS    CL1                 CATALOGED? (Y OR N)
TDATE    DS    CL15                DATE
TTIME    DS    CL8                 TIME
         DS    CL12Ø               FILLER
RC       DC    F'Ø'
NUMBER   DC    F'Ø'
MEMB     DC    CL8' '
MDSN     DC    CL44' '
DSNAME   DC    CL44' '
DSNM     DS    CL54                DATASET NAME
CATLG    DS    CL3                 CATALOGED? (Y OR N)
SEL      DC    CL1' '
R        DC    CL1' '
SDATE    DC    CL9' '
STIME    DC    CL5' '
DDNAMEAD DS    A                   ADDRESS OF THE DDNAME
MDSNDDAD DS    A                   ADDRESS OF THE DDNAME FOR MDSN
TRECN    DS    F                   TOTAL NUMBER OF RECORDS
DDWORKNA DS    CL8                 WORK DATASET DDNAME
MDSNDDNA DS    CL8                 WORK DATASET DDNAME FOR MDSN
PRINT    DS    CL133
BLANK    DC    CL133' '
         DS   ØD
STAMP    DS   ØCL12
DAY      DS   CL3      BLANK
DAYNO    DS   CL2      BLANK
MONTH    DS   CL3      BLANK
YEAR     DS   CL2      19
YEAR1    DS   CL2      BLANK
TIMESTAM DS   ØCL11
HH       DS   CL2      BLANK
         DS   CL1      BLANK
MM       DS   CL2      BLANK
         DS   CL1      BLANK
SS       DS   CL2      BLANK
```

```
         DS   CL1      BLANK
DD       DS   CL2      BLANK
JCLREC   DS    CL8Ø
JOBCARD1 DC    CL7Ø'//HZTOFFTR JOB (HZT,SW),(SPOOL),CLASS=F,          '
JOBCARD2 DC    CL7Ø'//   MSGCLASS=T                                   '
COMMAND1 DC    CL7Ø'// COMMAND  #$P OFFLOAD1      #                   '
COMMAND2 DC    CL7Ø'// COMMAND  #$TOFF1.ST,WS=(Q/)#                   '
COMMAND3 DC    CL7Ø'// COMMAND  #$TOFF1.ST,DISP=DELETE,Q=           #'
COMMAND4 DC    CL7Ø'// COMMAND  #$TOFFLOAD1,DSN=                      '
COMMAND5 DC    CL7Ø'// COMMAND  #$TOFFLOAD1,UNIT=CART,LABEL=SL        '
COMMAND6 DC    CL7Ø'// COMMAND  #$SOFFLOAD1,TYPE=TRANSMIT             '
STEP1EXE DC    CL7Ø'//IEFBR14  EXEC PGM=IEFBR14                       '
         DC    CL7Ø'//SYSPRINT DD SYSOUT=*                            '
         DC    CL7Ø'//* SPOOL OFFLOAD TRANSMIT TO CARTS               '
COMMENT1 DC    CL7Ø'//*                                               '
JOBLENGT EQU   (*-JOBCARD1)/7Ø     NUMBER OF JCL 7Ø BYTES LONG CARDS
EOF      DC    CL5'/*EOF'
SINGQUOT DC    XL1'7D'
         DS    ØD
PRINTDCB DCB   MACRF=PT,RECFM=FBA,LRECL=133,BLKSIZE=133,DSORG=PS,      *
               DDNAME=PRINTOUT
SUBMIDCB DCB   DSORG=PS,MACRF=(PM),RECFM=FB,LRECL=8Ø,BLKSIZE=8ØØØ
         LTORG
VARIDSEC DSECT   DUMMY SECTION
SAVE3    DS    18F
REPLY    DS    CL1
         DCBD  DSORG=PS             DUMMY SECTION
         END
RØ       EQU   Ø
...
R15      EQU   15
SUOFRECE CSECT
         USING *,R1Ø,R11            ESTABLISH ADDRESSABILITY
         STM   R14,R12,12(R13)      SAVE  REGISTERS
         LR    R1Ø,R15              SET FIRST BASE REGISTER
         LA    R11,2Ø48(R1Ø)        SET SECOND BASE REGISTER
         LA    R11,2Ø48(R11)        AND INCREMENT TO PROPER VALUE
         LR    R12,R13              STORE PREVIOUS SA ADDRESS
         GETMAIN R,LV=4ØØ
         LR    R9,R1 (R9) = ADDR. OF THE ALLOCATED VIRTUAL STORAGE AREA
         LTR   15,15
         BZ    OKGETMA1
         LA    R15,4
         B     ENDRET
OKGETMA1 ST    R9,R9SAVE
         USING VARIDSEC,R9
         LA    R13,SAVE3            LOAD ADDRESS OF THIS SAVE3 AREA
         ST    R12,SAVE3+4          CHAIN BACKWARDS
         ST    R13,8(R12)           CHAIN FORWARD
         OPEN  (PRINTDCB,(OUTPUT))
         MVC   FILE(8),=C'OFFLFILE'
*   DEFINE VARIABLE FILE FOR DIALOG SERVICE
         CALL  ISPLINK,(VDEFINE,NRECN,RECN,FIXED,LRECN),VL
         CALL  ISPLINK,(VDEFINE,NJOBNAME,JOBNAME,CHAR,LJOBNAME),VL
         CALL  ISPLINK,(VDEFINE,NVOLSER,VOLSER,CHAR,LVOLSER),VL
         CALL  ISPLINK,(VDEFINE,NIVOLS,IVOLS,CHAR,LIVOLS),VL
         CALL  ISPLINK,(VDEFINE,NOUTCLAS,OUTCLAS,CHAR,LOUTCLAS),VL
         CALL  ISPLINK,(VDEFINE,NDSN,DSN,CHAR,LDSN),VL
         CALL  ISPLINK,(VDEFINE,NSEL,SEL,CHAR,LSEL),VL
         CALL  ISPLINK,(VDEFINE,NRC,RC,FIXED,LRC),VL
DIFIPANE LA    R15,Ø
         ST    R15,RC
         ST    R15,RECN
         MVC   JOBNAME(8),BLANK
         MVC   VOLSER(6),BLANK
         MVC   IVOLS(4),BLANK
         MVC   OUTCLAS(8),BLANK
         MVC   R,BLANK
         MVI   SEL,C' '
         CALL  ISPLINK,(DISPLAY,PNLOFRE1),VL  FETCH JOBNAME, OUTCLASS
         LTR   R15,R15
         BNZ   ENDPRO3
*   CREATE AND OPEN TABLE FILE
         CALL  ISPLINK,(TBCREATE,FILE,NALKEY,NALNONKE,WRITE,REPLACE),VL
*            EXECUTE CATALOG SEARCH INTERFACE
         LA    R1,PARMLIST
         CALL  IGGCSIØØ
         LTR   R15,R15             TEST RETURN CODE
         BZ    OKRC                IF ZERO BYPASS CONVERSION
         MVC   PRINT(133),BLANK
         MVC   PRINT+1(16),=C'IGGCSIØØ RC NE Ø'
         PUT   PRINTDCB,PRINT
         B     ENDRTN
OKRC     EQU       *
         USING DATADSEC,R5
         LA    R5,WORKAREA         LOAD DSECT REG
         L     R1,CSICRETN         GET RETURN CODE
         LTR   R1,R1               TEST RETURN CODE
         BZ    DATARCOK            CONTINUE IF NO ERRORS
         MVC   PRINT(133),BLANK
         MVC   PRINT+1(16),=C'DATA RC NE Ø'
         PUT   PRINTDCB,PRINT
         B     ENDRTN
DATARCOK LA    R4,DSECTEND         GET BEGINNING OF INFO
         L     R6,CSIUSDLN         GET DATA USED LEN
         LA    R7,64               LENGTH OF ENTRY DATA
         USING MAPENTRY,R4
NEXTENT  LH    R8,EFLD2LN
         LTR   R8,R8               IS DATA LENGTH = Ø?
         BZ    CARRYLOO             YES
         LA    R9,VOLSENTR
         MVC   DSN(44),CSIENAME    MOVE DATASET NAME
         MVC   VOLSER(6),Ø(R9)
         XR    R3,R3
         LH    R3,EFLD2LN
 DATA LENGTH
         ST    R7,R7SAVE
         XR    R6,R6
         LR    R7,R3
         XR    R1,R1
         LA    R1,6
         DR    R6,R1
         LR    R1,R7
         BAL   R8,CONVEBOX
         MVC   IVOLS(4),RESULT1Ø+6
         L     R7,R7SAVE
         L     R9,RECN
         A     R9,=F'1'            INCREASE RECORD COUNTER
         ST    R9,RECN
         CALL  ISPLINK,(TBADD,FILE),VL  ADD ROW IN TABLE
         LTR   R15,R15             CHECK THE RC FORM THE SERVICE
         BZ    CARRYLOO     RC=Ø
         LR    R1,R15
         BAL   R8,CONVEBOX
         MVC   PRINT,BLANK
         MVC   PRINT+1(2Ø),=C'TBADD RC=           '
         MVC   PRINT+9(8),RESULT1Ø+2
         PUT   PRINTDCB,PRINT
CARRYLOO L     R6,CSIUSDLN
         SR    R1,R1
         LH    R1,EDATALN
         LA    R1,46(R1)
         AR    R7,R1
         AR    R4,R1
         CR    R7,R6
         BNM   ENDRTN
         B     NEXTENT             GO TO  NEXT ENTRY
*     SET ROW POINTER AT TOP, DISPLAY TABLE FILE
ENDRTN   CALL  ISPLINK,(TBTOP,FILE),VL
DISPTABL MVI   SEL,C' '
         CALL  ISPLINK,(TBDISPL,FILE,PNLOFRE2),VL
         LR    R7,R15
         LA    R1,Ø
         CR    R7,R1
         BE    DITARCZE     RC = Ø
*   RETURN CODE NE Ø, CLOSE TABLE FILE WITHOUT SAVING
         CALL  ISPLINK,(TBEND,FILE),VL
         B     DIFIPANE  DISPLAY PANEL PNLOFRE1 - PRIMARY OPTION MENU
DITARCZE CLI   SEL,C' '
         BE    DISPTABL            DISPLAY TABLE AGAIN
```

```
*            SEL NE ' '
*            TRANSLATE LETTER FROM LOWER CASE INTO UPPER CASE
*            TEST SEL VALUE AND EXECUTE
*            TEST IF SEL = B - DISPLAY ROW
         OI    SEL,X'CØ'
         CLI   SEL,C'S'
         BE    SSELECT     YES
         CALL  ISPLINK,(SETMSG,JMSDØØ5),VL   INVALID SELECTION CHAR
         B     DISPTABL    NOT 'S' - DISPLAY TABLE AGAIN
SSELECT  MVC   PRINT,BLANK
         MVC   PRINT+1(8),=C'JOBNAME='
         MVC   PRINT+9(8),JOBNAME
         PUT   PRINTDCB,PRINT
         MVC   PRINT,BLANK
         MVC   PRINT+1(8),=C'OUTCLASS='
         MVC   PRINT+1Ø(8),OUTCLAS
         PUT   PRINTDCB,PRINT
         MVC   PRINT,BLANK
         MVC   PRINT+1(4),=C'DSN='
         MVC   PRINT+5(44),DSN
         PUT   PRINTDCB,PRINT
*            PREPARE THE JOB
*            SUBMIT THE JOB FOR SPOOL
*            ALLOCATE INTERNAL READER DATA SET
         CALL  DYINTRDR,(DDNAMEAD),VL
         LA    R1,DDWORKNA
         L     R3,DDNAMEAD
         MVC   Ø(8,R1),Ø(R3)
*            OPEN OWN DCB'S: SUBMIDCB
         LA    R2,SUBMIDCB         SUBMIT DCB
         USING IHADCB,R2
         MVC   DCBDDNAM(8),DDWORKNA
         DROP  R2
         OPEN  (SUBMIDCB,(OUTPUT))
         MVC   PRINT,BLANK
         MVC   PRINT+1(33),=C'THE FOLLOWING JCL WAS SUBMITTED: '
         PUT   PRINTDCB,PRINT
         MVC   COMMENT1+5(44),DSN
*    INSERT SINGLE QUOTES AND SUBMIT
         MVC   COMMANDØ+12(1),SINGQUOT
         MVC   COMMANDØ+13(2Ø),=C'$TOFF1.SR,WS=(-JOB/)'
         MVC   COMMANDØ+68(1),SINGQUOT
         MVC   COMMAND1+12(1),SINGQUOT
         MVC   COMMAND1+68(1),SINGQUOT
*   CHECK IF JOBNAME=ALL JOBS
         CLC   JOBNAME(3),=C'ALL'
         BE    ALLJOBS
         MVC   COMMANDØ+13(2Ø),=C'$DOFF1.SR           '
         MVC   COMMAND1+26(19),=C'(JOB,Q/),JOBNAME=  '
         MVC   COMMAND1+43(8),JOBNAME
```

```
ALLJOBS  MVC   COMMAND2+12(1),SINGQUOT
         MVC   COMMAND2+25(8),OUTCLAS
         MVC   COMMAND2+68(1),SINGQUOT
         MVC   COMMAND3+12(1),SINGQUOT
         MVC   COMMAND3+28(4Ø),DSN
         MVC   COMMAND3+68(1),SINGQUOT
         MVC   COMMAND4+12(1),SINGQUOT
         MVC   COMMAND4+68(1),SINGQUOT
*    SUBMIT
         LA    R2,JOBLENGT  (R2) = NUMBER OF JCL 7Ø BYTES LONG CARDS
         LA    R3,JCLREC    (R3) = ADDRESS OF JCLREC
         LA    R4,JOBCARD1  (R4) = ADDRESS OF STEP1EXE
LSUBST1  MVC   PRINT,BLANK
         MVC   JCLREC(8Ø),BLANK
         MVC   Ø(7Ø,R3),Ø(R4)
         MVC   PRINT+1(8Ø),JCLREC
         PUT   PRINTDCB,PRINT
         PUT   SUBMIDCB,JCLREC
         A     R4,=F'7Ø'            INCREASE COUNTER VALUE
         BCT   R2,LSUBST1
SUBMIS2  MVC   JCLREC(8Ø),BLANK
         MVC   JCLREC(5),EOF
         PUT   SUBMIDCB,JCLREC
         MVC   PRINT,BLANK
         MVC   PRINT+1(8Ø),JCLREC
         PUT   PRINTDCB,PRINT
         CLOSE SUBMIDCB
         CALL  ISPLINK,(TBEND,FILE),VL   CLOSE TABLE, NO SAVING
*            DELETE VARIABLES DEFINITIONS
ENDPRO3  CALL  ISPLINK,(VDELETE,NRECN),VL
         CALL  ISPLINK,(VDELETE,NJOBNAME),VL
         CALL  ISPLINK,(VDELETE,NVOLSER),VL
         CALL  ISPLINK,(VDELETE,NIVOLS),VL
         CALL  ISPLINK,(VDELETE,NOUTCLAS),VL
         CALL  ISPLINK,(VDELETE,NDSN),VL
         CALL  ISPLINK,(VDELETE,NRC),VL
         CALL  ISPLINK,(VDELETE,NSEL),VL
         CLOSE (PRINTDCB)
         L     R9,R9SAVE
         FREEMAIN  R,LV=4ØØ,A=(R9)
ENDRET   L     R13,4(R13)
         LA    R7,Ø
         LR    R15,R7
         RETURN (14,12),RC=(15)
CONVEBOX EQU   *
         CVD   R1,PACKED
         MVC   COPYPATE(12),PATTERN
         ED    COPYPATE(12),PACKFIE2
         MVC   RESULT1Ø(1Ø),COPYPATE+2
         BR    R8
         DS    ØD
PACKED   DS    ØPL8
         DS    PL2
PACKFIE2 DS    PL6
PATTERN  DC    XL12'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø212Ø'
COPYPATE DS    CL12
*           ISPF SERVICES CONSTANTS
         DS    ØD
DISPLAY  DC    CL8'DISPLAY '
SELECT   DC    CL8'SELECT  '
SETMSG   DC    CL8'SETMSG  '
TBCREATE DC    CL8'TBCREATE'
TBOPEN   DC    CL8'TBOPEN  '
TBCLOSE  DC    CL8'TBCLOSE '
TBEND    DC    CL8'TBEND   '
TBADD    DC    CL8'TBADD   '
TBDISPL  DC    CL8'TBDISPL '
TBTOP    DC    CL8'TBTOP   '
VDEFINE  DC    CL8'VDEFINE '
VGET     DC    CL8'VGET    '
VPUT     DC    CL8'VPUT    '
VDELETE  DC    CL8'VDELETE '
CHAR     DC    CL8'CHAR    '
FIXED    DC    CL8'FIXED   '
NAMES    DC    CL8'NAMES   '
WRITE    DC    CL8'WRITE   '
REPLACE  DC    CL8'REPLACE '
*         PANELS
PNLOFRE1 DC    CL8'PNLOFRE1'       MAIN SELECTION PANEL
PNLOFRE2 DC    CL8'PNLOFRE2'       TABLE DISPLAY PANEL
*           TABLES
FILE     DC    CL8'        '       TABLE NAME
*           MESSAGES
JMSDØØ5  DC    CL8'JMSDØØ5 '       INVALID SELECTION CHARACTER
*         LENGTH PARAMETER IN 'CALL ISPLINK VDEFINE' MUST BE FULL WORD
LRECN    DC    F'4'    LENGTH OF THE RECN     FIELD
LJOBNAME DC    F'8'    LENGTH OF THE JOBNAME  FIELD
LVOLSER  DC    F'6'    LENGTH OF THE VOLSER   FIELD
LIVOLS   DC    F'4'    LENGTH OF THE IVOLS    FIELD
LOUTCLAS DC    F'8'    LENGTH OF THE OUTCLAS  FIELD
LDSN     DC    F'44'   LENGTH OF THE DSN      FIELD
LSEL     DC    F'1'    LENGTH OF THE SEL      FIELD
LRC      DC    F'4'    LENGTH OF THE RETURN CODE FIELD
*         NAME LISTS FOR TABLE SERVICE
         DS    ØD
NALNONKE DC    CL18'(DSN VOLSER IVOLS)'
NALKEY   DC    CL6'(RECN)'
*          CONSTANTS FOR VARIABLES DEFINITION
         DS    ØD
NRECN    DC    CL6'(RECN)'
```

