



187

MVS

April 2002

In this issue

- 3 Using REPLACE and COPY in an EVALUATE statement
 - 8 Archiving daily syslogs
 - 15 Controlling DFSMSHSM tapes
 - 25 Load modularized REXX procedure
 - 38 Maintaining a DASD configuration
 - 72 MVS news
-

© Xephon plc 2002

update

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: jaimek@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; \$505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1998 issue, are available separately to subscribers for £29.00 (\$43.50) each including postage.

MVS Update on-line

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

Editor

Jaime Kaminski

Disclaimer

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

Contributions

When Xephon is given copyright, articles published in *CICS 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 download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

© Xephon plc 2002. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs \$36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.

Using REPLACE and COPY in an EVALUATE statement

One of our programmers recently had a great idea while coding in COBOL: to use the REPLACE verb together with COPY statements (instead of COPY ... REPLACING) in an EVALUATE construct. The idea was sound, but he ran into problems when compiling the code. Also, because of the simple use of this combination, the benefits over previous methods were somewhat limited. I was asked to help solve his compilation problems and during my investigation and error reporting (to IBM) I had a few ideas myself. If more than one copybook was used and the selection of copybooks was variable, this method may bring a few benefits in code readability. In this article I describe by example what our programmer had tried to achieve, a solution to his compilation problems, and an example of how the method could be used. The original code looked something like this (changed for ease of reading):

```
*****
CASE-SELECT SECTION.
*****
EVALUATE TRUE
  WHEN CS-CASE-1
    REPLACE ==(PO)== BY ==CASE-1==.
    COPY CPYBK.
  WHEN CS-CASE-2
    REPLACE ==(PO)== BY ==CASE-2==.
    COPY CPYBK.
  WHEN CS-CASE-3
    REPLACE ==(PO)== BY ==CASE-3==.
    COPY CPYBK.
  WHEN CS-CASE-4
    REPLACE ==(PO)== BY ==CASE-4==.
    COPY CPYBK.
END-EVALUATE
EXIT.
*
```

The copybook CPYBK looked something like this:

```
*
*   COPYBOOK FOR REPLACE DEMONSTRATION
*
MOVE CTL-RC-RECORD-1
```

```

      TO (PO)-RECORD-1
MOVE CTL-RC-RECORD-2
      TO (PO)-RECORD-2
MOVE CTL-RC-RECORD-3
      TO (PO)-RECORD-3
MOVE CTL-RC-RECORD-4
      TO (PO)-RECORD-4
MOVE CTL-RC-RECORD-5
      TO (PO)-RECORD-5
MOVE CTL-RC-RECORD-6
      TO (PO)-RECORD-6
MOVE CTL-RC-RECORD-7
      TO (PO)-RECORD-7
MOVE CTL-RC-RECORD-8
      TO (PO)-RECORD-8
MOVE CTL-RC-RECORD-9
      TO (PO)-RECORD-9

```

What our programmer hoped to generate from his compilation should have looked something like this:

```

*****
CASE-SELECT SECTION.
*****
      EVALUATE TRUE
      WHEN CS-CASE-
          REPLACE ==(PO)== BY ==CASE-1==.
          COPY CPYBK.
*
*   COPYBOOK FOR REPLACE DEMONSTRATION
*
      MOVE CTL-RC-RECORD-
          TO CASE-1-RECORD-1
      MOVE CTL-RC-RECORD-2
          TO CASE-1-RECORD-2
      MOVE CTL-RC-RECORD-3
          TO CASE-1-RECORD-3
      MOVE CTL-RC-RECORD-4
          TO CASE-1-RECORD-4
      MOVE CTL-RC-RECORD-5
          TO CASE-1-RECORD-5
      MOVE CTL-RC-RECORD-6
          TO CASE-1-RECORD-6
      MOVE CTL-RC-RECORD-7
          TO CASE-1-RECORD-7
      MOVE CTL-RC-RECORD-8
          TO CASE-1-RECORD-8
      MOVE CTL-RC-RECORD-9
          TO CASE-1-RECORD-9
      WHEN CS-CASE-2

```

REPLACE ==(PO)== BY ==CASE-2==.
COPY CPYBK.

*
*
*

COPYBOOK FOR REPLACE DEMONSTRATION

MOVE CTL-RC-RECORD-
TO CASE-2-RECORD-1
MOVE CTL-RC-RECORD-2
TO CASE-2-RECORD-2
MOVE CTL-RC-RECORD-3
TO CASE-2-RECORD-3
MOVE CTL-RC-RECORD-4
TO CASE-2-RECORD-4
MOVE CTL-RC-RECORD-5
TO CASE-2-RECORD-5
MOVE CTL-RC-RECORD-6
TO CASE-2-RECORD-6
MOVE CTL-RC-RECORD-7
TO CASE-2-RECORD-7
MOVE CTL-RC-RECORD-8
TO CASE-2-RECORD-8
MOVE CTL-RC-RECORD-9
TO CASE-2-RECORD-9
WHEN CS-CASE-3
REPLACE ==(PO)== BY ==CASE-3==.
COPY CPYBK.

*
*
*

COPYBOOK FOR REPLACE DEMONSTRATION

MOVE CTL-RC-RECORD-
TO CASE-3-RECORD-1
MOVE CTL-RC-RECORD-2
TO CASE-3-RECORD-2
MOVE CTL-RC-RECORD-3
TO CASE-3-RECORD-3
MOVE CTL-RC-RECORD-4
TO CASE-3-RECORD-4
MOVE CTL-RC-RECORD-5
TO CASE-3-RECORD-5
MOVE CTL-RC-RECORD-6
TO CASE-3-RECORD-6
MOVE CTL-RC-RECORD-7
TO CASE-3-RECORD-7
MOVE CTL-RC-RECORD-8
TO CASE-3-RECORD-8
MOVE CTL-RC-RECORD-9
TO CASE-3-RECORD-9

END-EVALUATE
EXIT.

What he got, however, was:

```
000098          EVALUATE TRUE
000099          WHEN CS-CASE-1
000100          REPLACE ==(PO)== BY ==CASE-1==.
000101          COPY CPYBK.
000102C          *
000103C          * COPYBOOK FOR REPLACE DEMONSTRATION
000104C          *
000105C          *
000106C          1          MOVE CTL-RC-RECORD-1
000107C          1          TO CASE-1-RECORD-1
000108C          *
000109C          1          MOVE CTL-RC-RECORD-2
000110C          1          TO CASE-1-RECORD-2
000111C          *
000112C          1          MOVE CTL-RC-RECORD-3
000113C          1          TO CASE-1-RECORD-3
000114C          *
```

1PP 5648-A25 IBM COBOL for OS/390 & VM 2.2.0
COBOLPRG Date 02/20/2002 Time 16:16:19 Page 5

LineID PL SL —+—*A-1-B—+—2—+—3—+—4—+—5—+—6—+—7—|—+—8

Map and Cross Reference

```
0 000115C          1          MOVE CTL-RC-RECORD-4
000116C          1          TO CASE-1-RECORD-4
000117C          *
000118C          1          MOVE CTL-RC-RECORD-5
000119C          1          TO CASE-1-RECORD-5
000120C          *
000121C          1          MOVE CTL-RC-RECORD-6
000122C          1          TO CASE-1-RECORD-6
000123C          *
000124C          1          MOVE CTL-RC-RECORD-7
000125C          1          TO CASE-1-RECORD-7
000126C          *
000127C          1          MOVE CTL-RC-RECORD-8
000128C          1          TO CASE-1-RECORD-8
000129C          *
000130C          1          MOVE CTL-RC-RECORD-9
000131C          1          TO CASE-1
```

==000131==> IGYPS2121-S "CASE-1" was not defined as a data-name. The statement was discarded.

```
000132C          *
000133C          *
000134          1          -RECORD-9
```

==000134==> IGYPS0001-W A blank was missing before character "R" in column 13. A blank was assumed.

==000134==> IGYPS2072-S "-" was invalid. Skipped to the next verb, period or procedure-name definition.

```
000135      1          WHEN CS-CASE-2
000136          REPLACE ==(PO)== BY ==CASE-2==.
000137          COPY CPYBK.
```

After searching for possible solutions, and also contacting IBM, I found that a quick solution was to place a CONTINUE statement as the last statement in the copybook. After that the compilation ran successfully.

After providing a temporary fix for this problem, I had a few thoughts about the implementation.

The use of the REPLACE statement in this context is little more than the combined COPY and REPLACING statements, and this could be used just as easily. The big difference between the two methods is that the COPY ... REPLACING is just for one specific copybook, whereas the REPLACE statement changes all the following code up to the next REPLACE statement, up to REPLACE OFF, or until the end of the program.

So how could this method be of more use? The following scenario shows how useful this methodology can be.

If more than one copybook was being used and, say, depending on the business context, not every copybook was needed for every case, an interesting coding example would be something like this:

```
*
*****
CASE-SELECT SECTION.
*****
    EVALUATE TRUE
      WHEN CS-CASE-1
        REPLACE ==(PO)== BY ==CASE-1==.
        COPY CPYBKA.
        COPY CPYBKB.
        COPY CPYBKC.
      WHEN CS-CASE-2
        REPLACE ==(PO)== BY ==CASE-2==.
        COPY CPYBKA.
      WHEN CS-CASE-3
        REPLACE ==(PO)== BY ==CASE-3==.
        COPY CPYBKA.
        COPY CPYBKC.
```

```

        COPY CPYBKD.
    WHEN CS-CASE-4
        REPLACE ==(PO)== BY ==CASE-4==.
        COPY CPYBKA.
        COPY CPYBKB.
    END-EVALUATE
    EXIT.

```

*

The documentation can be found in IBM's *COBOL Language Reference* (SC26-9046-04) section 8.1.10 *REPLACE statement*.

However, there are a few points to note:

- The **REPLACE** statement must be preceded by a period (full stop) and terminated by a period.
- The **REPLACE** statement replaces all occurrences to the end of the code or until the next **REPLACE** statement.
- The copybooks, when used together with the **EVALUATE** statement, cannot contain full stops.
- **REPLACE** statements can themselves contain **COPY** statements.

At the time of writing the reported compiler problem (IBM) has a status of 'open'; until the problem is resolved the **CONTINUE** phrase must be used as the last statement in the copybook.

Rolf Parker
Systems Programmer (Germany)

© Xephon 2002

Archiving daily syslogs

The following REXX program and JCL can be used to control the archiving of daily syslogs on OS/390. The MPF exit 'SLOGMPF1' is used to issue the commands to:

- Start STC 'ARCSLG1', which uses the external writer to write the syslog output to an archive dataset.
- Issue the correct command to stop the writer started above when it has finished writing.

The exit looks for class 'L' output, which is our installation standard for sysout. Before running this for the first time you would need to manually allocate SYSLOG.DAILY.ARCHIVE. See 'ELOG2' for the attributes.