```
NJOBNAME DC    CL9'(JOBNAME)'
NVOLSER  DC    CL8'(VOLSER)'
NIVOLS   DC    CL7'(IVOLS)'
NOUTCLAS DC    CL9'(OUTCLAS)'
NDSN     DC    CL5'(DSN)'
NSEL     DC    CL5'(SEL)'
NR       DC    CL3'(R)'
NRC      DC    CL4'(RC)'
*           VARIABLES DEFINITIONS
         DS    ØD
RECN     DS    F                   RECORD NUMBER
IVOLS    DS    F                   NUMBER OF VOLSERS
DSN      DS    CL44                DATASET NAME
VOLSER   DS    CL6                 FIRST CART VOLSER
R7SAVE   DS    F
R9SAVE   DS    F
RC       DC    F'Ø'
NUMBER   DC    F'Ø'
JOBNAME  DC    CL8' '
OUTCLAS  DS    CL8                 OUTCLAS
SEL      DC    CL1' '
R        DC    CL1' '
RESULT1Ø DS    CL1Ø
PRINT    DS    CL133
BLANK    DC    CL133' '
* PARAMETER LIST FOR IGGCSIØØ INVOCATION                             *
PARMLIST DS    ØD
PARMRC   DC    A(MODRSNRT)         MODULE/REASON/RETURN
         DC    A(CSIFIELD)         SELECTION CRITERIA FIELDS
         DC    A(WORKAREA)         RETURNED INFO
* MODULE ID/REASON CODE/RETURN CODE                                  *
MODRSNRT    DS ØF
MODID       DC XL2'ØØØØ'           MODULE ID
RSNCODE     DC XL1'ØØ'             REASON CODE
RTNCODE     DC XL1'ØØ'             RETURN CODE
* PARAMETER FIELDS FOR CATALOG SEARCH INTERFACE (CSI)                *
CSIFIELD    DS ØF
CSIFILTK    DC CL44'SYS2.OFFLOAD1.**'       FILTER   KEY
CSICATNM    DC CL44' '             CATALOG NAME OR BLANKS
CSIRESNM    DC CL44' '             RESUME NAME OR BLANKS
CSIDTYPD    DS ØCL16               ENTRY TYPES
CSIDTYPS    DC CL16'A              ' ENTRY TYPE: NONVSAM
CSICLDI     DC CL1' '              RETURN D&I IF C A MATCH Y OR BLNK
CSIOPTS     DS ØCL4                CSI OPTIONS
CSIRESUM    DC CL1' '              RESUME FLAG       Y OR BLANK
CSIS1CAT    DC CL1' '              SEARCH CATALOG    Y OR BLANK
CSIRESRV    DC XL1'ØØ'             RESERVED
CSINUMEN    DC H'1'                NUMBER OF ENTRIES FOLLOWING
CSIENTS     DS ØCL8                VARIABLE NUMBER OF ENTRIES FOLLOW
CSIFLDNM    DC CL8'VOLSER  '       FIELD NAME
DATAREC  DS    CL8Ø
```

```
SAVE     DS    18F
DDNAMEAD DS    A                   ADDRESS OF THE DDNAME
DDWORKNA DS    CL8                 WORK DATASET DDNAME
JCLREC   DS    CL8Ø
JOBCARD1 DC    CL7Ø'//HZTOFFRE JOB (HZT,SW),(SPOOL),CLASS=F,         '
JOBCARD2 DC    CL7Ø'//   MSGCLASS=T,NOTIFY=FSSØ3                      '
COMMANDØ DC    CL7Ø'// COMMAND                                        '
COMMAND1 DC    CL7Ø'// COMMAND  #$TOFF1.SR,WS=(Q/),JOBNAME=           '
COMMAND2 DC    CL7Ø'// COMMAND  #$TOFF1.SR,Q=                         '
COMMAND3 DC    CL7Ø'// COMMAND  #$TOFFLOAD1,DSN=                      '
COMMAND4 DC    CL7Ø'// COMMAND  #$SOFFLOAD1,TYPE=RECEIVE              '
STEP1EXE DC    CL7Ø'//IEFBR14  EXEC PGM=IEFBR14                       '
         DC    CL7Ø'//SYSPRINT DD SYSOUT=*                            '
         DC    CL7Ø'//* SPOOL OFFLOAD RECEIVE FROM CARTS              '
COMMENT1 DC    CL7Ø'//*                                               '
JOBLENGT EQU   (*-JOBCARD1)/7Ø     NUMBER OF JCL 7Ø BYTES LONG CARDS
EOF      DC    CL5'/*EOF'
SINGQUOT DC    XL1'7D'
         DS    ØD
PRINTDCB DCB   MACRF=PT,RECFM=FBA,LRECL=133,BLKSIZE=133,DSORG=PS,      *
               DDNAME=PRINTOUT
SUBMIDCB DCB   DSORG=PS,MACRF=(PM),RECFM=FB,LRECL=8Ø,BLKSIZE=8ØØØ
         LTORG
WORKAREA DS    ØF
         DC    F'32ØØØ'            LENGTH DECLARED EXPLICIT
         DS    XL32ØØØ
DATADSEC DSECT
CSIUSRLN DS    F
CSIREQLN DS    F
CSIUSDLN DS    F
CSINUMFD DS    H
*  INFORMATION RETURNED FOR EACH ENTRY
CSICFLG DS     CL1
CSICTYPE DS    CL1
CSICNAME DS    CL44
CSICRETN DS    F     CSICRETN
DSECTEND DS    ØF
MAPENTRY DSECT
*  INFORMATION RETURNED FOR EACH ENTRY
CSIEFLAG DS    XL1   CSIEFLAG
CSIETYPE DS    XL1   CSIETYPE
CSIENAME DS    CL44  CSIENAME
EDATALN  DS    XL2
EFLD1LN  DS    XL2
EFLD2LN  DS    XL2
VOLSENTR DS    XL4
MAPEND   DS    ØXL1
VARIDSEC DSECT   DUMMY SECTION
SAVE3    DS    18F
         DCBD  DSORG=PS            DUMMY SECTION
         END
*    DYNAMIC ALLOCATION OF THE INTERNAL READER DATA SET               *
RØ       EQU   Ø
...
R15      EQU   15
DYINTRDR CSECT
         USING *,R1Ø,R11           ESTABLISH ADDRESSABILITY
         STM   R14,R12,12(R13)     SAVE3 REGISTERS
         LR    R1Ø,R15             SET FIRST BASE REGISTER
         LA    R11,2Ø48(R1Ø)       SET SECOND BASE REGISTER
         LA    R11,2Ø48(R11)       AND INCREMENT TO PROPER VALUE
         LR    R12,R13             STORE PREVIOUS SA ADDRESS
         LR    R2,R1  (R2) = POINTER TO ADDRESS OF THE PARM LIST
         LA    R13,SAVE3           LOAD ADDRESS OF THIS SAVE3 AREA
         ST    R12,SAVE3+4         CHAIN BACKWARDS
         ST    R13,8(R12)          CHAIN FORWARD
*            STORE ADDRESS OF THE DDWORKNA
         L     R3,Ø(R2)    (R3) = ADDRESS OF THE FIRST  PARAMETER
         LA    R1,DDWORKNA (R1)=ADDRESS OF THE DDWORKNA - OUTPUT PARAM
         ST    R1,Ø(R3)    STORE ADDRESS OF THE DDWORKNA
*            ESTABLISH DYNALLOC PARAMETERS
         LA    RØ,3ØØ
         GETMAIN R,LV=(RØ)
         LR    R8,R1
         USING S99RBP,R8
         LA    R4,S99RBPTR+4
         USING S99RB,R4
         ST    R4,S99RBPTR
         OI    S99RBPTR,S99RBPND
         XC    S99RB(RBLEN),S99RB
         MVI   S99RBLN,RBLEN
*                                    VERB CODE Ø1 ----------------
         MVI   S99VERB,S99VRBAL      REQUEST FOR DSNAME ALLOCATION
         LA    R5,S99RB+RBLEN
         USING S99TUPL,R5
         ST    R5,S99TXTPP
         LA    R6,S99TUPL+16 POINT JUST PAST THE FOUR  TEXT UNITS PTRS
*           1.ST TEXT UNIT - KEY: SYSOUT DATA SET AND ITS CLASS
         USING S99TUNIT,R6
         ST    R6,S99TUPTR
         LA    R7,DALSYSOU          SYSOUT
         STH   R7,S99TUKEY            2
         LA    R7,1
         STH   R7,S99TULNG            2
         STH   R7,S99TUNUM            2
         MVI   S99TUPAR,C'T'   CLASS  1
         LA    R6,S99TUNIT+7          7 = TOTAL
*           2.ND TEXT UNIT - THE SYSOUT PROGRAM NAME SPECIFICATION
         LA    R5,S99TUPL+4
         ST    R6,S99TUPTR
         LA    R7,DALSPGNM         PROGRAM NAME
         STH   R7,S99TUKEY             2
         LA    R7,1
         STH   R7,S99TUNUM             2
         LA    R7,6
         STH   R7,S99TULNG             2
         MVC   S99TUPAR(6),=C'INTRDR'  6
         LA    R6,S99TUNIT+12         12 = TOTAL
*           3.RD TEXT UNIT - DEALLOCATION AT CLOSE
         LA    R5,S99TUPL+4
         ST    R6,S99TUPTR
         LA    R7,DALCLOSE     CLOSE
         STH   R7,S99TUKEY             2
         LA    R7,Ø
         STH   R7,S99TUNUM             2
         LA    R6,S99TUNIT+4           4 = TOTAL
*           4.TH TEXT UNIT - KEY: RETURN DDWORKNA
         LA    R5,S99TUPL+4
         ST    R6,S99TUPTR
         OI    S99TUPTR,S99TUPLN
         LA    R7,DALRTDDN
         STH   R7,S99TUKEY
         LA    R7,1
         STH   R7,S99TUNUM
         LA    R7,8
         STH   R7,S99TULNG
         LR    R1,R8
         DYNALLOC
         LR    R7,R15
         LTR   R15,R15
         BZ    OKDYNALL
         LA    R1,4
         CR    R1,R7
         BNE   DYRCNEQ4
         B     FREEM
DYRCNEQ4 EQU   *
         LA    R1,8
         CR    R1,R7
         BNE   DYRCNEQ8
         B     FREEM
DYRCNEQ8 EQU   *
         LA    R1,12
         CR    R1,R7
         BNE   DYRCNE12
         LH    R7,S99ERROR
         B     FREEM
DYRCNE12 EQU   *
         B     FREEM
OKDYNALL EQU   *
         LA    R3,S99TUPAR
         LA    R2,8
LOOPTUPA EQU   *
         TM    Ø(R3),B'11ØØØØØØ'
         BO    OKALPNUM            OK ALPHANUMERIC
         MVI   Ø(R3),C' '
OKALPNUM EQU   *
         A     R3,=F'1'
         BCT   R2,LOOPTUPA
         MVC   DDWORKNA(8),S99TUPAR
FREEM    EQU   *
         FREEMAIN R,LV=3ØØ,A=(R8)
         L     R13,4(R13)
         LR    R15,R7
         RETURN (14,12),RC=(15)
*   CONSTANTS AND STORAGE
SAVE3    DS    18F
*   DYNALLOC CONSTANTS AND VARIABLES
DDWORKNA DS    CL8
         LTORG LTORG LTORG LTORG LTORG LTORG LTORG LTORG LTORG
         IEFZB4DØ                  DUMMY SECTION
         IEFZB4D2                  DUMMY SECTION
RBLEN    EQU   (S99RBEND-S99RB)
         DCBD  DSORG=PS            DUMMY SECTION
         END
* PANEL PNLOFFLØ:
)ATTR
> type(text) attn(on)
¬ area(dynamic) extend(on) scroll(on)
$ type(dataout) intens(high)
@ type(dataout) intens(low)
)BODY
%------------- SOFF - Spool Offload JSE Services Menu ------------------
%OPTION ===>_ZCMD                                     +SCROLL ===>_PSCR%
%                                                                      %
+ Welcome to the exciting Spool World!                +USERID - &ZUSER %
+ You are not alone.                                  +TIME   - &ZTIME %
+
+Specify the spool offload function to be performed and press <ENTER>
%                                                                      %
+
+ ¬DYNAREA                                                           ¬ +
+                                                                      +
+ Enter END command to terminate.                                      +
)INIT
  .HELP    = ISPØØØØ5     /* Help for this master menu  CBIPO      */
  &LINLEN = 68            /* Length of dynamic area lines          */
  &ZHTOP = ISRØØØØ3       /* Tutorial table of contents            */
  &ZHINDEX = ISR91ØØØ     /* Tutorial Index - first page           */
&MENU = '+
$ R @Receive @- Upload contents of the cart(s)$to JES2 Spool        +
$ T @Transmit@- Offload specific classes from JES2 Spool$to cart(s) +
$                                                                   +
$ X @EXIT        @- Exit from Spool Offload Menu                    '
IF (&CUTP = ' ') &CUTP = Ø    /* Initialize to display top of menu */
IF (&PSCR = ' ') &PSCR = PAGE /* Initialize scroll amount          */
&JUNK = TRUNC(&MENU,&CUTP)    /* Truncate menu at cut-off point    */
&DYNAREA = .TRAIL             /* Portion of menu to be displayed   */
)PROC
  &LASTLN = LVLINE(DYNAREA)   /* Last visable line of dynamic area */
  IF (&ZCMD ¬= ' ')           /* Make sure ZCMD value does not     */
    &ZQ = TRUNC(&ZCMD,'.')    /*  begin with a period              */
    IF (&ZQ = ' ')
      .MSG = ISPD241
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
  /**** Transmit to offload to the cart(s)    **********/
  T,'
RØ       EQU   Ø
....
R15      EQU   15
SUOFTRAN CSECT
         USING *,R1Ø,R11           ESTABLISH ADDRESSABILITY
         STM   R14,R12,12(R13)     SAVE3 REGISTERS
         LR    R1Ø,R15             SET FIRST BASE REGISTER
         LA    R11,2Ø48(R1Ø)       SET SECOND BASE REGISTER
         LA    R11,2Ø48(R11)       AND INCREMENT TO PROPER VALUE
         LR    R12,R13             STORE PREVIOUS SA ADDRESS
GETMA1   GETMAIN R,LV=4ØØ
         LR    R9,R1 (R9) = ADDR. OF THE ALLOCATED VIRTUAL STORAGE AREA
         LTR   15,15
         BZ    OKGETMA1
         LA    R15,4
         B     ENDRET
OKGETMA1 EQU   *
         USING VARIDSEC,R9
         LA    R13,SAVE3           LOAD ADDRESS OF THIS SAVE3 AREA
         ST    R12,SAVE3+4         CHAIN BACKWARDS
         ST    R13,8(R12)          CHAIN FORWARD
         OPEN  (PRINTDCB,(OUTPUT))
         MVC   FILE(8),=C'SUPEFILE'
*            DEFINE VARIABLE FILE FOR DIALOG SERVICE
*            RECN
         CALL  ISPLINK,(VDEFINE,NRECN,RECN,FIXED,LRECN),VL
*            MEMB
         CALL  ISPLINK,(VDEFINE,NMEMB,MEMB,CHAR,LMEMB),VL
*            VOLSER
         CALL  ISPLINK,(VDEFINE,NVOLSER,VOLSER,CHAR,LVOLSER),VL
*            CATLG
         CALL  ISPLINK,(VDEFINE,NCATLG,CATLG,CHAR,LCATLG),VL
*            DSN
         CALL  ISPLINK,(VDEFINE,NDSN,DSN,CHAR,LDSN),VL
```

```
*            DSNM
         CALL  ISPLINK,(VDEFINE,NDSNM,DSNM,CHAR,LDSNM),VL
*            SDATE
         CALL  ISPLINK,(VDEFINE,NSDATE,SDATE,CHAR,LSDATE),VL
*            STIME
         CALL  ISPLINK,(VDEFINE,NSTIME,STIME,CHAR,LSTIME),VL
*            SEL
         CALL  ISPLINK,(VDEFINE,NSEL,SEL,CHAR,LSEL),VL
*            RETURN CODE
         CALL  ISPLINK,(VDEFINE,NRC,RC,FIXED,LRC),VL
*            REPLY
         CALL  ISPLINK,(VDEFINE,NR,R,CHAR,LR),VL
DIFIPANE LA    R15,Ø
         ST    R15,RC
         ST    R15,RECN
         MVC   MEMB(8),BLANK
         MVC   VOLSER(6),BLANK
         MVC   CATLG(1),BLANK
         MVC   R,BLANK
         MVI   SEL,C' '
         CALL  ISPLINK,(DISPLAY,PNLOFTR1),VL
         LTR   R15,R15
         BNZ   ENDPRO3
         MVC   P
```

Szczepan Kowalski
Johannesburg Stock Exchange (Republic of South Africa)

E-mail alerts

If you would like to be notified when new issues of *MVS Update* have been placed on our Web site, you can sign up for our e-mail alert service, which notifies you when new issues (including new free issues) have been placed on our Web site. To sign up, go to http://www.xephon.com/mvsupdate.html and click the *Receive an e-mail alert* link.

A utility for record tailoring

THE PROBLEM

System programmers and storage administrators have to undertake considerable administration in their everyday work. This is further increased during migration to a new operating system, implementing a new project, or standards upgrade. These tasks are often resolved by generating statements for different utilities or elements of the JCL. When the requirements are more complex, we generate statements with a specific REXX procedure. Simple record tailoring can be done with ICETOOL.

Coding REXX procedures is always a time-consuming and error-prone process, although it typically consists of a few IF - THEN - ELSE statements and a few variables. Practice taught us that it is more convenient to have tailoring statements embedded in the job than to have these in a separate procedure. ICETOOL is not flexible enough because we can only manage fixed parts of the input dataset (from position, in length), which is not enough in some cases.

A SOLUTION

We wrote a utility which we have called ‘Tailor’ to make record tailoring easier. We have used this utility in many administration tasks. During record tailoring, we often need a way of using variable-length information from the input records using the following parameters:

- From a position to a constant
- From a constant to a constant
- From a constant for a specified length.

These facilities make Tailor a powerful tool for record tailoring. ‘Tailor’ uses the following datasets:

- IN – input sequential datasets
- OUT – output sequential dataset for tailored records

- SYSIN – dataset for tailoring parameters
- SYSPRINT – dataset for messages.

TAILOR tailors input records in a way defined by the parameters that have the syntax shown in Figure 1. Expressions consist of comparisons linked by the following logical operators:

- & – logical and
- ! – logical or.

A comparison operation is specified by combining operands with one of the following operators:

- < – less than
- <= – less than or equal to
- = – equal to
- ¬= – not equal to
- >= – greater than or equal to
- > – greater than
- IN – left string is in right string
- NI – left string is not in right string
- CO – right string contains left string
- NC – right string does not contain left string.

You can write comments in the parameter dataset. An asterisk in the first position marks a comment. SEGMENTs in the tailoring definition can be of the following types:

- position, length – part of the input record that is copied from a specified position for a specified length. Length 0 means to the end of the input record.
- position, constant – part of the input record is copied from the specified position to the specified constant, excluding the constant.
- constant, constant – part of the input record is copied starting from the first constant to the second constant excluding both of them.

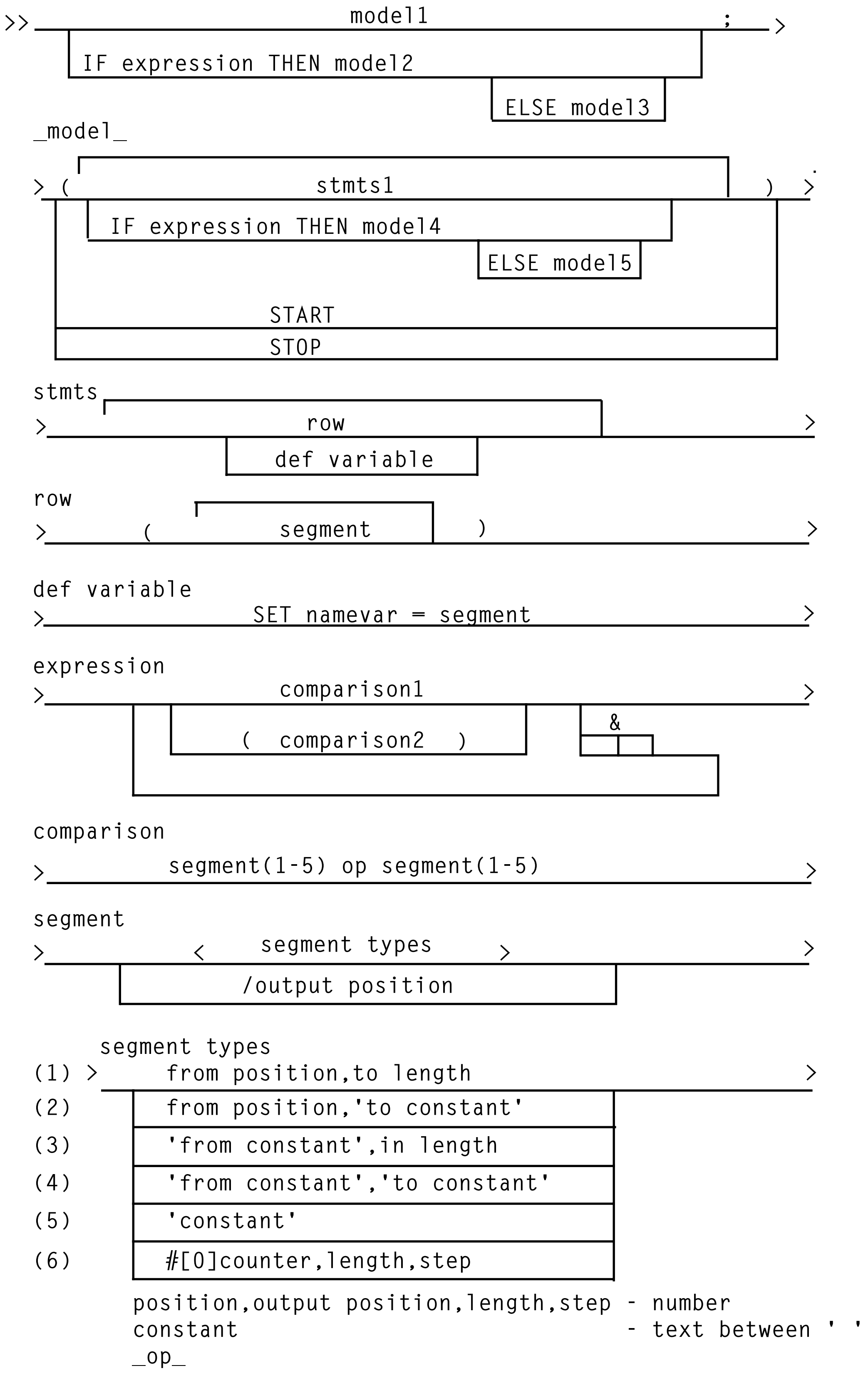

Figure 1: The Tailor syntax summary

- constant, length – part of the input record is copied starting from the constant for a specified length, excluding constant. Length 0 means to the end of the input record.
- constant – this constant will be placed in the output record.
- #[0]beginning of the numeration, length, step – program will set numeration in character format for a specified length, starting from the specified beginning and incrementing by step. If 0 is set after the #, numbers are printed with the left zeroes. For example #01,3,2 gives numbers 001 003 005 and so on.
- set variable = <some type from (1) to (6)>
- &variable name – referring to variable defined by set statement.
- START – keyword that means the beginning of the record tailoring. We use it in conditional statements only. Its purpose is to give the record for tailoring with a condition. START specifies the beginning of the block when we have a statement that generates multiple conditions.
- STOP – keyword specifies the end of record tailoring. It makes sense in conditional statements only. When we use it in statements that generate multiple records, we must specify it at the end.

Using brackets can change the standard hierarchy between operations. All numbers which are specified as a position must be positive; lengths must be >= 0. When we specify length=0 in the tailoring parts it means that we want to move everything from the specified beginning to the end of the record to the output record.

In the condition part, length=0 means that the condition can be satisfied anywhere from the specified beginning to the end of the input record.

The following examples provide practical indications of some of the potential uses of Tailor.

EXAMPLE 1

List information from all catalogs, make tailored output consisting of dataset name, DASD volume, and catalog:

```
//useridP   JOB MSGCLASS=X,MSGLEVEL=(2,1),NOTIFY=&SYSUID,CLASS=A
//LISTC1    EXEC   PGM=IDCAMS,COND=EVEN
//SYSPRINT  DD SYSOUT=X
//CAT       DD DSN=&&USERCAT,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=VB,LRECL=136,BLKSIZE=Ø),
//          SPACE=(TRK,(5Ø,2Ø),RLSE)
//SYSIN     DD *
  DELETE userid.#CATCONT.LIST
  DELETE userid.#CATALL.LIST
  SET MAXCC=Ø
  LISTCAT USERCATALOG OFILE(CAT) -
          CAT(CATALOG.OS5ICFM.VOS5CAT)
/*
//TAILOR    EXEC   PGM=TAILOR,REGION=ØK
//STEPLIB   DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=X
//IN        DD DSN=&&USERCAT,DISP=(SHR,DELETE)
//OUT       DD DSN=&&PARAM,DISP=(NEW,PASS),
//          UNIT=SYSDA,DCB=(RECFM=FB,LREC1L=8Ø,BLKSIZE=312Ø),
//          SPACE=(TRK,(1,1))
//SYSIN     DD *
      IF (<2,6> = <'USERCA'>)
      THEN ((<' LISTCAT CATALOG('><18,Ø><') ALL  OFILE(CAT)'>));
/*
//LISTC2    EXEC   PGM=IDCAMS,REGION=ØK
//SYSPRINT  DD SYSOUT=X
//CAT       DD DSN=userid.#CATCONT.LIST,DISP=(MOD,CATLG),
//          UNIT=SYSDA,DCB=(RECFM=VB,LRECL=136,BLKSIZE=Ø,BUFNO=15),
//          SPACE=(CYL,(55Ø,35Ø),RLSE)
//SYSIN     DD *
   LISTCAT CAT(CATALOG.OS5ICFM.VOS5CAT) ALL OFILE(CAT)
/*
//          DD DSN=&&PARAM,DISP=SHR
//*
//CATLIST   EXEC  PGM=TAILOR,COND=(9,LT)
//STEPLIB   DD DSN=userid.USER.LOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=X
//IN        DD DSN=userid.#CATCONT.LIST,DISP=SHR
//OUT       DD DSN=userid.#CATALL.LIST,DISP=(NEW,CATLG,KEEP),
//           UNIT=SYSDA,DCB=(RECFM=VB,LRECL=16Ø,BLKSIZE=66Ø4),
//           SPACE=(CYL,(1Ø,5Ø),RLSE)
//SYSIN     DD *
  IF (<1,Ø> CO <' CATALOG --'>)
  THEN (SET CAT = <' CATALOG --',Ø>  ) ;
  IF (<2,3> ¬= <'   '> & <1,1> ¬= <'1'>)
  THEN (SET DSNAME =<18,Ø>
        SET TYPE = <2,' '>
        SET NUM = <#Ø1,5,1>          ) ;
  IF (<1,Ø> CO <'DEVTYPE-'>)
  THEN ( SET DEVT =<'????'>
         IF (<1,Ø> CO <'3Ø1Ø2ØØC'>)
         THEN ( SET DEVT =<'3375'>)
         IF (<1,Ø> CO <'3Ø1Ø2ØØE'>)
         THEN ( SET DEVT =<'338Ø'>)
         IF (<1,Ø> CO <'3Ø1Ø2ØØF'>)
         THEN ( SET DEVT =<'339Ø'>)
         IF (<1,Ø> CO <'ØØØ22ØØØ'>)
         THEN ( SET DEVT =<'SYSDA'>)
         IF (<1,Ø> CO <'78ØØ8Ø8Ø'>)
         THEN ( SET DEVT =<'348Ø'>)
         IF (<1,Ø> CO <'78Ø48Ø8Ø'>)
         THEN ( SET DEVT =<'359Ø'>)
         IF (<1,Ø> CO <'78Ø48Ø83'>)
         THEN ( SET DEVT =<'359Ø'>)
        ) ;

  IF (<9,7> = <'VOLSER-'>)
  THEN (SET VOL =<27,6>
        (<&NUM> <' '> <&DSNAME> /51 <&TYPE> /64 <&VOL> <' '> <&DEVT>
        /77 <&CAT>));
/*
//
```

EXAMPLE 2

Renaming multiple datasets:

```
//useridC JOB (ACCT#),'D.N',
//           NOTIFY=&SYSUID,
//           CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//        EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//OUT       DD DSN=&&LIST,DISP=(MOD,PASS),
//           UNIT=SYSDA,DCB=(RECFM=VB,LRECL=136,BLKSIZE=792Ø),
//           SPACE=(TRK,(1,1),RLSE)
//SYSPRINT  DD SYSOUT=X
//SYSIN      DD *
  LISTC  LEVEL(applid) NAME OFILE(OUT)
/*
//        EXEC PGM=TAILOR
//STEPLIB DD DSN=userid.USER.LOAD,DISP=SHR
//IN      DD UNIT=SYSDA,DISP=(SHR,PASS),DSN=&&LIST
//OUT     DD DSN=&&LISTM,DISP=(NEW,CATLG,DELETE),
//        UNIT=SYSDA,DCB=(RECFM=FB,LRECL=8Ø,BLKSIZE=8ØØØ),
//        SPACE=(TRK,(1,1),RLSE)
//SYSPRINT DD SYSOUT=X
//SYSIN DD *
* dataset name beginning with applid is renamed to start with newapplid
 IF (<1,Ø> CO <'applid.'>)
 THEN ((<' ALTER '><18,Ø><' - '>)
      (<'   NEWNAME('><'newapplid.'><'applid.',0><')'>));
/*
//ALTER    EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=X
//SYSPRINT  DD SYSOUT=X
//SYSIN     DD DSN=&&LISTM,DISP=(SHR,PASS)
//
```

EXAMPLE 3

The following are input records:

```
IDCAMS  SYSTEM SERVICES     TIME: 10:36:01        10/01/90     PAGE   1
NONVSAM ------- USERID.ISPF.ISPPROF
     IN-CAT --- CATALOG.MVSICF1.VMVSTS1
NONVSAM ------- USERID.TEST
     IN-CAT --- CATALOG.MVSICF1.VMVSTS1
NONVSAM ------- USERID.USER.CLIST
     IN-CAT --- CATALOG.MVSICF1.VMVSTS1
NONVSAM ------- USERID.USER.CNTL
     IN-CAT --- CATALOG.MVSICF1.VMVSTS1
```

Control parameters specified the following action:

```
IF (<2,7> = <'NONVSAM'>)
THEN ((<'//DDIN'><#01,1,3><' DD DSN='><18,' '><',DISP=SHR'>));
```

We get the following output records:

```
//DIN001 DD DSN=USERID.ISPF.ISPPROF,DISP=SHR
//DIN002 DD DSN=USERID.TEST,DISP=SHR
//DIN003 DD DSN=USERID.ISPF.CLIST,DISP=SHR
//DIN004 DD DSN=USERID.ISPF.CNTL,DISP=SHR
```

TAILOR SOURCE

```
TAILOR:PROC OPTIONS(MAIN);
/*-------------------------------------------------------------------
                    Program For Record Tailoring

  The following LL(1) grammar is formed based on the syntax diagram.
  Grammar recognizes input parameters and generates an internal tree
  management structure for record tailoring based on it.