The system also requires time-initiated commands to be issued for the following (these can be issued using JES Time Initiated Commands, or perhaps an in-house written utility or ISV offering):

- Issue the 'WRITELOG' command to spool off the syslog output, at the required intervals (the example here just uses one minute past midnight, but you may want to make this a more frequent interval).
- Start the 'ARCSLG2' STC, which renames the archived syslog dataset and reallocates another, with a header in it.
- Start job 'LOGMAINT', which uses DFDSS to free off unused space in the syslog archive datasets and deletes the oldest ones (as required).

So, the sequence of events would be:

```

00:01:00  W L                               Time-initiated (WRITELOG)
00:02:00  IEE043I A SYSTEM LOG DATASET HAS BEEN QUEUED TO SYSOUT
          CLASS L
00:02:01  S ARCSLG1                         Issued by exit for above msg
00:03:00  IEF176I WTR 1234 WAITING FOR WORK, CLASS=L, DEST=LOCAL
00:03:00  P ARCSLG1.1234                   Issued by exit for above msg
00:10:00  S ARCSLG2                         Time initiated (issued after
          a reasonable wait for IEF176I)
00:30:00  Submit LOGMAINT                  Time initiated

```

SLOGMPF1 ASM

```

*****
*      MODULE      =  SLOGMPF1                *
*      DESCRIPTION =  COMMUNICATION TASK USER EXIT FOR MESSAGES: *
*                                                         *
*      IEE043I A SYSTEM LOG DATASET HAS BEEN QUEUED TO SYSOUT *
*      AND  IEF176I WTR CCUU WAITING FOR WORK, CLASS=L, DEST=LOCAL *
*                                                         *
*      THIS EXIT WILL ISSUE THE APPROPRIATE COMMANDS IN RESPONSE *
*      TO THESE MSGS SO THAT SYSLOG ARCHIVING CAN BE PERFORMED. *
*                                                         *
*      ENTRY POINT =  SLOGMPF1                *
*****

```

```

SLOGMPF1 CSECT
SLOGMPF1 AMODE 31
SLOGMPF1 RMODE ANY
*-----*
* HOUSEKEEPING... *
*-----*
        BAKR  R14,Ø          SAVE CALLER DATA ON STACK
        LR   R12,R15         GET ENTRY POINT
        USING SLOGMPF1,R12   MODULE ADDRESSABILITY
        L    R7,Ø(R1)        CTXT LOAD ADDRESS
        USING CTXT,R7        ESTABLISH ADDRESSABILITY TO CTXT
        L    R6,CTXTTTPJ     LOAD POINTER TO TEXT
        USING CTXTATTR,R6    ADDRESSABILITY TO TEXT
*-----*
* GET DYNAMIC WORKAREA... *
*-----*
GETSTOR  DS    ØH
        GETMAIN RU,LV=CMDLEN,SP=23Ø
        LR   R11,R1          ADDRESS RETURNED IN R1
        USING DATAAREA,R11  ADDRESSABILITY TO DYNAMIC AREA
        MVC  CMDPARMS(CMDLEN),CMDAREA SET UP DYNAMIC AREA
*-----*
* DETERMINE MESSAGE TYPE... *
*-----*
CHKMSG   DS    ØH
        CLC  CTXTTMSG(8),IEEØ43I  IEEØ43I MSG?
        BE   ISSUCMD1             YES..ISSUE 'S BSYSLOG'
        CLC  CTXTTMSG(8),IEF176I  IEF176I MSG?
        BE   CHEKCLAS             YES..CHECK ITS CLASS=L
        B    GOBACK               JUST IGNORE
CHEKCLAS DS    ØH
        CLC  CTXTTMSG+35(7),=C'CLASS=L' SYSLOG CLASS?
        BNE  GOBACK               NO...FORGET IT
        MVC  CMDTEXT,CMD2
        MVC  CMDTEXT+1Ø(4),CTXTTMSG+12  MOVE ADDRESS TO CMD
        B    ISSUECMD
ISSUCMD1 DS    ØH
        MVC  CMDTEXT,CMD1          SET UP COMMAND
ISSUECMD DS    ØH
        XR   RØ,RØ                CLEAR REGISTER
        LA  R1,CMDPARMS           POINT TO COMMAND
        SVC  34                   ISSUE COMMAND
*-----*
* RETURN... *
*-----*
GOBACK   DS    ØH
        FREEMAIN RU,LV=CMDLEN,A=(R11),SP=23Ø  FREE DYNAMIC AREA
        XR   R15,R15             ALWAYS RC=Ø
        PR   ,                   RETURN TO CALLER
        TITLE 'SLOGMPF1- DATA AREAS, ETC...'
*-----*

```

```

* CONSTANTS... *
*-----*
IEE043I DC CL8'IEE043I '
IEF176I DC CL8'IEF176I '
CMD1 DC CL14'S ARCSLG1 '
CMD2 DC CL14'P ARCSLG1.CCUU'
CMDAREA DS 0F SVC34 AREA
PARMLEN DC Y(CMDLEN) LENGTH INCLUDING HEADER
DC H'0' ZERO
DC CL14' '
DC CL26' (ISSUED BY SLOGMPF1)'
CMDLEN EQU *-CMDAREA
LTORG
*-----*
* DYNAMIC WORKAREA... *
*-----*
DATAAREA DSECT DYNAMIC STORAGE AREA
CMDPARMS DS 0F DYNAMIC SVC34 AREA
DS H LENGTH INCLUDING HEADER
DS H ZERO
CMDTEXT DS CL14 COMMAND TO BE ISSUED
DC CL26' ' 'ISSUED BY...' COMMENT
DATALEN EQU *-DATAAREA
*-----*
* REGISTER EQUATES... *
*-----*
YREGS
*-----*
* DSECTS... *
*-----*
IEZVX100
END SLOGMPF1

```

ARCSLG1 STC

```

//ARCSLG1 PROC
//* -----
//* Started by SLOGMPF1 in response to msg IEE043I, issued after
//* a 'W L' command. It takes the Syslog from Class L and mods it
//* onto dataset 'SYSLOG.DAILY.SAVELOG'. These commands can be
//* issued throughout the day.
//* -----
//IEFPROC EXEC PGM=IASXWR00,
// PARM='PL'
//IEFRDR DD DSN=SYSLOG.DAILY.ARCHIVE,DISP=MOD

```

ARCSLG2 STC

```

//ARCSLG2 PROC
/** -----

```

```

/* Renames the accumulated daily log SYSLOG.DAILY.ARCHIVE to
/* SYSLOG.ARCHIVE.Yyyyy.Mmm.Ddd
/** -----
//NAMELOG EXEC PGM=IKJEFT01,COND=(0,NE),
//          DYNAMNBR=50,PARM='%ELOG1'
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//SYSPROC  DD  DISP=SHR,DSN=SYS3.ELIB
/** -----
/* Reallocate and prime the daily log.
/** -----
//RESETLOG EXEC PGM=IKJEFT01,COND=(0,NE),
//          DYNAMNBR=50,PARM='%ELOG2'
//SYSPRINT DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  DUMMY
//SYSPROC  DD  DISP=SHR,DSN=SYS3.ELIB

```

ELOG1 REXX

```

/**REXX*****
/* ELOG1: Invoked by procedure ARCSLG2
/*****
Trace n
Address 'TS0'
/*****
/* Read date from current SYSLOG file
/*****
logname = "SYSLOG.DAILY.ARCHIVE"
"ATTR LOG DSORG(PS) RECFM(V B A)"
"ALLOC FI(SYSLOG) DA('"logname"') USING(LOG) SHR"
"EXECIO" 1 "Diskr SYSLOG (Stem logdata. FINIS )"
logdd = Substr(logdata.1,39,2)
logmm = Substr(logdata.1,42,2)
logyyy = Substr(logdata.1,45,4)
fdate = "Y"||logyyy||".M"||logmm||".D"||logdd
"FREE FI(SYSLOG)"
/*****
/* Rename syslog ready for archive
/*****
newname = "SYSLOG.ARCHIVE."||fdate
Rename logname newname
If rc > 0 Then
    "SEND 'ERROR OCCURRED IN ARCSLG2(ELOG1) - RENAME FAILED'",
    "USER(uuuuuuu uuuuuuu uuuuuuu) LOGON NOWAIT "

```

ELOG2 REXX

```

/**REXX*****

```



```

RELEASE DDNAME(DASD1) INCLUDE(SYSLOG.ARCHIVE.***)
RELEASE DDNAME(DASD2) INCLUDE(SYSLOG.ARCHIVE.***)
etc....
/**
/** DELETE ANY SYSLOG DATASETS OLDER THAN SPECIFIED DAYS...
/**
//S02      EXEC PGM=IKJEFT01,DYNAMNBR=50,
//          PARM='%ESLOG00'
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSEXEC  DD DSN=SYS3.ELIB,DISP=SHR
//SYSTSIN  DD DUMMY
//

```

ESLOG00 REXX

```

/**REXX*****
/* ESLOG00: Invoked by job LOGMAINT */
/*****
/* Delete SYSLOGs older than nn days (set in 'arcdays'). */
/*****
x = outtrap("OUT.",200,"NOCONCAT")
arcdays = 14
names. = ""
nnam = 0
"LISTC LVL('SYSLOG.ARCHIVE')"
x = OUTTRAP("OFF")

Do a = 1 to out.0
  If Left(out.a,8) = "NONVSAM " Then Do
    Parse Upper VAR out.a . . dsn
    nnam = nnam + 1
    names.nnam = dsn
  End
End
If nnam < arcdays Then Do
  Say "Nothing to delete, ending..."
  Exit
End
/* As the datasets are in date order, we'll just delete the oldest */
/* ones, first calculating how many we need to get rid of... */
Say "More than "arcdays" datasets found, deleting the oldest..."
xx = nnam - arcdays14
Address "TS0"
Do a = 1 to xx
  "DELETE '"names.a'" "
End
Return

```

Grant Carson
Systems Programmer (UK)

© Xephon 2002

Controlling DFSMSHSM tapes

THE PROBLEM

We have about one terabyte of disk space under SMS and consequently we use about a hundred Magstar tapes for migration and back-up of primary space. Datasets migrate to tapes according to defined management classes. As a result, a huge number of datasets become obsolete at ML2 or at back-up time each day. We recycle them regularly when the percentage of valid data decreases to below 25%.

We noticed that from time to time there are different problems with HSM tapes. Our analysis suggested the following problems:

- Damaged tapes that become inaccessible because of label errors or data checks.
- Software problems that cause malfunctioning of optimal tape utilization, reduction of spanning, etc.

The tape problems result in a discrepancy between information in DFHSM control datasets and the tape contents. That is why some of the tapes are not eligible for recycling although they have a small percentage of valid data; and even more of a problem is that some of the data is unusable.