Syntax checking is realized on the following grammar:
   <START>      -> <MODEL><NEXT_MODEL>
   <NEXT_MODEL> -> <MODEL><NEXT_MODEL>
   <NEXT_MODEL> -> NULL
-----------------------------------------------------------------
   <MODEL>                   -> 'IF' <CONDITIONALY_MODEL>
   <MODEL>                   -> <UNCONDITIONALY_MODEL> ';'
   <MODEL>                   -> NULL
   <CONDITIONALY_MODEL> -> <CONDITION>
                                'THEN' <UNCONDITIONALY_MODEL>
                                'ELSE' <UNCONDITIONALY_MODEL>
   <UNCONDITIONALY_MODEL> -> '(' <STATEMENTS> <NEXT_STATEMENTS> ')'
   <UNCONDITIONALY_MODEL> -> START
   <UNCONDITIONALY_MODEL> -> STOP
   <UNCONDITIONALY_MODEL> -> END
-----------------------------------------------------------------
   <STATEMENTS>           ->  <ROWS>
   <STATEMENTS>           -> 'IF' <CONDITIONALY_ROWS>
   <CONDITIONALY_ROWS>    -> <CONDITION>
                                'THEN' <UNCONDITIONALY_MODEL>
                                'ELSE' <UNCONDITIONALY_MODEL>
   <NEXT_STATEMENTS>      ->  <STATEMENTS> <NEXT_STATEMENTS>
   <NEXT_STATEMENTS>      ->  NULL
-----------------------------------------------------------------
   <ROWS>           -> ( <ROW> )
   <ROWS>           -> SET <DEF_VARIJABLE>
   <NEXT_ROWS>      -> <ROWS><NEXT_ROWS>
   <NEXT_ROWS>      -> NULL
-----------------------------------------------------------------
   <ROW>            -> <SEGMENT><NEXT_SEGMENT>
   <NEXT_SEGMENT>   -> <SEGMENT><NEXT_SEGMENTS>
   <NEXT_SEGMENT>   -> NULL
   <DEF_VARIABLE>   -> name_variable = <SEGMENT>
   <SEGMENT>        -> '<' <SEGMENT_TYPES> '>'
   <SEGMENT>        -> '/' <OUTPUT_POSITION>'
-----------------------------------------------------------------
   <CONDITION>        -> '(' <COMPARASION><NEXT_CONDITION> ')'
   <NEXT_CONDITION>   -> '&' <COMPARASION><NEXT_CONDITION>
   <NEXT_CONDITION>   -> '|' <COMPARASION><NEXT_CONDITION>
   <NEXT_CONDITION>   -> NULL
   <COMPARASION>      -> '(' <COMPARASION><NEXT_CONDITION> ')'
   <COMPARASION>      -> <UNCONDITIONALY_SEGMENT>
   'IN' | 'NI' | 'CO' | 'NC' | '<' | '<=' | '=' | '¬=' | '>' | '>='
                         <UNCONDITIONALY SEGMENT>
-----------------------------------------------------------------
 SEGEMENT TYPES
    (1) FROM POSITION,TO LENGTH
    (2) FROM POSITION,'TO CONSTANT'
    (3) 'FROM CONSTANT',IN LENGTH
    (4) 'FROM CONSTANT','TO CONSTANT'
    (5) 'CONSTANT'
    (6) #[Ø]COUNTER,LENGTH,STEP
    (7) &NAME VARIJABLE
---------------------------------------------------------------------*/
/*********************************************************************/
/*            DATASETS                                               */
/*********************************************************************/
DCL IN         FILE RECORD SEQL INPUT,
    SYSIN      FILE RECORD SEQL INPUT;
DCL OUT        FILE STREAM OUTPUT;
/*********************************************************************/
/*            WORKING VARIABLES                                      */
/*********************************************************************/
DCL 1 NODE          BASED(PTR_NODE),
     2 NEXT_NODE    PTR INIT(NULL),
     2 NEXT_LEVEL   PTR INIT(NULL),
     2 TYPE_NODE    CHAR(1) INIT(' ');
DCL PCHV            CHAR(32767) VAR BASED;
DCL PCHF            CHAR(32767) BASED;
DCL PBINFIXED       BIN FIXED BASED(PBF);
DCL RECORD_IN       CHAR(32ØØØ)  VAR,
    1 RECORDIN      BASED(ADDR(RECORD_IN)),
     2 LENRECIN     BIN FIXED,
     2 CH(32ØØØ)    CHAR(1);
DCL (PI_SEGMENTG    INIT(NULL), PBF,
     PTR_NODEG      INIT(NULL),
     PPTR_NODE      INIT(ADDR(PTR_NODEG)),
     PTR_NODE,
     PTR_VARG       INIT(NULL),
     PPTR_VAR       INIT(ADDR(PTR_VARG))) PTR;
DCL IND_TAILORING BIT INIT('1'B);       /* TAILORING INDICATOR       */
DCL NUMBER_LEV      BIN FIXED INIT(Ø);  /* NUMBER OF LEVELS          */
DCL NUMBER_OUT      BIN FIXED INIT(Ø);  /* NUMBER OF OUTPUT LINES    */
DCL LEN_REC_IN      BIN FIXED INIT(72); /* LENGTH OF INPUT RECORD    */
DCL RECORD_OUT      CHAR(32767)  VAR INIT('');
DCL LENRECOUT       BIN FIXED    BASED(ADDR(RECORD_OUT));
DCL CONSTANT1       CHAR(16)   INIT('/&#$''Ø123456789');
DCL CONSTANT2       CHAR(1Ø)   INIT('=|&#$¬()/''');
DCL (NOT_EOF        INIT('1'B),
     NOT_EOFT       INIT('1'B),
     INDP           INIT('Ø'B),
     INDD           INIT('Ø'B)) BIT;
DCL NPIC            PIC'(1Ø)9',
    CHN(1Ø)         CHAR(1) BASED(ADDR(NPIC));
DCL NPICZ           PIC'(1Ø)-',
    CHNZ(1Ø)        CHAR(1) BASED(ADDR(NPICZ));
DCL NUM_REC_IN      BIN FIXED(31) INIT(Ø),
    NUM_REC_OUT     BIN FIXED(31) INIT(Ø);
/*********************** BUILTIN FUNCTIONS *************************/
DCL (ADDR, SUBSTR, INDEX, NULL, LENGTH) BUILTIN;
/********************* ON CONDITIONS *******************************/
ON ERROR SNAP BEGIN;
              ON ERROR SYSTEM;
              PUT SKIP DATA(i,RECORD_IN,RECORD_OUT);
              END;
ON ENDFILE(SYSIN) NOT_EOFT='Ø'B;
ON ENDFILE(IN) NOT_EOF='Ø'B;
/***************** SYNTAX ANALYSIS OF PARAMETERS *******************/
I=#NEXT_NON_BLANK#(133);
```

```
DO WHILE(NOT_EOFT);
   IF SUBSTR(RECORD_IN,I,4) = 'TEST'
   THEN DO;
        PUT SKIP EDIT('>>> TAILOR ENDS - TEST REASON') (A);
        RETURN;
        END;
   ELSE
   IF SUBSTR(RECORD_IN,I,5) = 'PRINT'
   THEN DO;
        INDP='1'B;
        I=I+5;
        END;
   ELSE
   IF SUBSTR(RECORD_IN,I,4) = 'LIST'
   THEN DO;
        CALL LIST((PTR_NODEG),1);
        IF PTR_VARG ¬= NULL
        THEN CALL LIST((PTR_VARG),1);
        I=I+4;
        END;
   ELSE
   IF SUBSTR(RECORD_IN,I,2) = 'IF' | CH(I) = '('
   THEN DO;
        PPTR_NODE->NODE.NEXT_NODE = #START#;
        PPTR_NODE = PPTR_NODE->NODE.NEXT_NODE;
        END;
   ELSE CALL TAILOR_ERROR('1'B,Ø,'(,IF,START,STOP,PRINT,LIST,END');
   I=#NEXT_NON_BLANK#(I);
   IF CH(I) = '5E'X
   THEN I=#NEXT_NON_BLANK#(I+1);
END;
/************************ RECORD TAILORING *************************/
READ FILE(IN)    INTO(RECORD_IN);
DO WHILE(NOT_EOF);
 IF INDD THEN PUT SKIP DATA(RECORD_IN);/*##*/
   NUMBER_LEV=Ø;
   NUM_REC_IN=NUM_REC_IN+1;
   IF IND_TAILORING
   THEN CALL EXEC_TAILOR((PTR_NODEG),'1'B);
   ELSE IND_TAILORING =
        CHECKING(((PTR_NODEG->NODE.NEXT_LEVEL)->NODE.NEXT_LEVEL));
   READ FILE(IN) INTO(RECORD_IN);
END;
KRAJ:
PUT SKIP EDIT(' ### IN:',NUM_REC_IN,'       OUT:',NUM_REC_OUT) (A);

/**********************************************************************/
/* PROCEDURE FINDS NEXT NON-BLANK SYMBOL IN PARAMETERS                */
/* AND SKIP COMMENTS                                                  */
/**********************************************************************/
#NEXT_NON_BLANK#: PROCEDURE(J) RETURNS(BIN FIXED);
  DCL J BIN FIXED;
  DO UNTIL(¬ NOT_EOFT | J <= LEN_REC_IN);
     IF J > LEN_REC_IN & NOT_EOFT
     THEN DO;
          READ FILE(SYSIN) INTO(RECORD_IN);
          LEN_REC_IN= MIN(72,LENGTH(RECORD_IN));
          IF NOT_EOFT
          THEN PUT SKIP EDIT(RECORD_IN) (A);
          J=1;
          END;
     DO J=J TO LEN_REC_IN WHILE(CH(J)=' ');
     END;
     IF J=1 & CH(J) = '*' /* SKIP COMMENT */
     THEN J=LEN_REC_IN+1;
  END;
RETURN(J);
END #NEXT_NON_BLANK#;
/**********************************************************************/
/* INSERT OF NODE INTO INTERNAL TREE STRUCTURE                        */
/**********************************************************************/
#INSERT_NODE#: PROCEDURE(V,PREVIOS,NEXTLEVEL,NEXTNODE) RETURNS(PTR);
DCL V CHAR(1);
DCL (PREVIOS,NEXTLEVEL,NEXTNODE) PTR;
   ALLOC NODE;
   NODE.TYPE_NODE=V;
   NODE.NEXT_LEVEL=NEXTLEVEL;
   NODE.NEXT_NODE=NEXTNODE;
   IF PREVIOS ¬= NULL
   THEN PREVIOS->NODE.NEXT_NODE = PTR_NODE;
   RETURN(PTR_NODE);
END #INSERT_NODE#;
/**********************************************************************/
/* INSERT of the character field                                      */
/**********************************************************************/
#INSERT_CHAR#: PROCEDURE(PNTR_FIRST,CP,PNTR_NEXT) RETURNS(PTR);
DCL CP  CHAR(*);
DCL DCP BIN FIXED INIT(LENGTH(CP));
DCL 1 PPCHAR BASED,
      2 LEN_FIELD BIN FIXED,
      2 FIELD    CHAR(DCP REFER(LEN_FIELD));
DCL (PNTR_FIRST,PNTR,PNTR_NEXT) PTR;
   ALLOC PPCHAR SET(PNTR);
   PNTR->PPCHAR.FIELD = CP;
   PNTR_FIRST = #INSERT_NODE#('C',PNTR_FIRST,PNTR,PNTR_NEXT);
   RETURN(PNTR_FIRST);
END #INSERT_CHAR#;
/**********************************************************************/
/* INSERT OF BIN_FIXED FIELD                                          */
/**********************************************************************/
#INSERT_BINF#: PROCEDURE(V,PNTR_FIRST,BFP) RETURNS(PTR);
DCL V CHAR(1);
DCL BFP BIN FIXED;
DCL (PNTR_FIRST,PNTR) PTR;
   ALLOC PBINFIXED SET(PNTR);
   PNTR->PBINFIXED = BFP;
   PNTR_FIRST = #INSERT_NODE#(V,PNTR_FIRST,PNTR,NULL);
   RETURN(PNTR_FIRST);
END #INSERT_BINF#;

/*********************************************************************/
/* PROCEDURES FOR SYNTAX ANALYSIS BASED ON GRAMMAR                   */
/*********************************************************************/

#START#: PROCEDURE RETURNS(PTR);
DCL PNTR PTR;
   PNTR=#INSERT_NODE#('M',NULL,#MODEL#,#NEXT_MODEL#);
RETURN(PNTR);
END #START#;

#NEXT_MODEL#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
 IF NOT_EOFT &
    (CH(I) = '(' |
     SUBSTR(RECORD_IN,I,5) = 'START' |
     SUBSTR(RECORD_IN,I,4) = 'STOP'  |
     SUBSTR(RECORD_IN,I,3) = 'END')
 THEN PNTR=#INSERT_NODE#('M',NULL,#MODEL#,#NEXT_MODEL#);
 ELSE PNTR=NULL;
RETURN(PNTR);
END #NEXT_MODEL#;
#MODEL#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
   I=#NEXT_NON_BLANK#(I);
IF ¬NOT_EOFT
THEN PNTR=NULL;
ELSE IF SUBSTR(RECORD_IN,I,2) = 'IF'
     THEN PNTR=#CONDITIONALY_MODEL#;
     ELSE PNTR=#UNCONDITIONALY_MODEL#;
IF NOT_EOFT & CH(I) ¬= '5E'X
THEN CALL TAILOR_ERROR('Ø'B,Ø6,'5E'X);
ELSE I=#NEXT_NON_BLANK#(I+1);
RETURN(PNTR);
END #MODEL#;
#CONDITIONALY_MODEL#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL (PNTR INIT(NULL),PNTR1) PTR;
 I=#NEXT_NON_BLANK#(I+2);
PNTR1=ADDR(PNTR);
CALL #CONDITION#(PNTR,PNTR1);
PNTR,PNTR1=#INSERT_NODE#('?',NULL,PNTR,NULL);
I=#NEXT_NON_BLANK#(I);
IF SUBSTR(RECORD_IN,I,4) = 'THEN'
THEN DO;
     I=#NEXT_NON_BLANK#(I+4);
     PNTR1=#INSERT_NODE#('T',PNTR1,#UNCONDITIONALY_MODEL#,NULL);
     END;
IF PNTR = PNTR1
THEN CALL TAILOR_ERROR('1'B,Ø2,'THEN');
IF SUBSTR(RECORD_IN,I,4) = 'ELSE'
THEN DO;
     I=#NEXT_NON_BLANK#(I+4);
     PNTR1=#INSERT_NODE#('E',PNTR1,#UNCONDITIONALY_MODEL#,NULL);
     END;
IF PNTR = PNTR1
THEN CALL TAILOR_ERROR('1'B,Ø3,'ELSE');
RETURN(PNTR);
END #CONDITIONALY_MODEL#;

#UNCONDITIONALY_MODEL#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR INIT(NULL);
DCL BRZAG BIN FIXED INIT(Ø);
DCL BRKOSC BIN FIXED INIT(Ø);
IF CH(I) = '('
THEN DO;
     BRZAG=BRZAG+1;
     I=#NEXT_NON_BLANK#(I+1);
     PNTR=#INSERT_NODE#('R',NULL,#STATEMENTS#,#NEXT_STATEMENTS#);
     I=#NEXT_NON_BLANK#(I);
     IF CH(I) ¬= ')'
     THEN CALL TAILOR_ERROR('1'B,Ø4,')');
     I=#NEXT_NON_BLANK#(I+1);
     BRZAG=BRZAG-1;
     IF BRZAG > Ø
     THEN CALL TAILOR_ERROR('1'B,Ø5,') - UNBALANCED PARENTHESIS');
     END;
ELSE
IF SUBSTR(RECORD_IN,I,5) = 'START'
THEN DO;
     PNTR=#INSERT_NODE#('{',NULL,NULL,NULL);
     IND_TAILORING='Ø'B;
     I=#NEXT_NON_BLANK#(I+5);
     END;
ELSE
IF SUBSTR(RECORD_IN,I,4) = 'STOP'
THEN DO;
     PNTR=#INSERT_NODE#('}',NULL,NULL,NULL);
     I=#NEXT_NON_BLANK#(I+4);
     END;
ELSE
IF SUBSTR(RECORD_IN,I,3) = 'END'
THEN DO;
     PNTR=#INSERT_NODE#('¬',NULL,NULL,NULL);
     I=#NEXT_NON_BLANK#(I+3);
     END;
RETURN(PNTR);
END #UNCONDITIONALY_MODEL#;
#STATEMENTS#: PROCEDURE RETURNS(PTR) RECURSIVE;
```

```
DCL PNTR PTR;
   I=#NEXT_NON_BLANK#(I);
IF ¬NOT_EOFT
THEN PNTR=NULL;
ELSE IF SUBSTR(RECORD_IN,I,2) = 'IF'
     THEN PNTR=#CONDITIONALY_ROWS#;
     ELSE PNTR=#ROWS#;
RETURN(PNTR);
END #STATEMENTS#;
#NEXT_STATEMENTS#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
 IF NOT_EOFT & (CH(I) = '(' |
                SUBSTR(RECORD_IN,I,2) = 'IF' |
                SUBSTR(RECORD_IN,I,3) = 'SET')
 THEN PNTR=#INSERT_NODE#('R',NULL,#STATEMENTS#,#NEXT_STATEMENTS#);
 ELSE PNTR=NULL;
RETURN(PNTR);
END #NEXT_STATEMENTS#;
#CONDITIONALY_ROWS#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL (PNTR INIT(NULL),PNTR1) PTR;
I=#NEXT_NON_BLANK#(I+2);
PNTR1=ADDR(PNTR);
CALL #CONDITION#(PNTR,PNTR1);
PNTR,PNTR1=#INSERT_NODE#('?',NULL,PNTR,NULL);
I=#NEXT_NON_BLANK#(I);
IF SUBSTR(RECORD_IN,I,4) = 'THEN'
THEN DO;
     I=#NEXT_NON_BLANK#(I+4);
     PNTR1=#INSERT_NODE#('T',PNTR1,#UNCONDITIONALY_MODEL#,NULL);
     END;
IF PNTR = PNTR1
THEN CALL TAILOR_ERROR('1'B,13,'THEN');
IF SUBSTR(RECORD_IN,I,4) = 'ELSE'
THEN DO;
     I=#NEXT_NON_BLANK#(I+4);
     PNTR1=#INSERT_NODE#('E',PNTR1,#UNCONDITIONALY_MODEL#,NULL);
     END;
IF PNTR = PNTR1
THEN CALL TAILOR_ERROR('1'B,14,'ELSE');
RETURN(PNTR);
END #CONDITIONALY_ROWS#;
#ROWS#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
DCL BRZAG BIN FIXED INIT(Ø);
IF SUBSTR(RECORD_IN,I,3) = 'SET'
THEN DO;
     I=#NEXT_NON_BLANK#(I+3);
     PNTR=#DEF_VARIJABLE#;
     END;
ELSE
IF CH(I) = '('
THEN PNTR=#ROW#;
```

```
ELSE CALL TAILOR_ERROR('Ø'B,Ø7,'SET OR (');
RETURN(PNTR);
END #ROWS#;

#NEXT_ROWS#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
 IF NOT_EOFT & (CH(I) = '(' | SUBSTR(RECORD_IN,I,3) = 'SET')
 THEN PNTR=#INSERT_NODE#('R',NULL,#ROWS#,#NEXT_ROWS#);
 ELSE PNTR=NULL;
RETURN(PNTR);
END #NEXT_ROWS#;

#ROW#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
DCL BRZAG BIN FIXED INIT(Ø);
IF CH(I) = '('
THEN DO;
     BRZAG=BRZAG+1;
     I=#NEXT_NON_BLANK#(I+1);
     PNTR=#INSERT_NODE#('S',NULL,#SEGMENT#,#NEXT_SEGMENT#);
     IF CH(I) ¬= ')'
     THEN CALL TAILOR_ERROR('1'B,Ø8,')');
     I=#NEXT_NON_BLANK#(I+1);
     BRZAG=BRZAG-1;
     IF BRZAG > Ø
     THEN CALL TAILOR_ERROR('1'B,Ø9,') - UNBALANCED PARENTHESIS');
     END;
RETURN(PNTR);
END #ROW#;

#SEGMENT#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
DCL BRZAG BIN FIXED INIT(Ø);
IF CH(I) = '<'
THEN DO;
     BRZAG=BRZAG+1;
     I=#NEXT_NON_BLANK#(I+1);
     PNTR=#SEGMENT_TYPES#;
     IF CH(I) ¬= '>'
     THEN CALL TAILOR_ERROR('1'B,1Ø,'>');
     I=#NEXT_NON_BLANK#(I+1);
     BRZAG=BRZAG-1;
     IF BRZAG > Ø
     THEN CALL TAILOR_ERROR('1'B,11,' - UNBALANCED < >');
     END;
ELSE
IF CH(I) = '/'
THEN PNTR=#OUTPUT_POSITION#;
ELSE CALL TAILOR_ERROR('1'B,12,'  <');
RETURN(PNTR);
END #SEGMENT#;
```

```
#NEXT_SEGMENT#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
 IF NOT_EOFT & (CH(I) = '<' | CH(I) = '/')
 THEN PNTR=#INSERT_NODE#('S',NULL,#SEGMENT#,#NEXT_SEGMENT#);
 ELSE PNTR=NULL;
RETURN(PNTR);
END #NEXT_SEGMENT#;

#DEF_VARIJABLE#: PROC RETURNS(PTR) RECURSIVE;
DCL VAR_NAME CHAR(16) INIT('');
DCL (PNTR,WORK_PTR,WORK_PTR1) PTR;
I=#NEXT_NON_BLANK#(I);
IF VAR_NAME = ' '
THEN DO;
   DO J = I TO I+15 WHILE(INDEX(CONSTANT2,CH(J)) = 0); END;
   IF CH(J) ¬= '='
   THEN CALL TAILOR_ERROR('1'B,15,'=');
   ELSE DO K= J-1 TO I BY - 1 WHILE(CH(K) = ' '); END;
   IF K-I > 15
   THEN CALL TAILOR_ERROR('1'B,16,'VARIABLE NAME MUST BE 1-16 CHAR');
   VAR_NAME=SUBSTR(RECORD_IN,I,K-I+1);
   I=#NEXT_NON_BLANK#(J+1);
END;
DO PTR_NODE=PTR_VARG REPEAT(NODE.NEXT_NODE) WHILE(PTR_NODE ^= NULL)
                                   UNTIL(WORK_PTR1->PCHV = VAR_NAME);
   WORK_PTR = NODE.NEXT_LEVEL;
   WORK_PTR1 = WORK_PTR->NODE.NEXT_LEVEL;
END;
IF PTR_NODE = NULL
THEN DO;
     /* INSERT OF VARIABLE INTO NEXT VARIABLE */
     PNTR = #INSERT_CHAR#(NULL,VAR_NAME,NULL);
     PPTR_VAR = #INSERT_NODE#('V',PPTR_VAR,PNTR,NULL);
     END;
ELSE PNTR=WORK_PTR;
/* INSERT OF VARIABLE INTO MODEL*/
PNTR=#INSERT_NODE#('V',NULL,PNTR,#SEGMENT#);
RETURN(PNTR);
END #DEF_VARIJABLE#;

#SEGMENT_TYPES#: PROCEDURE RETURNS(PTR) RECURSIVE;
DCL PNTR PTR;
I=#NEXT_NON_BLANK#(I);
SELECT(CH(I));
  WHEN('''')    PNTR=#TYPE_CONSTANTS#;
  WHEN('#')     PNTR=#TYPE_NUMBERS#;
  WHEN('&')     PNTR=#TYPE_VARIJABLE#;
  OTHERWISE     PNTR=#TYPE_POZ_LEN#;
END;
I=#NEXT_NON_BLANK#(I);
RETURN(PNTR);
END #SEGMENT_TYPES#;
```

```
#OUTPUT_POSITION#: PROC RETURNS(PTR);
DCL (PNTR,PNTR_LAST) PTR;
DCL J BIN FIXED;
I=#NEXT_NON_BLANK#(I);
PNTR,PNTR_LAST = #INSERT_NODE#('/',NULL,NULL,NULL);
I=#NEXT_NON_BLANK#(I+1);
PNTR_LAST = #INSERT_BINF#('P',PNTR_LAST,#NUMBER#);
RETURN(PNTR);
END #OUTPUT_POSITION#;

#TYPE_CONSTANTS#: PROC RETURNS(PTR);
DCL (PNTR,PNTR_LAST) PTR;
DCL (P,D) BIN FIXED;
I=#NEXT_NON_BLANK#(I);
PNTR,PNTR_LAST = #INSERT_NODE#('5',NULL,NULL,NULL);
CALL #CONSTANT#(P,D);
PNTR_LAST=#INSERT_CHAR#(PNTR_LAST,SUBSTR(RECORD_IN,P,D),NULL);
IF CH(I) = ','
THEN DO;
     I=#NEXT_NON_BLANK#(I+1);
     IF CH(I)=''''
     THEN DO;
          /* THIS IS FROM CONSTANT TO CONSTANT */
          PNTR->NODE.TYPE_NODE = '4';
          CALL #CONSTANT#(P,D);
          PNTR_LAST=#INSERT_CHAR#(PNTR_LAST,
                                  SUBSTR(RECORD_IN,P,D),NULL);
          END;
     ELSE DO;
          D = #NUMBER#;
          IF D ¬= -1
          THEN DO;
               /* THIS IS FROM CONSTANT IN LENGTH */
               PNTR->NODE.TYPE_NODE ='3';
               PNTR_LAST=#INSERT_BINF#('D',PNTR_LAST,D);
               END;
          END;
     END;
RETURN(PNTR);
END #TYPE_CONSTANTS#;
#TYPE_NUMBERS#: PROC RETURNS(PTR);
DCL (PNTR,PNTR_LAST) PTR;
DCL J BIN FIXED;
I=#NEXT_NON_BLANK#(I);
I=I+1;
PNTR,PNTR_LAST = #INSERT_NODE#(' ',NULL,NULL,NULL);
IF CH(I) = 'Ø'
THEN DO;
     PNTR->NODE.TYPE_NODE = '#';
     I=#NEXT_NON_BLANK#(I+1);
     END;
```

```
ELSE PNTR->NODE.TYPE_NODE = '$';
DO J= 1 TO 3;
   PNTR_LAST = #INSERT_BINF#('P',PNTR_LAST,#NUMBER#);
   IF CH(I) = ','
   THEN I=#NEXT_NON_BLANK#(I+1);
END;
RETURN(PNTR);
END #TYPE_NUMBERS#;

#TYPE_VARIJABLE#: PROC RETURNS(PTR);
DCL (PNTR,PNTR_LAST,WORK_PTR,WORK_PTR1) PTR;
DCL VAR_NAME CHAR(16) VAR;
I=#NEXT_NON_BLANK#(I);
PNTR,PNTR_LAST = #INSERT_NODE#('&',NULL,NULL,NULL);
I=#NEXT_NON_BLANK#(I+1);
DO J=I+1 TO I+ 8 WHILE(INDEX(' =,>)/',CH(J))=Ø);
END;
VAR_NAME = SUBSTR(RECORD_IN,I,J-I);
DO PTR_NODE=PTR_VARG REPEAT(NODE.NEXT_NODE) WHILE(PTR_NODE ¬= NULL)
                                   UNTIL(WORK_PTR1->PCHV = VAR_NAME);
   WORK_PTR = NODE.NEXT_LEVEL;
   WORK_PTR1 = WORK_PTR->NODE.NEXT_LEVEL;
END;
IF PTR_NODE = NULL
THEN CALL TAILOR_ERROR('1'B,17,
              '"'||VAR_NAME||'"'||' - VARIABLE IS NOT DEFINED');
PNTR->NODE.NEXT_LEVEL=WORK_PTR;
I=#NEXT_NON_BLANK#(J);
RETURN(PNTR);
END #TYPE_VARIJABLE#;

#TYPE_POZ_LEN#: PROC RETURNS(PTR);
DCL (PNTR,PNTR_LAST) PTR;
DCL (P,D) BIN FIXED;
I=#NEXT_NON_BLANK#(I);
/* THIS IS FROM POSITION IN LENGTH */
PNTR,PNTR_LAST = #INSERT_NODE#('1',NULL,NULL,NULL);
PNTR_LAST = #INSERT_BINF#('P',PNTR_LAST,#NUMBER#);
IF CH(I) = ','
THEN I=#NEXT_NON_BLANK#(I+1);
ELSE CALL TAILOR_ERROR('1'B,18,',');
IF CH(I)=''''
THEN DO;
     PNTR->NODE.TYPE_NODE='2'; /* THIS IS FROM POSITION TO CONSTANT  */
     CALL #CONSTANT#(P,D);
     PNTR_LAST=#INSERT_CHAR#(PNTR_LAST,SUBSTR(RECORD_IN,P,D),NULL);
     END;
ELSE PNTR_LAST = #INSERT_BINF#('D',PNTR_LAST,#NUMBER#);
RETURN(PNTR);
END #TYPE_POZ_LEN#;

#CONSTANT#: PROC(POZ,LEN);
```

```
DCL (POZ,LEN,D) BIN FIXED;
I=I+1;
POZ=1;
DO D=I TO LEN_REC_IN UNTIL(CH(D) = '''' & CH(D+1) ¬='''' & POZ=Ø);
   IF CH(D) = ''''
   THEN POZ=1-POZ;
END;
/* TWO APOSTROPHES ARE CONVERTED INTO ONE */
LEN=D;
POZ=INDEX(SUBSTR(RECORD_IN,I,LEN-I),'''''');
DO WHILE(POZ > Ø);
   SUBSTR(RECORD_IN,I+POZ,LEN-I-POZ)=
          SUBSTR(RECORD_IN,I+POZ+1,LEN-I-POZ-1);
   LEN=LEN-1;
   POZ=INDEX(SUBSTR(RECORD_IN,I,LEN-I),'''''');
END;
LEN=LEN-I;
POZ=I;
I=#NEXT_NON_BLANK#(D+1);
END #CONSTANT#;

#NUMBER#: PROC RETURNS(BIN FIXED);
DCL (P,D) BIN FIXED;
I=#NEXT_NON_BLANK#(I);
IF INDEX('Ø123456789',CH(I)) > Ø
THEN DO;
     DO D=I+1 TO LEN_REC_IN WHILE(INDEX('Ø123456789',CH(D)) > Ø);
     END;
     GET STRING(SUBSTR(RECORD_IN,I,D-I)) LIST(P);
     I=#NEXT_NON_BLANK#(D);
     END;
ELSE CALL TAILOR_ERROR('1'B,19,' NUMBER ');
RETURN(P);
END #NUMBER#;
/*********************************************************************/
/*     PROCEDURES FOR ANALYSIS OF LOGICAL CONDITIONS                 */
/*********************************************************************/
#CONDITION#: PROCEDURE(PNTR,PNTR_LAST);
DCL (PNTR, PNTR_LAST) PTR;
I=#NEXT_NON_BLANK#(I);
IF CH(I) = '('
THEN DO;
     I=#NEXT_NON_BLANK#(I+1);
     CALL #COMPARASION#(PNTR,PNTR_LAST);
     I=#NEXT_NON_BLANK#(I);
     IF CH(I)='&' | CH(I)='|'
     THEN CALL #NEXT_COMPARASION#(PNTR,PNTR_LAST);
     IF CH(I) = ')'
     THEN I=#NEXT_NON_BLANK#(I+1);
     ELSE CALL TAILOR_ERROR('1'B,2Ø,')');
     END;
```