A SOLUTION

These are the reasons for writing the HSMCHK procedure that periodically checks tape information and generates statements for repair. The procedure analyses tapes starting from the TTOC (tape table of contents) and AUDIT VOLUMECONTROLS output. The user specifies the percentage of valid data that is to be the criterion for recycling. This parameter necessitates the utilization of a whole tape set for connected tapes.

The procedure follows the standard IBM methodology for repairing control data. The AUDIT statement with the FIX option does the automatic corrections, but many types of error remain after AUDIT. Based on the DFHSM literature and our experience we found ways of correcting the remaining errors.

Statements for different types of action are placed in the four datasets. Users can check them, uncomment some of the proposed statements, and execute them at the appropriate time.

Dataset `userid.HSM.#ACTION.AUDIT.LIST` contains statements for additional AUDITs that are necessary in case of error ARC0378I. If `AUDITMEDIACONTROL` finishes successfully, control information is updated and the error does not exist any more. If the tape is damaged, `AUDITMEDIACONTROL` cannot succeed and we recommend using `RECYCLE FORCE` in the second pass. If `RECYCLE FORCE` did not finish successfully the data on tape is lost. You have to execute the `FIXCDS` statement that is generated in the fourth dataset.

Dataset `userid.HSM.#ACTION.RECYCLE.LIST` contains statements about three subtypes of action:

- Regular recycling.
- Recycling of damaged tapes that have ERROR 58 in audit output.
- Optional recycling of partially-used tapes with a percentage of valid data lower than the specified percentage.

These tapes can be:

- Current ML2 or back-up tapes that HSM will use in the next session. This situation occurs regularly and doesn't need any action.
- Tapes that HSM used at some time in the past and left partially used. It didn't mark them as full because of a software or operator problem.

Unfortunately, there is no date stamp in HSM reports, so we cannot distinguish regular from irregular tapes. That is why we generate statements for marking them full and recycling, but put them inside comments.

Dataset `userid.HSM.#DELVOL.AUDIT.LIST` contains statements for deleting volumes that contain 0% of valid data or have status empty.

Dataset `userid.HSM.#FIXCDS.AUDIT.LIST` contains statements to execute at the end of the process. We recommend `FIXCDS` only for tapes that are totally damaged.

HSMCHK

```
/****** REXX *****/
/* Procedure checks utilization of back-up and migrate tapes and */
/* generates statements for recycling and repairing tapes that */
/* have a percent of valid data smaller then specified the percent */
/* */
/* Call: */
/* %HSMCHK percent */
/* */
/* percent - Percent of valid data that is the criteria for recycling*/
/* */
/*******/
/* Trace ?r */
```

ARG Percent

```
userid=SYSVAR(SYSUID)
prefix=SYSVAR(SYSPREF)
"PROFILE NOPREFIX"
signal on error
```

```
/*- Names of work datasets -----*/
DSN_HSM_TTOC =userid||'.HSM.#TTOC.LIST'
DSN_HSM_AUDIT =userid||'.HSM.#AUDIT.LIST'
DSN_HSM_ACT_REC =userid||'.HSM.#ACTION.RECYCLE.LIST'
DSN_HSM_ACT_DEL =userid||'.HSM.#ACTION.DELVOL.LIST'
DSN_HSM_ACT_AUD =userid||'.HSM.#ACTION.AUDIT.LIST'
DSN_HSM_ACT_FIX =userid||'.HSM.#ACTION.FIXCDS.LIST'

/* ----- */
/* Main Procedure */
/* ----- */

/*- Delete Work Files -----*/
msgstat=MSG("OFF") /* Inhibit the display of TSO/E information */
/* messages */

CALL Delete_DS DSN_HSM_TTOC
CALL Delete_DS DSN_HSM_AUDIT
CALL Delete_DS DSN_HSM_ACT_REC
CALL Delete_DS DSN_HSM_ACT_DEL
CALL Delete_DS DSN_HSM_ACT_AUD
CALL Delete_DS DSN_HSM_ACT_FIX

t=MSG(msgstat) /* Returns the previous status of message */.

/*- Free all empty HSM tapes -----*/
"HSEND CMD WAIT RECYCLE ALL PERCENTVALID(0) EXECUTE"

/*- Audit Volume Controls -----*/
```

```

"HSEND CMD WAIT AUDIT VOLUMECONTROLS(BACKUP) FIX",
  " ODS("DSN_HSM_AUDIT")"
"HSEND CMD WAIT AUDIT VOLUMECONTROLS(MIGRATION) FIX",
  " ODS("DSN_HSM_AUDIT")"

/*- List TTOC of HSM Tapes -----*/
"HSEND CMD WAIT LIST TTOC SELECT(NOTCONNECTED) ODS("DSN_HSM_TTOC")"
"HSEND CMD WAIT LIST TTOC SELECT(CONNECTED) ODS("DSN_HSM_TTOC")"

/*- Check TTOC information and generate repair actions -----*/
"ALLOC F(HSMTTOC) DA("''''DSN_HSM_TTOC''''") OLD"
  Call Get_HSM_TTOC_Info Percent
"FREE F(HSMTTOC)"

  Call Gen_HSM_Actions

/*- Check AUDIT informations and generate repair actions -----*/
"ALLOC F(HSMAUDIT) DA("''''DSN_HSM_AUDIT''''") OLD"
  Call Get_HSM_AUDIT_Info
"FREE F(HSMAUDIT)"

  Call Gen_HSM_Actions2

/*- Write action datasets -----*/
  If HsmActR.0 > 0
  Then Do
    Call Alloc_DS HSMACTR DSN_HSM_ACT_REC NEW 80 F
    "EXECIO 0 DISKW HSMACTR (OPEN) "
    "EXECIO * DISKW HSMACTR (STEM HsmActR. FINIS) "
    Say 'File with DFSMSHSM RECYCLE actions is created!'
  End

  If HsmActD.0 > 0
  Then Do
    Call Alloc_DS HSMACTD DSN_HSM_ACT_DEL NEW 80 F
    "EXECIO 0 DISKW HSMACTD (OPEN) "
    "EXECIO * DISKW HSMACTD (STEM HsmActD. FINIS) "
    Say 'File with DFSMSHSM DELVOL actions is created!'
  End

  If HsmActA.0 > 0
  Then Do
    Call Alloc_DS HSMACTA DSN_HSM_ACT_AUD NEW 80 F
    "EXECIO 0 DISKW HSMACTA (OPEN) "
    "EXECIO * DISKW HSMACTA (STEM HsmActA. FINIS) "

    Say 'File with DFSMSHSM AUDIT actions is created!'
  End

```

```

If HsmActF.Ø > Ø
Then Do
    Call Alloc_DS HSMACTF DSN_HSM_ACT_FIX NEW 8Ø F
    "EXECIO Ø DISKW HSMACTF (OPEN) "
    "EXECIO * DISKW HSMACTF (STEM HsmActF. FINIS) "

    Say 'File with DFSMSHSM FIXCDS actions is created!'
End

If HsmActR.Ø = Ø & HsmActD.Ø = Ø & HsmAct.Ø = Ø & HsmActF.Ø = Ø
Then Say 'No tape with valid percent less than 'Percent

error:
If prefix <> ''
Then "PROFILE PREFIX("prefix")"

Return

/* ----- */
/* Get HSM TTOC Information */
/* ----- */
Get_HSM_TTOC_Info: Procedure Expose HsmTape. Type. PctValid.
VolStatus.,
                VolInd. DSN_HSM_AUDIT
ARG Pct

HsmAllTtocInfo.Ø = Ø
" EXECIO * DISKR hsmttoc (STEM HsmAllTtocInfo. FINIS) "

k = Ø
NoConn = Ø
PctConnSet = Ø
Do i=1 to HsmAllTtocInfo.Ø
    FirstWord = WORD(HsmAllTtocInfo.i,1)
    SecondWord = WORD(HsmAllTtocInfo.i,2)

    Select
    When SecondWord = '348Ø' |,
        SecondWord = '348ØX' |,
        SecondWord = '359Ø-1'
    Then Do
        k = k + 1
        HsmTape.k = WORD(HsmAllTtocInfo.i,1)
        Type.k = WORD(HsmAllTtocInfo.i,3)
        PctValid.k = WORD(HsmAllTtocInfo.i,6)
        VolStatus.k = WORD(HsmAllTtocInfo.i,7)
        If Type.k = 'ML2'
        Then Type.k = 'MIGRATION'
        Else Type.k = 'BACKUP'
        If PctValid.k <= Pct
        Then VolInd.k = ' '

```

```

Else VolInd.k = 'N'

/* For connected tape */
PrevVol = WORD(HsmAllTtocInfo.i,9)
PrevVol = SUBSTR(PrevVol,2,4)
SuccVol = WORD(HsmAllTtocInfo.i,10)
SuccVol = SUBSTR(SuccVol,2,4)
If PrevVol = 'NONE' | SuccVol = 'NONE'
Then Do
    NoConn = NoConn + 1
    PctConnSet = PctConnSet + PctValid.k
End

End
When FirstWord = 'VOLSER'
Then Do
    NoConn = 0
    PctConnSet = 0
End
When FirstWord = '***END'
Then Do
    PctConnSet = PctConnSet / NoConn
    If (PctConnSet <= Pct)
    Then Do j = k - NoConn To k - 1
        VolInd.j = 'C'
    End

End
When SecondWord = 'ARC0378I'
Then Do
    VolInd.k = 'E'
End
When SecondWord = 'ERROR'
Then Do
    If WORD(HsmAllTtocInfo.i,1) = 'ARC0184I'
    Then Do
        Volume = WORD(HsmAllTtocInfo.i,13)
        Volume = LEFT(Volume,6)
        say HsmAllTtocInfo.i
        say ' HSEND AUDIT DIRECTORYCONTROLS VOLUMES('Volume')',
            ' FIX ODS('DSN_HSM_AUDIT')'
        "HSEND CMD WAIT AUDIT DIRECTORYCONTROLS VOLUMES("Volume,
        ") FIX ODS("DSN_HSM_AUDIT")"
    End

End
Otherwise
End /* Select */
End
HsmTape.0 = k
Type.0 = k
PctValid.0 = k
VolStatus.0 = k