```
ELSE CALL TAILOR_ERROR('1'B,21,'(');
END #CONDITION#;

#NEXT_COMPARASION#: PROCEDURE(PNTR,PNTR_LAST) RECURSIVE;
DCL (PNTR,PNTR1,PNTR_LAST) PTR;
PNTR=#INSERT_CHAR#(NULL,CH(I),PNTR);
I=#NEXT_NON_BLANK#(I+1);
CALL #COMPARASION#(PNTR,PNTR_LAST);
IF (CH(I)='&' | CH(I)='|')
THEN CALL #NEXT_COMPARASION#(PNTR,PNTR_LAST);
END #NEXT_COMPARASION#;

#COMPARASION#: PROCEDURE(PNTR,PNTR_LAST) RECURSIVE;
DCL (PNTR,PNTR1,PNTR2,PNTR_LAST) PTR;
DCL OPER CHAR(2);
IF CH(I)='('
THEN DO;
     PNTR2=ADDR(PNTR1);
     CALL #NEXT_COMPARASION#(PNTR1,PNTR2);
     PNTR_LAST->NODE.NEXT_NODE=PNTR1;
     PNTR_LAST=PNTR2;
     END;
ELSE DO;
     PNTR1=#INSERT_NODE#('S',NULL,#SEGMENT#,NULL);
     DO J=I+1 TO LEN_REC_IN UNTIL(CH(J) = ' ' | CH(J) = '/');
     END;
     IF INDEX('<=< > >= ¬=INNICONC',SUBSTR(RECORD_IN,I,J-I)) = Ø
     THEN CALL TAILOR_ERROR('1'B,22,'<,<=,=,>=,>,¬=,IN,NI,CO,NC');
     OPER = SUBSTR(RECORD_IN,I,J-I);
     PNTR_LAST=#INSERT_CHAR#(PNTR_LAST,OPER,PNTR1);
     I=#NEXT_NON_BLANK#(J);
     PNTR_LAST=#INSERT_NODE#('S',PNTR1,#SEGMENT#,NULL);
     END;
END #COMPARASION#;
/*********************************************************************/
/*                        TAILORING PROCEDURES                       */
/*********************************************************************/
EXEC_TAILOR: PROC(PNTR,IND) RECURSIVE;
DCL PNTR PTR INTERNAL;
DCL IND BIT;
IF PNTR ¬= NULL
THEN DO;
     SELECT(PNTR->NODE.TYPE_NODE);
     WHEN('?') IND=CHECKING((PNTR->NODE.NEXT_LEVEL));
     WHEN('T','E') IF (( IND & PNTR->NODE.TYPE_NODE = 'T' ) |
                      ( ¬IND & PNTR->NODE.TYPE_NODE = 'E' ) )
                   THEN CALL EXEC_TAILOR((PNTR->NODE.NEXT_LEVEL),IND);
     WHEN('S') CALL CUT_PASTE((PNTR->NODE.NEXT_LEVEL));
     WHEN('V') CALL NEW_VARIABLE(PNTR);
     WHEN('{') IND_TAILORING = IND;
     WHEN('}') IND_TAILORING = ¬IND;
     WHEN('¬') NOT_EOF = ¬IND;
     OTHERWISE CALL EXEC_TAILOR((PNTR->NODE.NEXT_LEVEL),IND);
     END;

     IF NOT_EOF & IND_TAILORING & PNTR->NODE.TYPE_NODE ¬= 'V'
     THEN DO;
          IF PNTR->NODE.TYPE_NODE = 'R' & LENGTH(RECORD_OUT) > Ø
          THEN DO;
               PUT FILE(OUT) SKIP EDIT(RECORD_OUT) (A);
               NUM_REC_OUT=NUM_REC_OUT+1;
               IF INDP
               THEN PUT SKIP EDIT(NUM_REC_OUT,' ',RECORD_OUT) (A);
               RECORD_OUT='';
               END;
          IF PNTR->NODE.NEXT_NODE ¬= NULL
          THEN CALL EXEC_TAILOR((PNTR->NODE.NEXT_NODE),IND);
          END;
     END;
END EXEC_TAILOR;

CHECKING: PROC(PNTR) RETURNS(BIT) RECURSIVE;
DCL PNTR PTR, (D1,D2) BIN FIXED;
DCL PNTR1 PTR;
DCL (IND,IND1) BIT;
PNTR1=PNTR->NODE.NEXT_LEVEL;
IF PNTR1->PCHV = '&' |   /* LOGICAL OPERATORS */
   PNTR1->PCHV = '|'
THEN DO;
     PNTR=PNTR->NODE.NEXT_NODE;
     IND =CHECKING(PNTR);
     PNTR = PNTR->NODE.NEXT_NODE;
     IND1=CHECKING(PNTR);
     SELECT(PNTR1->PCHV);
     WHEN('&') IND=IND & IND1;
     WHEN('|') IND=IND | IND1;
     END;
     END;
ELSE IND=COMPARASION(PNTR);
RETURN(IND);
END CHECKING;

COMPARASION: PROC(PNTR) RETURNS(BIT);
DCL (PNTR,PNTR1,PNTR2,PNTR1C,PNTR2C) PTR;
DCL OPER CHAR(2);
DCL (P1,P2,D1,D2) BIN FIXED,IND BIT;
DCL 1 PP   BASED,
      2 P   BIN FIXED,
      2 D   BIN FIXED,
      2 PCH PTR;

PNTR1=PNTR->NODE.NEXT_LEVEL;
OPER = PNTR1->PCHV;
```

```
/* LEFT */
PNTR=PNTR->NODE.NEXT_NODE;
PNTR1=PNTR->NODE.NEXT_LEVEL;
PNTR1=SEGMENT(PNTR1);
P1=PNTR1->PP.P;
D1=PNTR1->PP.D;
PNTR1C=PNTR1->PP.PCH;
/* RIGHT */
PNTR=PNTR->NODE.NEXT_NODE;
PNTR2=PNTR->NODE.NEXT_LEVEL;
PNTR2=SEGMENT(PNTR2);
P2=PNTR2->PP.P;
D2=PNTR2->PP.D;
PNTR2C=PNTR2->PP.PCH;

SELECT(OPER);
WHEN('IN','NI')
  DO;
       IND = (INDEX(SUBSTR(PNTR2C->PCHF,P2,D2),
                    SUBSTR(PNTR1C->PCHF,P1,D1)) > Ø);
  IF OPER = 'NI'
  THEN IND = ¬IND;
  END;
WHEN('CO','NC')
  DO;
       IND = (INDEX(SUBSTR(PNTR1C->PCHF,P1,D1),
                    SUBSTR(PNTR2C->PCHF,P2,D2)) > Ø);
  IF OPER = 'NC'
  THEN IND = ¬IND;
  END;
WHEN('= ','¬=')
  DO;
  IND = (SUBSTR(PNTR1C->PCHF,P1,D1) = SUBSTR(PNTR2C->PCHF,P2,D2));
  IF OPER = '¬='
  THEN IND = ¬IND;
  END;
WHEN('< ')
    IND = (SUBSTR(PNTR1C->PCHF,P1,D1) < SUBSTR(PNTR2C->PCHF,P2,D2));
WHEN('<=')
    IND = (SUBSTR(PNTR1C->PCHF,P1,D1) <= SUBSTR(PNTR2C->PCHF,P2,D2));
WHEN('> ')
    IND = (SUBSTR(PNTR1C->PCHF,P1,D1) > SUBSTR(PNTR2C->PCHF,P2,D2));
WHEN('>=')
    IND = (SUBSTR(PNTR1C->PCHF,P1,D1) > SUBSTR(PNTR2C->PCHF,P2,D2));
END; /* SELECT */
FREE PNTR1->PP;
FREE PNTR2->PP;
RETURN(IND);
END COMPARASION;
SEGMENT: PROC(PNTR) RETURNS(PTR);
DCL (PNTR,PNTRB,PNTR1) PTR, (P INIT(Ø),D,I) BIN FIXED;
DCL TIP CHAR(1);
TIP = PNTR -> NODE.TYPE_NODE;
PNTR1=PNTR;
PNTR=PNTR->NODE.NEXT_NODE;
SELECT(TIP);
WHEN('5') DO; /* CONSTANT */
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          D=LENGTH(PNTRB->PCHV);
          PNTRB=SUB_STRING(3,D,PNTRB);
          END;
WHEN('1') DO; /* FROM POSITION IN LENGTH */
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          P = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          D = PNTRB->PBINFIXED;
          IF D = Ø
          THEN DO;
               IF P <= LENGTH(RECORD_IN)
               THEN I= LENGTH(RECORD_IN)-P+1;
               ELSE I= Ø;
               END;
          ELSE I= D;
          PNTRB=SUB_STRING(P+2,I,ADDR(RECORD_IN));
          END;
WHEN('2') DO; /* FRIOM POSITION TO  CONSTANT */
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          P = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          D=INDEX_CONSTANTS(P,PNTRB->PCHV) - 1;
          IF D < Ø THEN D = LENGTH(RECORD_IN) - P + 1;
          PNTRB=SUB_STRING(P+2,D,ADDR(RECORD_IN));
          END;
WHEN('4') DO; /* FROM CONSTANT TO CONSTANT */
          PNTRB = PNTR->NODE.NEXT_LEVEL;
          P=INDEX_CONSTANTS(1,PNTRB->PCHV);
          IF P ¬= Ø
          THEN DO;
               P=P+LENGTH(PNTRB->PCHV);
               PNTR=PNTR->NODE.NEXT_NODE;
               PNTRB = PNTR->NODE.NEXT_LEVEL;
               D=INDEX_CONSTANTS(P,PNTRB->PCHV) - 1;
               IF D < Ø
               THEN IF P <= LENGTH(RECORD_IN)
                    THEN D = LENGTH(RECORD_IN)-P+1;
                    ELSE D = Ø;
               END;
          ELSE D = Ø;
          PNTRB=SUB_STRING(P+2,D,ADDR(RECORD_IN));
          END;
WHEN('3') DO; /* FROM CONSTANT IN LENGTH */
```

```
          PNTRB = PNTR->NODE.NEXT_LEVEL;
          P=INDEX_CONSTANTS(1,PNTRB->PCHV);
          IF P ¬= Ø
          THEN DO;
               P=P+LENGTH(PNTRB->PCHV);
               PNTR=PNTR->NODE.NEXT_NODE;
               PNTRB = PNTR->NODE.NEXT_LEVEL;
               D = PNTRB->PBINFIXED;
               IF D = Ø
               THEN DO;
                    IF P <= LENGTH(RECORD_IN)
                    THEN I= LENGTH(RECORD_IN)-P+1;
                    ELSE I= Ø;
                    END;
               ELSE I= D;
               END;
          ELSE I,D = Ø;
          PNTRB=SUB_STRING(P+2,I,ADDR(RECORD_IN));
          END;
WHEN('&') DO; /* VARIABLE */
          PNTRB=PNTR1->NODE.NEXT_LEVEL;
          PNTRB=PNTRB->NODE.NEXT_NODE;
          IF PNTRB ¬= NULL
          THEN D=LENGTH(PNTRB->PCHV);
          ELSE D=Ø;
          PNTRB=SUB_STRING(3,D,PNTRB);
          END;
WHEN('#') DO; /* NUMERATION WITH LEADING ZEROES */
          PNTRB,PNTR1=PNTR->NODE.NEXT_LEVEL;
          NPIC = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          D = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB = PNTR->NODE.NEXT_LEVEL;
          PNTR1->PBINFIXED=NPIC+PNTRB->PBINFIXED;
          PNTRB=SUB_STRING(11-D,D,ADDR(NPIC));
          END;
WHEN('$') DO; /* NUMERATION WITH LEADING BLANKS */
          PNTRB,PNTR1=PNTR->NODE.NEXT_LEVEL;
          NPICZ = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB=PNTR->NODE.NEXT_LEVEL;
          D = PNTRB->PBINFIXED;
          PNTR=PNTR->NODE.NEXT_NODE;
          PNTRB = PNTR->NODE.NEXT_LEVEL;
          PNTR1->PBINFIXED=NPICZ+PNTRB->PBINFIXED;
          PNTRB=SUB_STRING(11-D,D,ADDR(NPICZ));
          END;
WHEN('/') DO; /* NEXT OUTPUT POSITION */
          PNTRB=NULL;
          END;
```

```
OTHERWISE DO;
          PUT SKIP DATA(TIP);
          STOP;
          PNTRB=NULL;
          END;
END;
RETURN(PNTRB);
END SEGMENT;

NEW_VARIABLE: PROC(PNTR); /* FORMING OF NEW VARIABLE */
DCL (PNTR,PNTR1,PNTR2,PPP,PP1) PTR;
DCL DCP BIN FIXED;
DCL 1 PP   BASED,
      2 P   BIN FIXED,
      2 D   BIN FIXED,
      2 PCH PTR;
DCL 1 PPCHAR BASED,
      2 LEN_FIELD BIN FIXED,
      2 FIELD    CHAR(DCP REFER(LEN_FIELD));
PNTR1= PNTR->NODE.NEXT_LEVEL;
PNTR2= PNTR1->NODE.NEXT_NODE;
IF PNTR2 ¬= NULL
THEN FREE PNTR2->PPCHAR;
PNTR2= PNTR->NODE.NEXT_NODE;
PPP=SEGMENT((PNTR2));
DCP=PPP->PP.D;
PP1=PPP->PP.PCH;
ALLOC PPCHAR SET(PNTR2);
PNTR2->FIELD=SUBSTR(PP1->PCHF,PPP->PP.P,PPP->PP.D);
PNTR1->NODE.NEXT_NODE=PNTR2;
FREE PPP->PP;
END NEW_VARIABLE;

CUT_PASTE: PROC(PNTR);
DCL (PNTR,PNTR1) PTR;
DCL (IP,POZ,LEN) BIN FIXED;
DCL 1 PP   BASED,
      2 P   BIN FIXED,
      2 D   BIN FIXED,
      2 PCH PTR;
PNTR1= SEGMENT((PNTR));
IF PNTR1 ¬= NULL
THEN DO;
     POZ=PNTR1->PP.P;
     LEN=PNTR1->PP.D;
     PNTR1C=PNTR1->PP.PCH;
     IF PNTR1C ¬= NULL
     THEN DO;
          IF POZ > Ø & LEN > Ø
          THEN RECORD_OUT=RECORD_OUT||SUBSTR(PNTR1C->PCHF,POZ,LEN);
          END;
     FREE PNTR1->PP;
     END;
ELSE DO;  /* PROCESSING OUTPUT POSITION */
     PNTR=PNTR->NODE.NEXT_NODE;
     IF PNTR ¬= NULL
     THEN DO;
          PNTR=PNTR->NODE.NEXT_LEVEL;
          IP=PNTR->PBINFIXED;
          IF IP <= LENGTH(RECORD_OUT)
          THEN DO;
               J=LENGTH(RECORD_OUT)-IP+1;
               IF SUBSTR(RECORD_OUT,IP,J) ¬= ' '
               THEN PUT SKIP EDIT('### WARNING - OUTPUT POSITION',IP,
                        ' OVERWRITES SEGMENT OF OUTPUT RECORD',
                        RECORD_OUT, REPEAT('*',J))
                            (A,A,A,SKIP,A,SKIP,X(IP-1),A);
               END;
          ELSE DO;
               IF IP > LENGTH(RECORD_OUT)
               THEN RECORD_OUT=RECORD_OUT||REPEAT(' ',
                                           IP-LENGTH(RECORD_OUT)-1);
               END;
          END;
     END;
END CUT_PASTE;
/*********************************************************************/
/* FORMING NEW VARIABLE                                              */
/*********************************************************************/
SUB_STRING: PROCEDURE(POZ,LEN,PCHPS) RETURNS(PTR);
DCL (POZ,LEN) BIN FIXED;
DCL (PCHPS,PNTR) PTR;
DCL 1 PP   BASED(PNTR),
      2 P   BIN FIXED,
      2 D   BIN FIXED,
      2 PCH PTR;
   ALLOC PP;
   PNTR->PP.P=POZ;
   PNTR->PP.D=LEN;
   PNTR->PP.PCH=PCHPS;
   RETURN(PNTR);
END SUB_STRING;