```

```

VolInd.Ø = k
Drop HsmAllTtocInfo.
Return

/*-----*/
/* Gen HSM Actions I */
/*-----*/
Gen_HSM_Actions: Procedure Expose HsmTape. Type. PctValid. VolStatus.,
                          VolInd. HsmActR. HsmActD. HsmActA. HsmActF.

r = Ø
d = Ø
a = Ø
f = Ø
Do i=1 to HsmTape.Ø
  If VolInd.i ^= 'E' & VolInd.i ^= 'N'
  Then Do
    Select
      When VolStatus.i = 'EMPTY' | PctValid.i = Ø
      Then Do
        d = d + 1
        HsmActD.d=' /*- VOL('HsmTape.i') Status('VolStatus.i,
                    ') PctValid('PctValid.i') Type('Type.i') -*/'
        d = d + 1
        HsmActD.d=' HSEND DELVOL 'HsmTape.i Type.i'(MARKFULL)'
        d = d + 1
        HsmActD.d=' HSEND DELVOL 'HsmTape.i Type.i'(PURGE)'
        d = d + 1
        HsmActD.d=' '
      End
      When VolStatus.i = 'FULL'
      Then Do
        r = r + 1
        HsmActR.r=' /*- VOL('HsmTape.i') Status('VolStatus.i,
                    ') PctValid('PctValid.i') Type('Type.i') -*/'
        r = r + 1
        HsmActR.r=' HSEND RECYCLE VOLUME('HsmTape.i') EXECUTE '
        r = r + 1
        HsmActR.r=' '
      End
      Otherwise
      Do
        r = r + 1
        HsmActR.r=' /*- VOL('HsmTape.i') Status('VolStatus.i,
                    ') PctValid('PctValid.i') Type('Type.i') -*/'
        r = r + 1
        HsmActR.r=' /*# HSEND DELVOL 'HsmTape.i Type.i,
                    '(MARKFUL) #*/'
        r = r + 1
        HsmActR.r=' /*# HSEND RECYCLE VOLUME('HsmTape.i')',
                    ' EXECUTE #*/ '
      End
    End
  End
End

```

```

        r = r + 1
        HsmActR.r=' '
        End
    End /* Select */
End

If VolInd.i = 'E'
Then Do
    a = a + 1
    HsmActA.a=' /*-Tape in ERROR VOL('HsmTape.i') Status(',
    VolStatus.i') PctValid('PctValid.i') Type('Type.i')-*/'
    a = a + 1
    HsmActA.a='          HSEND AUDIT MEDCTL VOLUMES('HsmTape.i') FIX '
    a = a + 1
    HsmActA.a=' /*# HSEND RECYCLE VOLUME('HsmTape.i,
                ') EXECUTE FORCE #*/'
    a = a + 1
    HsmActA.a=' '

    f = f + 1
    HsmActF.f=' /*-Tape in ERROR VOL('HsmTape.i') Status(',
    VolStatus.i') PctValid('PctValid.i') Type('Type.i')-*/'

    If Type.i = 'MIGRATION'
    Then TypeF = 'Y'
    Else TypeF = 'B'
    f = f + 1
    HsmActF.f=' /*@ HSEND FIXCDS 'TypeF HsmTape.i' DELETE @*/'
    f = f + 1
    HsmActF.f=' '
    End
End

HsmActR.Ø = r
HsmActD.Ø = d
HsmActA.Ø = a
HsmActF.Ø = f
Return

/*-----*/
/* Get HSM AUDIT Information */
/*-----*/
Get_HSM_AUDIT_Info: Procedure Expose HsmErrTape. ErrNo.

HsmAllAuditInfo.Ø = Ø
" EXECIO * DISKR hsmaudit (STEM HsmAllAuditInfo. FINIS) "

k = Ø
Volumes = ''
Do i=1 to HsmAllAuditInfo.Ø
    ErrWord = WORD(HsmAllAuditInfo.i,2)

```

```

If ErrWord = 'ERR'
Then Do
  Volume = WORD(HsmAllAuditInfo.i,4)
  If Volumes  $\neq$  Volume
  Then Do
    k = k + 1
    HsmErrTape.k = Volume
    ErrNo.k = WORD(HsmAllAuditInfo.i,3)
    Volumes = Volume
  End
End
End
HsmErrTape.0 = k
ErrNo.0 = k
Drop HsmAllAuditInfo.
Return

/* ----- */
/* Gen HSM Actions II */
/* ----- */
Gen_HSM_Actions2: Procedure Expose HsmErrTape. ErrNo. HsmActR.,
                        DSN_HSM_AUDIT
r = HsmActR.0
Do i=1 to HsmErrTape.0
  say ' HSEND AUDIT DIRECTORYCONTROLS VOLUMES('HsmErrTape.i')',
    ' FIX ODS('DSN_HSM_AUDIT')'
  "HSEND CMD AUDIT DIRECTORYCONTROLS VOLUMES("HsmErrTape.i")",
  " FIX ODS("DSN_HSM_AUDIT")"

  If ErrNo.i = 58
  Then Do
    r = r + 1
    HsmActR.r=' /*- Tape in ERROR ('ErrNo.i') 'HsmErrTape.i')',
              ' -*/'

    r = r + 1
    HsmActR.r=' HSEND RECYCLE VOLUME('HsmErrTape.i') EXECUTE FORCE'
    r = r + 1
    HsmActR.r=' '
  End
End
HsmActR.0 = r
Return

/* ----- */
/* Allocate Dataset */
/* ----- */
Alloc_DS: Procedure
Arg DD_name DS_name Disp Length_rec Rec_fm

If Disp='NEW' & SYSDSN(''Ds_name'') = 'OK'

```

```

Then Disp='SHR'

If Disp = 'NEW'
Then "ALLOC F("DD_name") DA("''''DS_name''''") "Disp" CATALOG",
    " SPACE(50,50) LRECL("Length_rec") RECFM("Rec_fm" B)",
    " BLKSIZE(0) RELEASE"
Else "ALLOC F("DD_name") DA("''''DS_name''''") "Disp" REUSE"
Return

/* ----- */
/* Delete Dataset */
/* ----- */
Delete_DS: Procedure
Arg DS_name
If SYSDSN('''DS_name''') = 'OK'
THEN "DELETE" ''DS_name'' "PURGE"
Return

```

HSMCHK PROCEEDURE CALL

```

//useridD   JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(0,0),NOTIFY=&SYSUID
//HSMCHK1   EXEC PGM=IKJEFT01,DYNAMNBR=50,
//          REGION=4M
//SYSPROC   DD DSN=userid.CLIST,DISP=SHR
//SYSTSPRT  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSTSIN   DD *
            %HSMCHK  10
/*
//

```

JOB FOR HSM ACTIONS

```

//useridD   JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(0,0),NOTIFY=&SYSUID
//HSMACT    EXEC PGM=IKJEFT01,DYNAMNBR=50,
//          REGION=4M
//SYSTSPRT  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSTSIN   DD DSN=userid.HSM.#ACTION.DELVOL.LIST,DISP=SHR
//          DD DSN=userid.HSM.#ACTION.RECYCLE.LIST,DISP=SHR
//*         DD DSN=userid.HSM.#ACTION.AUDIT.LIST,DISP=SHR
//*         DD DSN=userid.HSM.#ACTION.FIXCDS.LIST,DISP=SHR
//

```

Emina Spasic and Dragan Nikolic
Systems Programmers
Postal Savings Bank (Yugoslavia)

© Xephon 2002

Load modularized REXX procedure

THE PROBLEM

Although the REXX language provides basic support for modularized programs (for example calls to external routines), the execution environment has several deficiencies. For example:

- There is a massive overhead for each call to a routine contained in an external library; I have measured elapsed times in the order of one second for each and every call to an external routine.
- Although this time-overhead problem can be solved by combining all routines in a single physical module, such a binding is static. This means that the calling module needs to be updated every time a called routine is changed.
- If such a module provides general routines, this module must be physically included in every procedure that uses this functionality. This in turn makes it more difficult to make changes.

A SOLUTION

REXXPREP acts as a preprocessor similar to that known from the C language. REXXPREP reads the primary procedure into main storage. It loads the associated source procedures from the library for any include (`#INCLUDE`) instructions it encounters in the primary procedure. It then invokes the REXX interpreter using this expanded in-storage procedure. This means that the resulting procedure is, in effect, a single module and so runs without the time overhead involved with the invocation of external routines. Because the included routines are fetched dynamically, the current version is always used.

FILES

Files used are:

- `SYSIN` – primary input. RECFM F[B], LRECL=80.
- `SYSLIB` – secondary input; the library that contains the included routines (PDS or PDSE). RECFM F[B], LRECL=80.

- **SYSPRINT** – list file on which the expanded procedure is listed. **SYSPRINT** can be set to **DUMMY** no listing is required. This feature is useful for locating run-time errors. **RECFM F[B]A, LRECL=121, BLKSIZE=2420.**

Any messages and errors are written to the job log (WTO routing code 11).

INVOCATION

```
ADDRESS TSO "ALLOC F(SYSIN) DA(dsname) SHR REUS"
ADDRESS TSO "ALLOC F(SYSLIB) DA(dsname) SHR REUS"
ADDRESS TSO "ALLOC F(SYSPRINT) DA(dsname) NEW REUS options"
ADDRESS LINK "REXXPREP parm"
```

The optional parameter is passed to the expanded procedure where it can be obtained with the **PARSE ARG** varname statement.

MESSAGES AND ERRORS

The following messages and errors are output on the job log:

- *MEMBER memname NOT FOUND.* Memname not found on DD:SYSLIB. Program completes with return code -8.
- *SYSLIBMEMBERmemnameACCESS ERROR, REASON CODE: nnnn.* The DSERV GET service issued a non-zero return code with reason code nnnn (decimal) while attempting to fetch memname from DD:SYSLIB. Processing terminated with return code -28.
- *PDSE ALIAS: aliasname PRIMARY NAME: primaryname.* (Information). The name specified on the #INCLUDE instruction is longer than eight characters and so used as an alias name. The member with the associated primary name was fetched from DD:SYSLIB.

Other severe errors cause processing to be terminated immediately with the following completion codes (no explicit error message is issued):

- -12 – OPEN error on DD:SYSIN.
- -16 – line pointer buffer overflow.

- -20 – line buffer overflow.
- -24 – DD:SYSLIB not allocated.

OUTPUT LISTING

The expanded procedure is listed on DD:SYSPRINT when it is defined. The list entries have the following format:

- nnnnnn – source statement.
- nnnnnn= – resolved statement number.
- #INCLUDE statements are listed without statement number.

RESTRICTIONS

To simplify the program, the following restrictions have been made to the program:

- The INCLUDE statement must start in column 1. The member name must start in column 10.
- The included member must not itself include any further members.
- The two allocated buffers for the expanded source and the line pointers must have a fixed size of 1MB (BUFSIZE EQUate) and 100 KB (pBUFSIZE EQUate), respectively. These buffers are sufficiently large to contain 12,500 80-character records and their pointers.

The program shown below could easily be modified to handle these changes of detail.