INDEX_CONSTANTS: PROCEDURE(POZ,KONS) RETURNS(BIN FIXED);
DCL (POZ,I) BIN FIXED;
DCL KONS CHAR(*) VAR;
IF POZ <= LENGTH(RECORD_IN)
THEN I=INDEX(SUBSTR(RECORD_IN,POZ),KONS);
ELSE I=Ø;
IF I=Ø
THEN PUT SKIP EDIT(' *** CONSTANT >', KONS,
                   '< DOES NOT EXIST FROM POSITION:',POZ,' IN:') (A)
                    (RECORD_IN) (SKIP,A);
```

```
RETURN(I);
END INDEX_CONSTANTS;

/*********************************************************************
  PROCEDURE FOR PRINTING OF ERRORS
 ********************************************************************/
TAILOR_ERROR: PROC(BREAK,NUMBER,TG);
DCL BREAK     BIT,
    NUMBER    BIN FIXED,
    TG        CHAR(*);
   PUT SKIP EDIT(RECORD_IN,'*') (A,SKIP,X(I-1),A);
   IF BREAK
   THEN PUT SKIP EDIT('*** ERROR   ') (A);
   ELSE PUT SKIP EDIT('*** WARNING ') (A);
   PUT  EDIT('TAILOR',NUMBER,' EXPECTED ',TG) (A,P'99',A,A);
   IF BREAK
   THEN STOP;
END TAILOR_ERROR;
/*********************************************************************
  PROCEDURE FOR PARAMETER LIST
 ********************************************************************/
LIST: PROC(PNTR,LEVEL) RECURSIVE;
DCL PNTR PTR INTERNAL, LEVEL BIN FIXED;
   IF PNTR ¬= NULL
   THEN DO;
        PUT SKIP EDIT(LEVEL,PNTR->NODE.TYPE_NODE) (X(LEVEL),F(2),A);
        PTR_NODE=PNTR;
        SELECT(NODE.TYPE_NODE);
        WHEN('P','D')
             IF NODE.NEXT_LEVEL ¬= NULL
             THEN PUT EDIT(' >',NODE.NEXT_LEVEL->PBINFIXED,'<') (A);
        WHEN('C')
             IF NODE.NEXT_LEVEL ¬= NULL
             THEN PUT EDIT(' >',NODE.NEXT_LEVEL->PCHV,'<') (A);
        OTHERWISE;
        END;
        IF NODE.NEXT_LEVEL ¬= NULL & INDEX('PDIC',NODE.TYPE_NODE) = 0
        THEN  CALL LIST((PNTR->NODE.NEXT_LEVEL),LEVEL+1);
        IF PNTR->NODE.NEXT_NODE ¬= NULL
        THEN  CALL LIST((PNTR->NODE.NEXT_NODE),LEVEL);
        END;
END LIST;
END TAILOR;
```

Emina Spasic and Dragan Nikolic
Systems Programmers

Dataset creation date checking in batch

THE PROBLEM

There are some situations where a job in production expects data from another machine every day. For example, at a pre-determined time every day data would be sent from Unix, NT, or some other operating system through FTP or even from other mainframes using XMIT, to the production machine. After receiving the data, a production job will process the data.

However, in some situations the daily data may not get to the mainframe, because of a network problem or some other creation problem. What happens if this daily data does not get to the mainframe can vary. However, there is considerable scope for the production job to use the existing data received the day before. This is easily done if it is a GDG-based dataset, where the production job might use the current generation level of GDG, which may not be the latest one.

This can create considerable problems for the production support people, who can have difficulty finding the problem and resolving it. Much more seriously, the job may finish successfully without any errors, but it might have used the wrong data which it has processed already.

A SOLUTION

In order to avoid the above problem, I have produced two simple REXX routines (CHECKDS and CHECKGDG), which can be run in batch for checking the dataset creation date with the current date for PS/PDS dataset and for GDG dataset.

CHECKDS

```
/* REXX*/
parse upper arg dsnname
a = outtrap(dslist.)
Address "TSO"
```

```
"listds '"dsnname"' history"
a = outtrap(off)
msg = substr(dslist.2,1,9)
 if msg = 'IKJ58503I' then
 do
   say 'Data set NOT found'
   exit(12)
 end
 creationdate = substr(dslist.3,29,8)
 jdate = date('j')
 jyear = '20'||substr(jdate,1,2)
 jday = substr(jdate,3,3)
 jdate = jyear||'.'||jday
 say 'Dataset ('dsnname') creation date :' creationdate
 say 'Today date                        :' jdate
 if creationdate = jdate then exit(0)
 else exit(8)
exit (0)
```

SAMPLE JCL TO RUN CHECKDS

```
//<JOBCARD>
//STEP01  EXEC PGM=IKJEFT01
//SYSPROC DD DSN=<dsnname>,DISP=SHR   ( Dataset name where the above
//SYSPRINT DD SYSOUT=*                program copied
//SYSTSPRT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSTSIN  DD *
 CHECKDS <PS/PDS dataset name>
/*
```

CHECKGDG

```
/* REXX */
 parse upper arg dsnname
 trace i
 a = outtrap(gdglist.)
 Address "TSO"
 "listc ent('"dsnname"')"
 a = outtrap(off)
 lastdsnrow = gdglist.0 - 1
 lastdsn = strip(substr(gdglist.lastdsnrow,17,40))
 msg = substr(gdglist.1,1,8)
 if msg = 'IDC3012I' then
 do
   say 'Data set NOT found'
   exit(12)
 end
 a = outtrap(dslist.)
 "listc ent('"lastdsn"') history"
 a = outtrap(off)
 creationdate = substr(dslist.4,53,8)
 jdate = date('j')
 jyear = '2Ø'||substr(jdate,1,2)
 jday = substr(jdate,3,3)
 jdate = jyear||'.'||jday
 say 'Dataset ('dsnname') creation date :' creationdate
 say 'Today date                        :' jdate
 if creationdate = jdate then exit(Ø)
 else exit(8)
exit (Ø)
```

SAMPLE JCL TO RUN CHECKGDG

```
//<JOBCARD>
//STEPØ1  EXEC PGM=IKJEFTØ1
//SYSPROC DD DSN=<dsnname>,DISP=SHR   ( Dataset name where the above //
SYSPRINT DD SYSOUT=*                  program copied
//SYSTSPRT DD SYSOUT=*
//SYSTSOUT DD SYSOUT=*
//SYSTSIN  DD *
 CHECKGDG <GDG base name>
/*
```

The JCL shown here should be a step before the step which uses the daily data. The above step will set the return code equal to 12 if the specified dataset is not in the catalog. It will set the return code equal to 08 if the specified data set creation date is not the current date, and will set the return code equal to 00 if the specified dataset creation date is the current date. Based on the return code, the decision can be made within the JCL to run or not run the next step. While checking for the GDG dataset, it is enough to specify the GDG base name. The routine will check the creation date of the last generation of the GDG dataset with the current date.

Systems Programmer (UK) © Xephon 2001

Using the mainframe as a file server

FILE SERVING

File sharing is the ability to share files or data in a network with different privileges. This has to be supported by a strong back-up and recovery procedure. Multi-user operating systems such as Unix use NFS (Network File System) for sharing files across networks. NFS has been available for the mainframe for a long time now. However, NFS requires the workstations (clients) to have NFS client software installed, and the mainframe version of NFS is relatively slow. Both these factors are a major drawback, and mitigate against widespread deployment.

Unlike NSF, another product called Samba does not require client software to be installed on client machines. This is because Samba uses Microsoft's SMB protocol which is being used in all versions of Microsoft's Windows operating system. Samba is simple to install, but it is not supported by IBM. IBM's response to this was a product called DFS/SMB. IBM began shipping DFS/SMB with OS/390 Version 2 Release 8. DFS/SMB is similar to Samba, but it has the advantage of being developed and supported by IBM. DFS/SMB is very fast, and when this is considered in association with the mainframes' strong back-up and recovery it makes the mainframe an ideal file server.

DFS/SMB (DISTRIBUTED FILE SERVICE)

Both Samba and DFS/SMB implement the SMB protocol (also known as CIFS – Common Internet File System). SMB is a protocol on the top of NetBIOS over TCP/IP.

DFS/SMB renders file sharing (file serving), but does not have the ability to access the files shared by another SMB server. Prior to OS/390 Version 2 Release 10 it supported only HFS files. But with Release 10, it started supporting conventional OS/390 datasets (known as Record File System – RFS). This article considers sharing HFS files because the Unix file system is similar to the Windows file system in many respects. But, it does not discuss the intricacies of the installation process, rather it acts as a guide and provides an overview of maintanence and support issues.

Not only does DFS/SMB act as a file server, it also provides print sharing and makes OS/390 printers available to Windows workstations (remember that this also requires the OS/390 Infoprint Server).

DFS/SMB is administered and controlled from OS/390. Therefore, there must be a TCP/IP network connection between all the client and server. So the mainframe needs to have TCP/IP, and OMVS (or Unix System Services – USS) should run in full function mode.

Implementation

It is not necessary to go through each step of installation and implementation because it is very clearly documented in the IBM Redbooks. For example, *S/390 File and Print Serving* SG245330 (refer only to the DFS/SMB part). Another useful text is the *Distributed File Service SMB Administration Guide and Reference* (available under DFS bookshelf at the following URL: http://www.s390.ibm.com/os390/bkserv/r10pdf/dfs.html). Separate books are available for each release of OS/390. Make sure you pick up the right one, because IBM has introduced new file sharing functionality with every release of OS/390 (for example, the RFS facility was not available in Release 9).

Once the ServerPac installation of OS/390 is completed, it is easy to run a Unix script. DFS/SMB can be enabled in OS/390 Version 2 Release 7 and Version 2 Release 8 with the application of few sysmods. Remember that the administrator should have UID 0. A brief overview of the installation process is shown below:

1 Run the script file (dfs_cpfiles). Copy configuration files into the /etc/dfs directory.

2 Customize the envar file.

3 Map Windows ID with OS/390 userid in SMBIDMAP

4 devtab.

This indicates the (physical) HFS dataset name to be exported

1 dfstab.

2 smbtab.

3 hfsattr.

4 RACF definitions.

5 Copy started task procedures.

Address spaces

The SMB server has two address spaces.

- DFSKERN – which provides file and print services
- DFSCNTL – which controls other processes.

DFSKERN can be run within the DFSCNTL address space, but IBM recommends that DFSKERN should run in its own address space (controlled by IOE_DAEMONS_IN_AS=DFSKERN in */dfscntl/envar* file). DFSKERN can be stopped or started by DFSCNTL. Another useful process is EXPORT, which is controlled, started, and run within the DFSCNTL address space. 'Export' communicates with DFSKERN and exports file systems for use (to make file systems defined in *xxxtab* files available in the network).

Administration

Once the initial set-up has been completed, the following procedure has to be followed whenever a file system (HFS) is to be shared and made available for SMB clients (workstations):

1 Make an entry in devtab.

2 Add an entry in dfstab.

3 Make an entry in smbtab.

4 Add a mount command for the filesystem (in the started task, jobstep before DFS is started). This step is not mandatory, but strongly recommended.

5 If it is a new user, make an entry in smbidmap and ask the user to issue smbpw to set the Windows password in his/her ID's DCE

profile.

6 Refresh definitions with DFS commands.

Devtab and dfstab help SMB to identify the filesystem, and smbtab identifies share name.

If a shared filesystem has any filesystems mounted under its subdirectory, then you may have to enter those filesystem in devtab and dfstab. This is not required in smbtab because an upper level directory has already been defined.

Refer to the manual for the syntax of xxxtab file entries. The number of concurrent users (for any shared filesystem) can be restricted by smbtab entries. After making changes, it is necessary to issue the following commands (remember to issue OMVS commands in lowercase):

- If there are changes in devtab, dfstab, and smbtab:

  ```
  Dfsshare -all -type ufs
  ```

  This command reads smbtab, starts sharing and exports the related filesystems automatically. Remember that if there are any underlying subdirectories mounted as a filesystem, then these filesystems have to be exported using the 'export' command. Simply defining the parent filesystem in devtab, dfstab is not enough. This is because, when a filesystem is mounted and shared, all the subdirectories created under the filesystem are shared, no matter what the level of hierarchy it is. However, when you mount a separate filesystem under one of its subdirectories, that filesystem has to be defined in devtab, dfstab and exported separately.

- If there are changes in devtab, and dfstab:

  ```
  F DFS,START EXPORT
  (equivalent  OMVS command :  dfsexport  -all)
  ```

  It attempts to export all the filesystems defined in the dfstab file and following messages can be found in the joblog.

  EXPORT joblog (it ignores already attached filesystems and attaches any new filesystems successfully). For example:

```
IOEX18107A Dfsexport: /dev/ufsNN:hfsNN (id nnn): Already attached: cannot reattach
IOEX18110I Dfsexport: /dev/ufsNN:hfsNN:  Attached successfully.
```

- If there are changes in smbidmap:

  ```
  F DFS,SEND DFSKERN,RELOAD,SMBMAP
  ```

  This reads new smbidmap definitions. The following message will be displayed in the DFSKERN joblog:

  ```
  IOEX18217I Sharename XXXXX on device /dev/ufsNN shared successfully
  ```

Security and userid mapping

There should be a mechanism to map Windows IDs with RACF userids. The SMBIDMAP file does this mapping, but it only does the userid mapping, it does not handle password matching. SMB allows a couple of methods to handle passwords. The recommended method is to use encrypted passwords. The following security definitions (refer to the manual) have to be done for enabling encrypted passwords (there is no need to change the Windows registry using this method, which would otherwise be required under other methods).

```
SETR CLASSACT(KEYSMSTR)
SETR CLASSACT(DCEUUIDS)
RDEFINE KEYSMSTR DCE.PASSWORD.KEY SSIGNON(KEYMASKED(16dighexkey))
ALU tsouserid DCE       /* should be done for each and every userid */
```

It is necessary to make this entry in the envar file:

```
  IOE_SMB_CLEAR_PW should be set to NOTALLOWED
```

The SMBPW (should be in lowercase when entered in the Unix shell) command should be invoked by the user to set the Windows login password. This program hashes the password and stores it in the corresponding userid's DCE profile. This will be compared when users access the SMB shares.

If the *IOE_MVS_DFSDFLT* variable for the dfskern process is set to a valid userid, then access will be allowed under this userid when authentication fails (provided the file or directory has permission to view with DFLT ID).

Permission bits cannot be set from Windows workstations. They are set by the following variable in envar when Windows users create a new file. All newly created directories and files (files created from SMB clients) will have the permission bits as directed by following entries.

```
IOE_SMB_DIR_PERMS=700
IOE_SMB_FILE_PERMS=700
```

Filesharing between heterogeneous platforms

Because files are shared between two different platforms, there has to be a conversion process. There is a control file called *hfsattr* that has file extensions and a conversion attribute. This is similar to the Websphere application server's config file httpd.conf. It is pointed to by the *IOE_HFS_ATTRIBUTES_FILE* in envar. For example, it has entries such as:

```
Addtype     .txt     text/html        ebcdic     1.0
Addtype     .gif     image/gif        binary     1.0
```

SMB considers only the second and fourth column (shown in italics).

Experience with DFS/SMB

If you ever want to move your PC directory or file to a mainframe disk, you can do it in a matter of seconds. Just drag your directory and drop it into the shared mainframe directory. This acts like a GUI interface for Unix directories which makes life easier when transferring files between mainframe and desktop.

If a file has to be referred from both a Windows and a Unix environment, then a proper extension has to be given and it should be coded in the hfsattr file. Wordpad should be used to edit any text files, because it writes and views Unix EOF, CR, and LF. You do not need to worry about other files, like Microsoft Word or image files, because they are created and viewed only from the Windows side and they are transferred in binary mode, which resides passively in HFS files).

The Windows Zip application can be used to Zip files in a Unix directory and Unzip these later on. This makes life easier for people who are unfamiliar with the Unix equivalent. Users can disable disable/enable write permissions for an existing file by doing right click and change properties (read only). You can calculate the amount of free space remaining (from the allocated HFS space) from Windows (right click on the mapped drive name and properties), look for a file and you can do all the functions that you would typically do with a mapped drive.

Unix is case sensitive, but when accessing the same files from Windows, it views them differently (because Windows is not case sensitive). Likewise, there are slight differences when referring symbolic links from Windows. These differences are explained in the latest version of the DFS/SMB book.

Shared filesystems (HFS) must be mounted before they are made usable in DFS/SMB. Automount mounts only when the file is referred. Even if it is automount managed, it is good practice to mount them before exporting. The recommended method is to code the IKJEFT01 job step before DFS is started (in the same job or DFS Started task). The IKJEFT01 step may have TSO MOUNT commands for mounting each and every filesystem (HFS) defined in dfstab files. All the filesystems should be mounted in read/write mode.

It is a good practice to create multiple HFS files and mount them under different subdirectories, instead of a single huge HFS. If you ever want to share large amounts of space, it is better to create a number of HFS files and mount them hierarchically to create the same amount of space. This would reduce I/O to individual HFSs and thereby decrease response times.

This requires users to have a userid on the mainframe. When the user changes a Windows login, he/she has to login to OMVS and change the password with the smbpw program. Users who do not have mainframe knowledge may require some training to use it, but it is negligible compared to using NFS-type products.

Each process has its associated envar file. For example, DFSCNTL and DFSKERN have different envar files in their respective directories */home/dfskern/envar* or */home/dfscntl/envar*.

The following are a few important variables in envar, which we have not discussed, but require to be defined correctly in order to operate DFS/SMB:

```
IOE_SMB_COMPUTER_NAME=compname
IOE_SMB_PRIMARY_WINS=ip address
IOE_SMB_SECONDARY_WINS=ip address
IOE_SMB_DOMAIN_NAME=domain name
_EUV_AUTOLOG should be set to NO
```

To disable DCE RPC and enable SMB file/print serving:

```
IOE_PROTOCOL_SMB=ON
IOE_PROTOCOL_RPC=OFF
```

A few useful OMVS (USS) commands are shown below:

- *df /directory name* – displays freespace and HFS file name under which it exists.
- *df | grep 'filesystemname'* – displays freespace for the filesystemname.
- *mv* – moves file or files from one place to another, it might be useful when restoring files from the backup.
- *chown user:group directory* – may have to be issued when mounting the filesystem for the first time. Because, after it is mounted, Unix inherits permission bits from the newly-created or mounted HFS. It may not be correct. Only the superuser can issue this command.
- *chmod 777 filename/directory* – used to set or change permission bits for filename or directory.

THE BENEFITS

In simple terms, HFS files are shared and (exported) available to be accessed from Windows workstations (or any client which supports SMB protocol). Access is acquired by mapping (net use) OS/390 HFS file to a directory in Windows (net use). So, data written into the directory goes to HFS in OS/390. Security is handled by Smbidmap (mapping file) and smbpw (the program called from the Unix shell). The Windows workstation support burden is reduced, because there is no need to install client software.

Back-up and recovery

Using mainframe HFS as the fileserver allows users to take advantage of the mainframe's robust back-up and recovery capabilities. This eliminates large quantities of PC-related back-up media and drives, and the associate administration and maintenance of these functions. The USS administrator or mainframe administrator can take over the responsibilities of the file server administrator. HFS files can be backed-up using DFDSS, HSM, or ADSM.

ADSM can be used to take back-ups at file level. DFDSS and HSM can be used to take back-up at file system level. But it will require some administration work for recovering individual files. Back-up cannot be taken at the file level using these utilities. They see the entire file as an HFS from the MVS viewpoint. File system (HFS) may be mounted or unmounted at the time of backup. DSS may give return code of 4, while taking back-up of a mounted file system. Obviously, back-up has to be taken as a *logical* backup with the *tolerate enqueue failure* keyword.

If the entire file system has to be reverted to an old version, then the file system needs to be unmounted first. Now, the file system can be deleted. When restoring the HFS from the back-up, it can be restored back to the same name and mounted onto the same mount point where it was unmounted earlier. Once this operation is complete, the DFS export command has to be given to export the file system again.

HSM can be set to take a back-up of HFS at every cycle (refer to the manual for details).

Another method is to use Unix tar or pax commands to make back-ups of Unix files. Anyone who is familiar with tar/pax commands can make file back-ups and keep the resulting tar/pax file in a directory. Then the OCOPY command (a TSO command) can be used to move the tar file from the Unix to the MVS tape file. The OCOPY command can also be used to restore the file back to USS. Using this method, individual files can be easily restored from the back-up, instead of restoring the entire filesystem. This can be considered to be file-level back-up, but it does involve some administrative work for processing back-up and restores.

The list of JCLs suporting this methodology is shown below. Batch is the recommended method (instead of doing it online by logging onto OMVS, and issuing the pax command), since there would be some control and accountability over this process. OGET or OPUT may also be used in place of OCOPY, but OCOPY allows DD to be referred which necessitates OCOPY to be used in the JCL.

- *Bpxobkp* – moves tar/pax files from USS to MVS tape/cartridge. In other words, it makes a back-up of tar files in cartridges.
- *Bpxopax* – this JCL can be used to pax files in Unix.
- *Bpxorest* – restores pax/tar files from the back-up to USS
- *Bkpfls* – adrdssu JCL to take a back-up of HFS files.

It is interesting to note that tar/pax files can be moved to MVS as PS datasets (in binary mode) and MVS PS datasets can be processed by utilities such as IEBGENER, DFDSS, etc. The following JCL is tested and runs well in OS/390 Version 2 Release 10.

Back-up archive file (created by pax or tar):

```
//BPXBKUP JOB (ACCT),'NAME',
//  NOTIFY=&SYSUID,CLASS=H,MSGCLASS=T
//IKJ1 EXEC PGM=IKJEFT01,REGION=8M
//UNIXDD1 DD PATH='/u/dirx/file1',PATHOPTS=ORDONLY
//** file1 is the pax or tar file name
//CARTDD1 DD DSN=xxxx.yyyy.file1,DISP=(,PASS),
//   UNIT=CART,LABEL=(1,SL),VOL=(,RETAIN,,6),
//   LRECL=80,BLKSIZE=8000,RECFM=FB
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 ocopy indd(unixdd1) outdd(cartdd1) binary
/*
//IKJ2 EXEC PGM=IKJEFT01,REGION=8M
//UNIXDD2 DD PATH='/u/dirx/file2',PATHOPTS=ORDONLY
//CARTDD2 DD DSN=xxxx.yyyy.file2,DISP=(,PASS),
//   UNIT=CART,LABEL=(2,SL),VOL=(,RETAIN,,6),
//   LRECL=80,BLKSIZE=8000,RECFM=FB
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 ocopy indd(unixdd2) outdd(cartdd2) binary
/*
//IKJn EXEC PGM=IKJEFT01,REGION=8M
//UNIXDDn DD PATH='/u/dirx/filen',PATHOPTS=ORDONLY
//CARTDDn DD DSN=xxxx.yyyy.filen,DISP=(,PASS),
//   UNIT=CART,LABEL=(n,SL),VOL=(,RETAIN,,6),
//   LRECL=80,BLKSIZE=8000,RECFM=FB
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 ocopy indd(unixddn) outdd(cartddn) binary
/*
//* repeat this step for each pax / tar file
```

To create a portable archive file the following step can be repeated to create ‘n’ number of archives:

```
//BPXPAX JOB (ACCT),'NAME',
//  NOTIFY=&SYSUID,CLASS=H,MSGCLASS=T
//BPXPAX1 EXEC PGM=BPXBATCH,REGION=8M,
//     PARM='SH pax -wf /u/usrid/paxpaxfl /subdir1/*'
//*      paxpaxfl - this is the output file name (archive name)
//*      subdir1  - directory to be backed up
//*      usrid    - usrid or a valid directory name
//*      STDOUT and STDERR can point to any directory for which
//*             the user has write access.
//STDOUT  DD PATH='/u/usrid/paxout.out',
//    PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
//STDERR  DD PATH='/u/usrid/paxerr.err',
//    PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU
//STDIN DD DUMMY
//SYSPRINT DD SYSOUT=*
/*
```

To restore an archive file from the back-up:

```
//BPXREST JOB (ACCT),'NAME',
//  NOTIFY=&SYSUID,CLASS=H,MSGCLASS=T
//IKJ1 EXEC PGM=IKJEFTØ1,REGION=8M
//UNIXDD1 DD PATH='/u/subdir1/paxarch',   - enter directory&file name
//    PATHOPTS=(OWRONLY,OCREAT,OTRUNC),PATHMODE=SIRWXU  - set perm bits
//CARTDD1 DD DSN=cart.dataset.name,DISP=(OLD,KEEP),
//   UNIT=CART,LABEL=(n,SL),VOL=SER=vvvsss        - enter volser and sl
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 ocopy indd(cartdd1) outdd(unixdd1) binary
/*

//BKUPFLS  JOB  (ACCTINFO),'BKUP JOB',CLASS=H,
//         MSGCLASS=X,NOTIFY=&SYSUID
//*  Submit from the image where HFS is mounted
//*  may give rc=4 when HFS is mounted (which is ok)
//STEP1   EXEC PGM=ADRDSSU,REGION=4M
//CART DD DSN=XXXX.YYYYY,TRTCH=COMP,VOL=(,,,7),
//     UNIT=CART,DISP=(NEW,KEEP),LABEL=RETPD=#NN
//SYSPRINT  DD  SYSOUT=*
//SYSIN  DD  *
 DUMP DS( -
 INCLUDE( -
 HLQ.HFS.FILE.NAME1 -
 HLQ.HFS.FILE.NAME2 -
 HLQ.HFS.FILE.NAME3 -
 HLQ.HFS.FILE.NAME4 -
 HLQ.HFS.FILE.NAMEN -
 )) -
 OUTDD(CART) TOL(ENQF) SPHERE COMP
/*
```

Sridhar Nelliyappan Manivel
Systems Programmer (USA)

z/OS Version 1 Release 2

INTRODUCTION

Version 1 Release 2 of z/OS was announced on 27 February, 2001 and is due for general release in October 2001 (see announcement letter ZP01-0164). Release 2 of z/OS will include enhancements to Kerberos, allowing users to be authenticated across multiple systems, along with additional mechanisms to help protect systems from attack. It will support both enhanced ASCII and ANSI '98 C++ standard compliance in the fourth quarter.

It will also allow basic tasks such as defining TCP/IP configuration files and a base Parallel Sysplex environment to be created more easily and, in the fourth quarter, a more robust failure recovery capability will be provided by using system-managed CF structure duplexing.

There is an emphasis on new management tools, starting with a Kerberos credential server and Kerberos application services, to provide stronger encryption, automated restart across TCP/IP network outages, and improved performance in a Parallel Sysplex environment.

With support for Kerberos third-party authentication, it provides Lightweight Directory Access Protocol (LDAP) directory client server, and the z/OS Unix System Services (USS) versions of FTP, Telnet, and RSH.

LDAP Directory service enhancements will be provided in usability, performance, and integration. An LDAP configuration utility will automate a basic setup usable by any customer, and the LDAP Server will allow for more clients to be concurrently connected. The LDAP SDBM function will enhance the capability to manage RACF-defined users and groups using LDAP.

SECURITY

Considerable emphasis has been placed on security. For example, the host-based Intrusion Detection Services (IDS) will complement network-based IDS sensors and scanners. It can discard attacking packets before they cause damage, discard packets exceeding established thresholds, and limit the number of connections from data-hungry users.

Version 1 Release 2 FTPClient and FTPServer will support SSL for ensuring confidentiality of data being transferred. Clients will be able to use digital certificates for authentication of the requestor.

Also on the security front, z/OS will be adding support for VISA, Europay, and the functions needed for ZKA certification. It will also be adding cryptographic functions needed by applications that personalize smartcards for use in PoS, debit, and stored-value applications.

The PCI Cryptographic Coprocessor supports the loading of customized cryptographic functions on zSeries 900 and S/390 Generation 5 and 6 (G5/G6) processors. With Version 1 Release 2, zSeries PCI cryptographic coprocessors, along with a special contract with IBM, will let sites define and build custom cryptographic functions themselves.

Digital certificates are addressed by the SSL function of z/OS. There is increased interoperation with certificate authority software through the incorporation of PKIX standards. Version 1 Release 2 also supports Transaction Layer Security standards and dynamic modification to System SSL configuration parameters without disrupting SSL sessions already in progress.

The TN3270 function, in conjunction with client access software, will support the use of digital certificates in place of user IDs and passwords to sign the user on to SNA applications. Host On Demand users will be able to sign on to multiple SNA applications with a single digital certificate. User passwords need not be known or defined on the target host systems.

NETWORKING

z/OS Communications Server gets Parallel Sysplex qualities of service and workload distribution functions, TCP/IP restart, and storage management enhancements. Convergence to IP networks is supported through compatibility with leading networking infrastructure providers, improved migration to dynamic routing protocols, consistent name resolution, updated DNS support (BIND9), and multiple FTP enhancements. Applications will be able to request qualities of service based on specific workload traffic.

In the fourth quarter, HiperSockets, a high-speed low-latency TCP/IP communication between logical partitions, is designed to encourage deployment of new Linux and z/OS applications on z900 servers.

Application support comes via enhanced ASCII support to port applications from ASCII platforms to Unix System Services, ANSI ’98 C++ Standard Compliance, including the Standard Template Library (STL), to port C++ applications from ASCII to USS, and functions for code set conversion between Unicode and a large set of EBCDIC and ASCII code pages.

STRATEGY IMPLICATIONS

A theme that has been consistent in recent releases of OS/390 and now z/OS is the reduction in operator skill needed for system maintenance. z/OS will allow basic tasks such as defining TCP/IP configuration files and a base Parallel Sysplex environment to be created more easily. Extended use of msys for setup for z/OS configuration, and Web-based software delivery and installation are likely to increase productivity and reduce the required skillset for managing resources

HARDWARE REQUIREMENTS

As with Release 1 of z/OS, Version 1 Release 2 will run on the z900 or comparable server, Generation 5 (G5) and Generation 6 (G6) S/390 Parallel Enterprise Servers, and all models of the Multiprise 3000 Enterprise Server. For a complete overview of z/OS Version 1 Release 2 software prerequisites, refer to the *z/OS Planning for Installation* (GA22-7504) publication. The z/OS Version 1Release 1 Product Upgrade Package (PUP) for OS/390 Version 2 Release 10 will be available until at least March 2002. Remember that the upgrade package can be used only for OS/390 Version 2 Release 10 customers migrating to z/OS Version 1 Release 1. Further information can be found at the following URL: http://www.ibm.com/servers/eserver/zseries/.

MVS news

Tivoli has announced Release 4 of NetView for OS/390, adding TCP/IP management services for OS/390 and z/OS, and extending its automation functions to distributed Unix. New TCP/IP communication services include TN3270 client, REXEC, and Remote Shell (RSH) server and client and there are also additional graphical views for monitoring TCP/IP connection status, diagnosing problems, and controlling distributed devices.

Also new is TCP/IP trace management, real-time graphing of any standard SNMP MIB-based performance data, and interactive control of devices. Special support is provided for real-time status information on the OS/390 and z/OS TCP/IP stack and TN3270 connections. The automation engine can now receive any standard message logged to a Unix system log, including those from OS/390 and z/OS Unix System Services. Unix messages can trigger a number of actions, including centralized commands or REXEC commands to start or stop services, provide operator notification, and/or log the message, adding to a trap processing capability.

For further information contact:

Tivoli Systems, 9442 Capital of Texas Highway, North Austin, TX 78759, USA.
Tel: 512 436 8000
Fax: 512 794 0623

Tivoli Systems, Sefton Park, Bells Hills, Buckinghamshire, SL2 4HD, UK.
Tel: 01753 896 896
Fax: 01753 896 899

http://www.tivoli.com

* * *

IBM's IMS Workload Router (WLR) and Dynamic Resource Control Facility (DRC) now support IMS Version 7. WLR distributes IMS transactions on predefined paths via MSC links, provides for weighted distribution of IMS transactions, provides for assignment of transactions or groups of transactions to a designated server system, supports parallel MSC sessions between the router and server systems, and provides for automatic workload reconfiguration in the event of both planned and unplanned outages.

WLR works with the IMS TM to provide routing or balancing of a transaction workload among two or more IBM systems through the Multiple Systems Coupling (MSC) facility.

The DRC on-line monitoring facility can help manage IMS for maximum performance, providing resource and helping to see potential problems. It provides displays for address spaces, dependent regions, control blocks, buffer pools, and database usage and resources.

DRC functions can be executed on a remote IMS system, so multiple IMS systems can be accessed without a direct connection to the target system.

Contact your local IBM representative for further information.

http://www.software.ibm.com

* * *