EXAMPLE

```

Execution job
//S1      EXEC PGM=REXXPREP,PARM='1000 5 10'
//SYSLIB  DD   DSN=userid.library,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSIN   DD   *
/* REXX - INTEREST CALCULATION */

```

```

PARSE ARG x rate years
sum = COMPINT(x,rate,years)
SAY 'amount after 10 years:' sum
EXIT
#INCLUDE COMPINT

```

```

COMPINT SYSLIB member
/* calculate compound interest */
COMPINT:
  /* C(N) = C * Q ** N */
  PARSE ARG c, p, n
  /* c = initial capital */
  /* p = interest rate */
  /* n = number of years */
  NUMERIC DIGITS 6 /* precision */
  q = p/100
  ac = c /* initialize accumulated capital */
  DO i = 1 TO n
    ai = ac * q /* annual interest */
    ac = ac + ai
  END
RETURN ac

```

SYSPRINT output

```

1  PARSE ARG x rate years
2  /* REXX - INTEREST CALCULATION */
3  sum = COMPINT(x,rate,years)
4
5  SAY 'amount after 10 years:' sum
6  EXIT
#INCLUDE COMPINT
7  /* calculate compound interest */
8  COMPINT:
9    /* C(N) = C * Q ** N */
10   PARSE ARG c, p, n
11   /* c = initial capital */
12   /* p = interest rate */
13   /* n = number of years */
14   NUMERIC DIGITS 6 /* precision */
15   q = p/100
16   ac = c /* initialize accumulated capital */
17   DO i = 1 TO n
18     ai = ac * q /* annual interest */
19     ac = ac + ai
20   END
21  RETURN ac

```

SYSTSPRT (SAY) output

amount after 10 years: 1628.91

REXXPREP

Simple IF, ELSE, and EIF (End-IF) structured macros are included as in-line code:

* simple block macros (IF, ELSE, EIF)

```
MACRO
  IF    &COND,&KWD
.* IF (cond),THEN
.* cond: [N](E, P, Z, H, L, M, O)
.* kywd: not used
      GBLA  &IFH
      GBLC  &IFLBL(10)
      LCLA  &K

      LCLC  &C,&LBL

&IFH   SETA  &IFH+1

&C     SETC  '&COND(1) '

&LBL   SETC  'IF&SYSNDX '
&IFLBL(&IFH) SETC '&LBL '
      AIF   ('&C'(1,1) EQ 'N').A1
      JN&C  &LBL
      AGO   .A2

.A1    ANOP

&K     SETA  K,&C-1
&C     SETC  '&C'(2,&K)
      J&C   &LBL

.A2    MEND

MACRO
  EIF
.* End-IF (at the same hierarchical level)
      GBLA  &IFH
      GBLC  &IFLBL(10)
      LCLC  &LBL

&LBL   SETC  '&IFLBL(&IFH) '
&LBL   DS    0H
&IFH   SETA  &IFH-1
      MEND

MACRO
  ELSE
      GBLA  &IFH
      GBLC  &IFLBL(10)
      LCLC  &LBL1,&LBL2

&LBL1  SETC  '&IFLBL(&IFH) '
&LBL2  SETC  'IF&SYSNDX '
```

```

&IFLBL(&IFH) SETC '&LBL2'
          J      &LBL2
&LBL1    DS      ØH
          MEND

```

```

          TITLE 'REXX Preprocessor'

```

```

**
* REXXPREP loads an external REXX procedure from DD:SYSIN
* into memory and invokes the REXX interpreter with this procedure
* as command.
* The external REXX procedure can specify further external procedures
* (#INCLUDE membername) that are copied into memory from DD:SYSLIB.
*
* Note: Such included external procedures cannot themselves specify
*       any further external procedures (nesting level = 1).
*
* Syntax:
* #INCLUDE columns 1-8
* membername columns 10-41 (left-justified)
* A membername larger than 8 is considered to be a (case-sensitive)
* alias name for a PDSE member (the associated member will be fetched)
**
* Invocation: ADDRESS LINK "REXXPREP PARM"
*
* Return code:
* -8: #INCLUDE member cannot be fetched from PDS
* -12: OPEN error on DD:SYSIN
* -16: line pointer buffer overflow
* -20: line buffer overflow
* -24: DD:SYSLIB not allocated
* -28: PDSE access error on DD:SYSLIB
* other: return code from executed command
**
* Explicit DD statements:
*   SYSIN: primary input (RECFM F[B], LRECL=80, PS)
*   SYSLIB: secondary input (RECFM F[B], LRECL=80, PDS or PDSE)
*   SYSPRINT: list file (RECFM F[B]A, LRECL=121, PS)
**

```

```

          PRINT NOGEN
          SPACE
REXXPREP CSECT
REXXPREP AMODE 31
REXXPREP RMODE 24
          SPACE
* initialize addressing
          STM   R14,R12,12(R13)    save registers
          BALR  R12,Ø              base register
          USING *,R12
          LA   R15,SA              A(save-area)
          ST   R13,4(R15)         backward ptr

```

```

        ST    R15,8(R13)          forward ptr
        LR    R13,R15            A(new save-area)
* get parameter
        L     R2,0(R1)           A(exec parameter)
        LH    R1,0(R2)           L(parm data)
        LA    R0,2(R2)           A(parm data)
        STM   R0,R1,AARG
        SPACE
        MVC   RC,=H'-12'         RC: open error
        OPEN  (SYSIN,(INPUT))
        CH    R15,=H'4'
        JH    EXIT                open error
        MVC   RC,=H'0'
        OPEN  (SYSLIB,(INPUT))
        STH   R15,RCLIB
        OPEN  (SYSPRINT,(OUTPUT))
        STH   R15,RCPRINT
        MVC   PRLINE,=CL121' '   clear print-line buffer
        USING IHADCB,SYSLIB
* allocate line buffer
        L     R5,=A(BUFSIZE)
        STORAGE OBTAIN,LENGTH=(R5),ADDR=ABUF,LOC=ANY
        L     R6,ABUF
* allocate buffer for pointers
        L     R9,=A(PBUFSIZE)
        STORAGE OBTAIN,LENGTH=(R9),ADDR=AINSTPGM,LOC=ANY
        L     R8,AINSTPGM
        USING LINEPTR,R8
* read loop for source
GETLOOP GET  SYSIN
        USING INCDSECT,R1
        LR    R10,R1             save record address
        CLC   =C'#INCLUDE ',0(R1)
        IF    (E),THEN           include member
            LH  R15,RCLIB
            LTR R15,R15
            IF  (NZ),THEN
                MVC RC,=H'-24'   no SYSLIB defined
                J   EOJ
            EIF
            MVC MEMNAME,INCNAME
            CLI MEMNAMEX,C' '     test for extended name
            JNE LONGNAME         get name for PDSE alias
FIND     FIND SYSLIB,MEMNAME,D   find member
        LTR  R15,R15
        IF  (NZ),THEN
FINDERR  LA   R1,M1              msg: member not found
        BAL  R14,DISPLAY
        MVC  RC,=H'-8'
        J    GETLOOP

```

```

EIF
MVC PRLINE,=CL121' '   clear print line
MVC PRDATA(80),0(R10)
CLC =H'0',RCPRINT
IF (E),THEN           DD:SYSPRINT defined
    PUT SYSPRINT,PRLINE
EIF
READLOOP READ  DECB,SF,SYSLIB,BUF
        CHECK DECB
* get length of read block
    LH  R1,DCBBLKSI
    L   R2,DECB+16      A(IOB)
    SH  R1,14(R2)      unread length
* R1: actual block length
    SR  R0,R0
    LH  R15,DCBLRECL   record length
    DR  R0,R15
* R1: number of records in block
    LR  R4,R1
    LA  R2,BUF
DEBLOCK LR  R1,R2      extract individual SYSLIB records
        BAL R11,SETLINE move to line buffer
        LA  R2,80(R2)
        BCT R4,DEBLOCK
        J   READLOOP   get next SYSIN record
EOFLIB EQU GETLOOP
        ELSE ,         normal data line
        BAL R11,SETLINE
        J   GETLOOP
EIF
SPACE
EOFIN  DS  0H
* R8: current end of list
    S   R8,AINSTPGM   subtract start address
    ST  R8,LINSTPGM   size of entries
    SPACE
EOJ    CLOSE (SYSIN''SYSLIB''SYSPRINT)
* process (if no error)
    LH  R15,RC
    LTR R15,R15
    IF  (Z),THEN
        LOAD EP=IRXEXEC
        LR  R15,R0
        LA  R0,0
        CALL (15),(NULL,AARGLIST,FLAGS,AINSTBLK,NULL,AEVALBLK,NULL,X
            NULL),VL
        LH  R1,EVLEN+2
        LTR R1,R1
        IF  (NZ),THEN
            BCTR R1,0

```

```

        EX   R1,PACK
        CVB  R15,PL8
        STH  R15,RC
        EIF
        EIF
        SPACE
* release allocated memory
EXIT   ICM   R1,R15,ABUF
        IF    (NZ),THEN
            STORAGE RELEASE,LENGTH=BUFSIZE,ADDR=(R1)
        EIF
        ICM   R1,R15,AINSTPGM
        IF    (NZ),THEN
            STORAGE RELEASE,LENGTH=PBUFSIZE,ADDR=(R1)
        EIF
        SPACE 1
        LH   R15,RC                load program return code
        L    R13,4(R13)            restore A(old save-area)
        RETURN (14,12),RC=(15)
        SPACE 2
SA     DS    18A                    save area
        SPACE 1
* symbolic register equates
R0     EQU   0
R1     EQU   1
R2     EQU   2
R3     EQU   3
R4     EQU   4
R5     EQU   5
R6     EQU   6
R7     EQU   7
R8     EQU   8
R9     EQU   9
R10    EQU   10
R11    EQU   11
R12    EQU   12
R13    EQU   13
R14    EQU   14
R15    EQU   15
        SPACE 1
ERR1   MVC   RC,=H'-16'           line pointer buffer overflow
        J    E0J
        SPACE 1
ERR2   MVC   RC,=H'-20'           line buffer overflow
        J    E0J
        SPACE 1
PACK   PACK  PL8,EVDATA(0)
PL8    DS    PL8
        TITLE 'Subroutines'
LONGNAME DS  0H get (primary) member name for long PDSE alias

```

```

USING DESL,INLIST
MVC  ANAME,MEMNAME
MVC  DESL_NAME_PTR,=A(AENTRY) set pointer
DESERV FUNC=GET,AREAPTR=PTR,DCB=SYSLIB,CONN_INTENT=HOLD,      X
      RSNCODE=RSN,RETCODE=RTC,                                  X
      NAME_LIST=(INLIST,1)
LTR  R15,R15          test return code
IF   (NZ),THEN
  LH  R0,RSN+2        actual reason code
  CLC =X'03EA',RSN+2  member not found?
  JE  FINDERR         :yes
* otherwise general error
MVC  M2NAME,MEMNAME
CVD  R0,PL8
UNPK M2RSN,PL8
OC   M2RSN,=4C'0'    de-sign
LA   R1,M2
BAL  R14,DISPLAY
MVC  RC,=H'-28'      RC: access error
J    EXIT
EIF
L    R2,PTR          load DESB address
USING DESB,R2
LA   R2,DESB_DATA   address of data area
USING SMDE,R2
LH   R3,SMDE_PNAME_OFF offset to primary name
LA   R3,0(R2,R3)    address of primary name entry
USING SMDE_NAME,R3
LA   R0,SMDE_NAME_VAL primary name
LH   R1,SMDE_NAME_LEN length of primary name
O    R1,=X'40000000'
LA   R14,PNAME       target address
LA   R15,L'PNAME     length of target address
MVCL R14,R0          move to <pname>
MVC  MEMNAME,=CL32' '
MVC  MEMNAME(8),PNAME move to <memname>
SPACE
* display alias access
MVC  M3ANAME,ANAME
MVC  M3PNAME,PNAME
LA   R1,M3
BAL  R14,DISPLAY
SPACE
J    FIND            find primary member
SPACE 2
SETLINE DS 0H set line in buffer
* R1: A(line), CL80
RECLEN EQU 80        standard REXX record length
      LA R15,RECLEN(R1) end-of-record +1
* R15: A(end-of-record +1)

```

```

        LA    R14,RECLLEN-1
* R14: maximum no. of characters that can be removed
* remove trailing blanks
DEBLANK BCTR R15,Ø           bump down current pointer
        CLI  Ø(R15),C' '
        JNE  BREAK          non-blank found
        BCT  R14,DEBLANK
* R14: size of truncated line -1
BREAK   C    R9,=A(LLINEPTR) residual length of pointer buffer
        JL  ERR1           overflow
        ST  R6,ALINE
        LA  R14,1(R14)     length of source line
        ST  R14,LLINE
        MVC PRDATA,Ø(R1)   move line to print buffer
        LR  RØ,R1          A(source line)
        LR  R1,R14         L(source line)
        LR  R7,R1         target length
        CR  R7,R5         residual length in line buffer
        JH  ERR2           overflow
        MVCL R6,RØ        move line to target buffer
        LA  R8,LLINEPTR(R8)
        S   R9,=A(LLINEPTR)
        CLC =H'Ø',RCPRINT
        BNER R11          DD:SYSPRINT not defined
        AP  LINENO,=P'1'
        MVC PRLINENO,=X'4Ø2Ø2Ø2Ø2Ø2Ø2Ø2Ø'
        ED  PRLINENO,LINENO
        PUT SYSPRINT,PRLINE
        BR  R11
SPACE 2
DISPLAY DS  ØH display message on joblog
        STM R14,R2,REGS
* R1: A(message)
        LR  R2,R1
        WTO TEXT=(R2),ROUTCDE=(11)
        LM  R14,R2,REGS
        BR  R14
REGS    DS  5A           temporary save area for registers
        TITLE 'Data Areas'
* messages
M1      DC  AL2(M1E-M1MSG)
M1MSG   DC  C'MEMBER '
MEMNAME DS  CL32
MEMNAMEX EQU MEMNAME+8
        DC  C' NOT FOUND'
M1E     EQU  *
        SPACE 1
M2      DC  AL2(M2E-M2MSG)
M2MSG   DC  C'SYSLIB MEMBER '
M2NAME  DS  CL32

```

```

M2RSN    DC    C' ACCESS ERROR, REASON CODE:'
          DS    CL4
M2E      EQU    *
          SPACE 1
M3       DC    AL2(M3E-M3MSG)
M3MSG    DC    C'PDSE ALIAS:'
M3ANAME  DS    CL32
          DC    C' PRIMARY NAME:'
M3PNAME  DS    CL8
M3E      EQU    *
          SPACE 2
PTR      DS    A
LINENO   DC    PL4'Ø'
          SPACE
RTC      DS    F                return code
RSN      DS    XL4             reason code
          SPACE
INLIST   DC    XL(L,DESL_ENTRY)'Ø'
          SPACE
PNAME    DS    CL32             primary name
          SPACE
AENTRY   DC    AL2(32)         alias entry
ANAME    DS    CL32
          SPACE
RC       DS    H
RCLIB    DS    H                RC: OPEN SYSLIB
RCPRINT  DS    H                RC: OPEN SYSPRINT
          SPACE 1
NULL     DC    A(Ø)            zero address
          SPACE 1
AARGLIST DC    A(ARGLIST)
ARGLIST  DS    ØA
AARG     DS    A
LARG     DS    A
          DC    2F'-1'
          SPACE 1
FLAGS    DC    X'80000000'      invoke as command
AINSTBLK DC    A(INSTBLK)
AEVALBLK DC    A(EVALBLK)
          SPACE
EVALBLK  DS    ØF    align
          DS    F                reserved
EVSZIE   DC    A((EVDATAE-EVALBLK)/8)
EVLEN    DC    AL4(L,EVDATA)    L(data)
          DS    F                reserved
EVDATA   DS    CL64             field for returned data
EVDATAE  EQU    *
          SPACE
          DS    ØF    align
INSTBLK  DC    CL8'IRXINSTB'

```

```

        DC    F'128'          L(INSTBLK header)
        DS    F              reserved
AINSTPGM DS    A            address of INSTBLK entries
LINSTPGM DS    A            size of INSTBLK entries
        DC    CL8' '        member name (unused)
        DC    CL8' '        DD name (unused)
        DC    CL8'MVS'      subcom
        DS    F              reserved
        DC    F'Ø'          L(DSN), unused
        DC    CL54' '       DSN, unused
        ORG   INSTBLK+128
        SPACE
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,MACRF=GL,EODAD=EOFIN
SYSLIB  DCB   DDNAME=SYSLIB,DSORG=PO,MACRF=R,EODAD=EOFLIB
SYSPRINT DCB  DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,RECFM=FBA,      X
        LRECL=121,BLKSIZE=242Ø
        SPACE
KB      EQU   1Ø24
MB      EQU   1Ø24*KB
BUFSIZE EQU   1*MB
PBUFSIZE EQU  2ØØ*KB
ABUF    DC    A(Ø)
        SPACE
PRLINE  DC    CL121' '
        ORG   PRLINE
PRCTL   DS    C
PRLINENO DS   CL8
        DS    CL2
PRDATA  DS    CL8Ø
        SPACE
        LTORG
        SPACE
BUF     DS    CL3276Ø
        TITLE 'DSECTs'
LINEPTR DSECT
ALINE   DS    A            address of line
LLINE   DS    F            length of line
LLINEPTR EQU  *-LINEPTR    length of buffer entry
        SPACE
INCDSECT DSECT
INCCMD  DS    C'#INCLUDE '
INCNAME DS    CL32
        SPACE
        DCBD  DSORG=PS,DEV=DA
        IGWSMDE
        IGWDES
        END

```

Maintaining a DASD configuration

INTRODUCTION

The 'OPSREC' system is a group of procedures on MVS which are used to maintain our DASD configuration, control the initialization of disks, maintain the list of back-up suites for offsite recovery, and generate disk maps. The associated back-up system controls the back-up and restore of fullpack offsite back-ups (for D/R). The back-up suites are in the form 'BCKMVSx' where 'x' is a valid letter or number. There are up to 60 back-ups per suite (these suites will be arranged logically, eg by application). The system can be used just to maintain a list of volumes and control the initialization of them; if the back-up generation function is not required this could easily be removed.

DATASETS

The following datasets are used by the system (and, again, the associated off-site back-up system):

- BQIBI06.OPSREC.BACKUPS – contains a list of the back-up 'suites'; these are generated by an option within the system, after selecting 'M' from the main menu. They are DFDSS back-ups, but the skeletons can easily be tailored to use whatever utility is required.
- BQIBI06.OPSREC.CONTROL – contains the 'RECDSKM' member, the central control file used by this system, and a member for each system that requires back-ups to be generated, describing what back-up suites run there. The format of these two control files is described below.
- BQIBI06.OPSREC.GDGS – contains the GDG base definitions for the back-ups.
- BQIBI06.OPSREC.LISTING – contains the disk layout master listing and the latest (updated) disk listing. The 'DISKMAST' member is a master listing that has the values from RECDSKM

edited into it to create a disk listing. The listing shows the controller name/type and chpids that access the strings (by CPU).

- BQIBI06.OPSREC.MSGS – ISPF library (ISPMLIB).
- BQIBI06.OPSREC.PANELS – ISPF library (ISPPLIB).
- BQIBI06.OPSREC.REXX – ISPF library (SYSEXEC), REXX EXECs, and edit macros.
- BQIBI06.OPSREC.SKELS – ISPF library (ISPSLIB).

RECDSKM CONTROL MEMBERS

This control file contains a list of all current DASD addresses, their associated valid, DASD type, back-up suite that they are in, etc. Note that the first record is a header record (the REXX which processes the records expects this to be there, and will start from record 2).

```

ADDR  VOLID  TYPE  SUITE  COM  RECSTR  SNAP  SHR  =====COMMENTS=====
0700  TEST01  93   NONE   N   BILLING  0000  N   Old test volume
0701  TEST02  93   MVSG   Y   TESTING  0000  Y   New test volume

```

- ADDR – DASD address.
- VOLID – current valid. If the volume is a spare, this will be ‘MV’ followed by the 4-digit address. This is the naming convention for spare volumes at our site.
- TYPE – DASD type. 80 = 3380D, 81 = 3380E, 82 = 3380K, 93 = 3390m3.
- SUITE – back-up suite, eg ‘MVSA’ (for BCKMVSA suite), or ‘NONE’ if the volume is not backed up.
- COM – recovery site flag. ‘Y’ = restored at back-up site, ‘N’ = not restored at back-up site, ‘X’ = taken to back-up site but excluded from normal restore, ‘T’ = just initialized at back-up site (eg storage volumes).
- RECSTR – recovery string. Just used for documenting the use of the volume (system it is used by, etc). Could be used for reports or billing.


```

%PF3 - End
)INIT
.ZVARS = '(OPTYP)'
)PROC
VER (&OPTYP,NB,LIST,1,2)
)END

```

POPSR171

```

)ATTR
  ¬ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)
  £ TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
  $ TYPE(TEXT) INTENS(HIGH) CAPS(OFF)
  _ TYPE(INPUT) PAD(_)
)BODY WIDTH(80) EXPAND(@@)
%@-@ OPSREC - Processing on:¬sys %@-@
+
+
+          Select one of the following ==>_Z+
+
+          _____ OPSREC _____
+
+          1) +Disk Initialization
+          2) +Browse "RECDISK" Control File
+          3) +Edit "RECDISK" Control File
+          4) +Edit Disk Listing Master File
+          5) +Generate Updated Disk Listing
+          6) +Print Disk Listing
+          7) +Browse Disk Listing
+
+          M) +Maintain Backup Suites
+
+          _____
+
+          ¬mainmsg1          +
+          ¬mainmsg2          +
+
+
% PF3+to End.
)INIT
.ZVARS = '(OPR)'
)PROC
VER (&OPR,NB,LIST,1,2,3,4,5,6,7,M)
)END

```

POPSR172

```

)ATTR
  ¬ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)
  £ TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
  $ TYPE(TEXT) INTENS(HIGH) CAPS(OFF)

```



```

+
%      1)+List GDG Base Definitions      Listing (Long/Short):_GDGLS+
+
%      2)+Define GDG Base(s)           %New GDGs will NOT be defined
%                                         until AFTER bkup suite regen.
+
+ Disks initialized during this session:
+
¬opms1
¬opms2
¬opms3
¬opms4
¬opms5
+
%ENTER - Continue
%PF3   - End
)INIT
  .ZVARS = '(GDGFNC)'
)PROC
  VER (&GDGFNC,NB,LIST,1,2)
  VER (&GDGN,NB)
  IF (&GDGFNC = '1')
    VER (&GDGLS,NB,LIST,LONG,SHORT)
)END

```

POPSR179

```

)ATTR
  ¬ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)
  £ TYPE(OUTPUT) INTENS(LOW)  CAPS(OFF)
)BODY WIDTH(80) EXPAND(@@)
%@-@ OPSREC - Display Backup Affinities @-@
+
+
+
+

```

<pre> MVSSYS3 :¬MVSCGB ¬MVSCGC </pre>	<pre> +BCKMVS- run on here +Taken to RECOVERY site </pre>
<pre> MVSSYS1 :¬MVSFPB ¬MVSFPC </pre>	<pre> +BCKMVS- run on here +Taken to RECOVERY site </pre>
<pre> MVSSYS2 :¬MVSPRB ¬MVSPRC </pre>	<pre> +BCKMVS- run on here +Taken to RECOVERY site </pre>

```

+
+
+
+
%PF3   - End
)END

```

REXX

```

/* ===== */
/* EOPSREC: Driver for "OPSREC" Processing on MVS: */
/*      - Maintain DASD lists and main control file */
/*      - Perform DASD initializes */
/*      - Generate DASD maps */
/*      - Maintain offsite back-up suites and GDG cycles */
/*      - Generate DFDSS fullpack back-up suites */
/*      - Generate DFDSS fullpack back-up jobs, if wanted */
/*      */
/*      Some panels, msgs are common with the fullpack */
/*      backup system... */
/*      */
/* CONTROL FILES: */
/*      */
/*      BQIBI06.OPSREC.CONTROL(RECDSKM) */
/*      Contains the list of DASD by address. */
/*      */
/*      BQIBI06.OPSREC.CONTROL(@sys) */
/*      Where 'sys' is SYS1, SYS2 or SYS3. */
/*      Contains a list of the suite name suffixes */
/*      (one letter) for the backup suites associated */
/*      with each system, and a flag ('Y' or 'N') to */
/*      indicate whether or not BKUPSITE restore jobs */
/*      need to be generated for them. */
/*      */
/*      BQIBI06.OPSREC.LISTING */
/*      Contains the Disk Master List, and the latest */
/*      generated disk map. */
/* ===== */
/* Trace ir */
/* ----- */
/* Ensure we are trying to run the jobs on a known system */
/* ----- */
Address "TS0"
"WHEREAMI" /* So, where are we? */
retc = rc
sys = "???"
ttm = ""
mainmsg1 = ""
mainmsg2 = ""
msg1 = ">>> Disk updates have taken place. Please <<<"
msg2 = ">>> ensure Back-up Suites are re-generated! <<<"
If retc = 4 Then sys = "SYS2" /* SYS2 */
If retc = 8 Then sys = "SYS1" /* SYS1 */
If retc = 20 Then sys = "SYS3" /* SYS3 */

Address "ISPEXEC"
If sys = "" Then Do
  "ISPEXEC SETMSG MSG(MOPR170A)"

```

```

Return
End

userid = USERID()                               /* For notify/jobname */
If Left(userid,5) <> "BQIBI" Then Do             /* Just for systems */
  "ISPEXEC SETMSG MSG(MOPR170S)"
  Return
End
/* ----- */
/* Ensure the control datasets exist - if not, inform user... */
/* ----- */
Address "TSO"
listng = "BQIBI06.OPSREC.LISTING"
contrl = "BQIBI06.OPSREC.CONTROL"
gdgdef = "BQIBI06.OPSREC.GDGS"
ctlstat = SYSDSN("'"contrl"'")
libstat = SYSDSN("'"listng"'")
gdgstat = SYSDSN("'"gdgdef"'")

Address "ISPEXEC"

If libstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say " "
  Say " EOPSREC: Listing Files dataset cannot be located..."
  Say " "
  Return
End
If ctlstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say " "
  Say " EOPSREC: Control dataset cannot be located..."
  Say " "
  Return
End
If gdgstat = "DATASET NOT FOUND" Then Do /* Dataset doesn't exist... */
  Say " "
  Say " EOPSREC: GDG Defines dataset cannot be located..."
  Say " "
  Return
End
opms1 = ""
opms2 = ""
opms3 = ""
opms4 = ""
opms5 = ""
opct = 0
/* ----- */
/* Read in the control record for the system we're running on */
/* (contains the list of back-up suites that are valid for THIS */
/* system)... */
/* ----- */
this_systems_backups = ""

```

```

Address "TS0"
ctl = contrl("@sys")
"ALLOC FI(TEMP1) DA('ctl') SHR"
"EXECIO * DISKR TEMP1 (Stem okbkups. FINIS"
"FREE FI(TEMP1)"
If okbkups.0 = 0 Then Do
  Say " "
  Say " EOPSREC: The 'System Identification Record' for "sys
  Say "          cannot be located... Contact Systems..."
  Say " "
  Return
End

this_systems_backups = okbkups.1
/* ----- */
/* This is the main control section of this program.      */
/* ----- */
Do Forever
  opr = ""
  "ISPEXEC DISPLAY PANEL(POPSR171)"
  If RC = 8 Then Leave
  Select
    When opr = '1' Then
      Call INITIALIZE_DISKS
    When opr = '2' Then
      Call BROWSE_CONTROL
    When opr = '3' Then
      Call EDIT_CONTROL
    When opr = '4' Then
      Call EDIT_MASTER
    When opr = '5' Then
      Call GENERATE_LISTING
    When opr = '6' Then
      Call PRINT_LISTING
    When opr = '7' Then
      Call BROWSE_LISTING
    When opr = 'M' Then
      Call SUITE_MAINT
    Otherwise Nop
  End
End
Return
/* Do Forever */
/* Bye bye for now... */
/* ++++++ */

INITIALIZE_DISKS:
/* ===== */
/* This routine does the following: */
/* a) Displays a panel for the user to select either: */
/*    1) Initialize a current spare volume as an MVS volume */
/*    2) Initialize a current MVS volume as a spare */
/* b) Calls the relevant sub-routine as requested. */

```

```

/* ===== */
Address "ISPEXEC"
Do Forever
  optyp = ""
  "ISPEXEC DISPLAY PANEL(POPSR170)"      /* Request initialize type */
  If rc = 8 Then                          /* RC8 = PF3 */
    Return
  Select
    When optyp = "1" Then
      Call INITIALIZE_MVSVOL
    When optyp = "2" Then
      Call INITIALIZE_SPARE
  End
End
Return

```

```

/* ++++++ */

```

```

INITIALIZE_MVSVOL:
/* ===== */
/* Initialize a volume currently marked as spare with a proper */
/* valid. */
/* */
/* This routine does the following: */
/* a) Reads in the 'BQIBI06.OPSREC.CONTROL(RECDSKM)' control */
/* file. */
/* b) Displays the panel to request the address of the volume */
/* to be initialized. */
/* c) Checks that the address is a spare (denoted by a valid */
/* of 'MV' followed by the 4-character address. */
/* d) Displays another panel showing the details of the init- */
/* ialization and giving the user the chance to quit. */
/* e) Creates the initialization job using a skeleton and */
/* submits it. */
/* f) Edits the control file to update the new valid and to */
/* insert '?'s in other fields such as backup suite (note */
/* that the edit user is left in edit here, to update the */
/* fields with '?'s if they wish. */
/* ===== */

```

```

Address "TS0"
recdsm. = ""
"ALLOC FI(TEMP1) DA('BQIBI06.OPSREC.CONTROL(RECDSKM)') SHR"
"EXECIO * DISKR TEMP1 (Stem recdsm. FINIS"
"FREE FI(TEMP1)"

```

```

Address "ISPEXEC"
sofar = ""
opcu = ""
opvl = ""
Do Forever

```

```

flag = "N"
Do Until flag <> "N"
  "ISPEXEC DISPLAY PANEL(POPSR172)" /* Request disk address */
  If rc = 8 Then /* RC8 = PF3 */
    Return
  If Length(opcu) = 3 Then /* Pad 3 byte ccuu... */
    opcu = "Ø"opcu
  Do a = 1 to recdskm.Ø
    Parse Upper Var recdskm.a rcadd rcvol rcdev rcsuite rccom rcrec .
    If opcu = rcadd Then Do
      flag = "F" /* Found requested ccuu */
      Leave
    End
  End
End
If flag = "F" Then /* It's found OK... */
  If rcvol <> "MV"rcadd Then /* ...but it's not a spare */
    flag = "S"
  If flag = "N" Then /* ...not found */
    "ISPEXEC SETMSG MSG(MOPR17ØH)"
  If flag = "S" Then Do /* ...not a 'real' spare... */
    "ISPEXEC SETMSG MSG(MOPR17ØI)"
    flag = "N"
  End
  If Pos(opcu,sofar) <> Ø Then Do /* Addr already done? */
    "ISPEXEC SETMSG MSG(MOPR17ØK)" /* Yes - disallow */
    flag = "N"
  End
End

If flag <> "N" Then Do /* OK (so far)... */
  opdevt = ""
  Select
    When rcdev = "8Ø" Then
      opdevt = "338ØD"
    When rcdev = "81" Then
      opdevt = "338ØE"
    When rcdev = "82" Then
      opdevt = "338ØK"
    When rcdev = "93" Then
      opdevt = "339Øm3"
    Otherwise
      opdevt = "??????"
  End
  opcrv1 = "MV"opcu
  opsu = rcsuite

  "ISPEXEC DISPLAY PANEL(POPSR173)" /* Show details... */
  If rc = 8 Then Do /* RC8 = PF3 */
    flag = "N"
    "ISPEXEC SETMSG MSG(MOPR17ØJ)" /* Say we have PF3ed out... */
    Leave
  End
End

```

```

sofar = sofar" "opcu          /* Volumes init'ed so far */
opct = opct + 1
opms1 = "Volumes Initialized so Far ("opct"):"
If opct < 6 Then
  opms2 = opms2" "opcu "as" opv1" "
Else
If opct < 11 Then
  opms3 = opms3" "opcu "as" opv1" "
Else
If opct < 16 Then
  opms4 = opms4" "opcu "as" opv1" "
Else
If opct > 15 Then
  opms5 = opms5" "opcu "as" opv1" "
"ISPEXEC FTOPEN TEMP"          /* Open ZTEMPF */
"ISPEXEC FTINCL SOPSR170"     /* Do File Tailoring */
"ISPEXEC FTCLOSE"            /* Close temp file */
Call SUBMIT_JOB                /* Go and submit it */
"ISPEXEC SETMSG MSG(MOPR170L)" /* Say it's subbed */

"ISPEXEC VPUT (opcu opv1 rcdev rcsuite rccom rcrec)"
"ISPEXEC EDIT DATASET('BQIBI06.OPSREC.CONTROL(RECD SKM)'),
  MACRO(DISKUP2)"

mainmsg1 = msg1
mainmsg2 = msg2

End
End          /* Do until... */
End          /* Do forever */
Return
/* ++++++ */

INITIALIZE_SPARE:
/* ===== */
/* Re-initialize a volume (that is currently marked in the */
/* Control File as being in-use) as a Spare. */
/* */
/* This routine does the following: */
/* a) Reads in the 'BQIBI06.OPSREC.CONTROL(RECD SKM)' control */
/* file. */
/* b) Displays an extra panel to double-check that it is OK */
/* to re-initialize a 'production' volume as a spare. */
/* c) Displays a panel to request the ccuu and new volid. */
/* d) Checks to make sure that the volume being initialized */
/* is not already a spare. */
/* e) Submits a job to carry out the initialize. */
/* f) Edits the control file to update the new volid, etc. */
/* ===== */

Address "ISPEXEC"

```

```

opse1 = ""
"ISPEXEC DISPLAY PANEL(POPSR175)"          /* Make sure they REALLY */
If rc = 8 Then Do                          /* want to do this... */
    "ISPEXEC SETMSG MSG(MOPR170N)"
    Return
End
If opse1 = "NO" Then Do
    "ISPEXEC SETMSG MSG(MOPR170N)"
    Return
End

Address "TS0"
recdskm. = ""
"ALLOC FI(TEMP1) DA('BQIBI06.OPSREC.CONTROL(RECDSKM)') SHR"
"EXECIO * DISKR TEMP1 (Stem recdskm. FINIS"
"FREE FI(TEMP1)"
sofar = ""
opms1 = ""
opms2 = ""
opms3 = ""
opms4 = ""
opms5 = ""
opcu = ""
opct = 0
Address "ISPEXEC"

Do Forever
    flag = "N"
    Do Until flag <> "N"
        "ISPEXEC DISPLAY PANEL(POPSR174)" /* Request disk address */
        If rc = 8 Then                    /* RC8 = PF3 */
            Return
        If Length(opcu) = 3 Then          /* Pad 3 byte cuu to ccuu */
            opcu = "0"opcu
        Do a = 1 to recdskm.0
            Parse Var recdskm.a rcadd rcvol rcdev rcsuite rccom rcrc . rcsh rccm
            If opcu == rcadd Then Do      /* MUST use '==' here... */
                flag = "F"              /* Found requested ccuu */
                Leave
            End
        End
        If flag = "F" Then                /* It's found OK... */
            If rcvol = "MV"rcadd Then    /* ...but it is not a spare */
                flag = "S"
            If flag = "N" Then
                "ISPEXEC SETMSG MSG(MOPR170H)" /* tut tut - not found */
            If flag = "S" Then Do
                "ISPEXEC SETMSG MSG(MOPR170M)" /* tut tut - already spare */
                flag = "N"
            End
            If Pos(opcu,sofar) <> 0 Then Do /* Addr already done? */

```

```

"ISPEXEC SETMSG MSG(MOPR170K)" /* Yes - disallow */
flag = "N"
End

If flag <> "N" Then Do /* OK (so far)... */
opdevt = ""
Select
When rcdev = "80" Then
opdevt = "3380D"
When rcdev = "81" Then
opdevt = "3380E"
When rcdev = "82" Then
opdevt = "3380K"
When rcdev = "93" Then
opdevt = "3390m3"
Otherwise
opdevt = "??????"
End
opcrv1 = rcv01
opv1 = "MV"opcu
opsu = rcsuite

"ISPEXEC DISPLAY PANEL(POPSR173)" /* Show details... */
If rc = 8 Then Do /* RC8 = PF3 */
flag = "N"
"ISPEXEC SETMSG MSG(MOPR170J)" /* Say we have PF3ed out... */
Leave
End

sofar = sofar" "opcu /* Volumes init'ed so far */
opct = opct + 1
opms1 = "Volumes Initialized so Far ("opct"):"
If opct < 6 Then
opms2 = opms2" "opcu "as" opv1" "
Else
If opct < 11 Then
opms3 = opms3" "opcu "as" opv1" "
Else
If opct < 16 Then
opms4 = opms4" "opcu "as" opv1" "
Else
If opct > 15 Then
opms5 = opms5" "opcu "as" opv1" "

"ISPEXEC FTOPEN TEMP" /* Open ZTEMPF */
"ISPEXEC FTINCL SOPSR170" /* Do file tailoring */
"ISPEXEC FTCLOSE" /* Close temp file */
Call SUBMIT_JOB /* Go and submit it */
"ISPEXEC SETMSG MSG(MOPR170L)" /* Say it's subbed */
rccm = Strip(rccm)
"ISPEXEC VPUT (opcu opcrv1 rcdev rcsuite rccom rcrec rcsh rccm)"

```

```

        "ISPEXEC EDIT DATASET('BQIBI06.OPSREC.CONTROL(RECDSKM)'),
          MACRO(DISKUP4)"
        mainmsg1 = msg1
        mainmsg2 = msg2
    End
End                                         /* Do until..          */
End                                         /* Do forever...       */
Return
/* ++++++ */
BROWSE_CONTROL:
Address "ISPEXEC"
"ISPEXEC BROWSE DATASET('BQIBI06.OPSREC.CONTROL(RECDSKM)')"
Return
/* ++++++ */
EDIT_CONTROL:
Address "ISPEXEC"
"ISPEXEC EDIT DATASET('BQIBI06.OPSREC.CONTROL(RECDSKM)')
MACRO(DISKUP3)"
Return
/* ++++++ */
EDIT_MASTER:
/* ===== */
/* This routine does the following:          */
/*                                           */
/* a) Edits 'BQIBI06.OPSREC.LISTING(DISKMAST)'. */
/* ===== */

Address "ISPEXEC"
"ISPEXEC EDIT DATASET('BQIBI06.OPSREC.LISTING(DISKMAST)')"
Return
/* ++++++ */
GENERATE_LISTING:
/* ===== */
/* This routine does the following:          */
/*                                           */
/* a) Reads in 'BQIBI06.OPSREC.CONTROL(RECDSKM)' to obtain */
/*     the latest address/volid/status information.          */
/* b) Reads in 'BQIBI06.OPSREC.LISTING(DISKMAST)' to obtain */
/*     the number of records (necessary for the 'DISKUP'     */
/*     edit macro to issue the 'REP' command).              */
/* c) VPUTs each address/volid pair into the ISPF profile.  */
/* d) Edits 'BQIBI06.OPSREC.CONTROL(DISKMAST)' using the   */
/*     'DISKUPDT' Edit Macro, to update the disk map.       */
/* ===== */
Address "TSO"
recdsm. = ""
"ALLOC FI(TEMP1) DA('BQIBI06.OPSREC.CONTROL(RECDSKM)') SHR"
"EXECIO * DISKR TEMP1 (Stem recdsm. FINIS"
"FREE FI(TEMP1)"

diskmap. = ""
"ALLOC FI(TEMP1) DA('BQIBI06.OPSREC.LISTING(DISKMAST)') SHR"

```

```

"EXECIO * DISKR TEMP1 (Stem diskmap. FINIS"
"FREE FI(TEMP1)"
mapsz = diskmap.0
diskmap. = ""

Address "ISPEXEC"
"ISPEXEC CONTROL DISPLAY LOCK"
"ISPEXEC DISPLAY MSG(MOPR170B)"          /* Show we are doing it */

Do a = 2 to recdskm.0                    /* Ignore the header */
  Parse Upper Var recdskm.a rcaddr rcvol rcdevt . . . . rcshr .
  rcvol = Left(rcvol" ",6)                /* Ensure 6 bytes long */
  Interpret "rcad"a" = rcaddr"
  Interpret "rcvl"a" = rcvol"
  Interpret "rcsr"a" = rcshr"
  "ISPEXEC VPUT (rcad"a")"                /* Save address... */
  "ISPEXEC VPUT (rcvl"a")"                /* ...and VOLID */
  "ISPEXEC VPUT (rcsr"a")"                /* ...and shared flag */
End

recd0 = recdskm.0
"ISPEXEC VPUT (recd0)"
"ISPEXEC VPUT (mapsz)"

"ISPEXEC EDIT DATASET('BQIBI06.OPSREC.LISTING(DISKMAST)')
MACRO(DISKUP)"
dsrc = rc
If dsrc > 8 Then
  "ISPEXEC SETMSG MSG(MOPR170D)"          /* Failure... */
Else
  "ISPEXEC SETMSG MSG(MOPR170C)"          /* Success... */
Return

/* ++++++ */

PRINT_LISTING:
/* ===== */
/* This routine does the following: */
/* */
/* a) Prints 'BQIBI06.OPSREC.LISTING(DISK2)' to SYSPROG1. */
/* ===== */

Address "TS0"
"ALLOC FI(TEMP1) DA('BQIBI06.OPSREC.LISTING(DISK2)') SHR"
"EXECIO * DISKR TEMP1 (Stem mapprt. FINIS"
"FREE FI(TEMP1)"
"ALLOC FI(PRTIT) DEST(SYSPROG1) SYSOUT(A) NOHOLD RECFM(F A)"
"EXECIO * DISKW PRTIT (Stem mapprt. FINIS"
"FREE FI(PRTIT)"

Address "ISPEXEC"
"ISPEXEC SETMSG MSG(MOPR170E)"

```

Return

```
/* ++++++ */
```

BROWSE_LISTING:

```
/* ===== */  
/* This routine browses the latest Disk Listing (may be useful */  
/* after re-generating the latest list). */  
/* ===== */
```

Address "ISPEXEC"

"ISPEXEC BROWSE DATASET('BQIBI06.OPSREC.LISTING(DISK2)')"

Return

```
/* ++++++ */
```

SUBMIT_JOB:

```
/* ===== */  
/* Submit the job built in 'ZTEMPF'. */  
/* ===== */
```

Address "ISPEXEC"

"ISPEXEC VGET (ZTEMPF)" /* Get temp filename */

Address "TSO"

"SUBMIT "ztempf"

Return

```
/* "ISPEXEC EDIT DATASET('"ztempf"')" */ /* Debugging... */
```

```
/* ++++++ */
```

SUITE_MAINT:

```
/* ===== */  
/* This routine does the following: */  
/* */  
/* a) Allows Browse of the back-up suites. */  
/* b) Allows Edit of the back-up suites. */  
/* c) Calls another routine to re-generate the back-up suites. */  
/* d) Define GDG base(s) for recently added back-ups. */  
/* ===== */
```

Address "ISPEXEC"

Do Forever

 suitefnc = ""

 "ISPEXEC DISPLAY PANEL(POPSR176)" /* Request initialize type */

 If rc = 8 Then /* RC8 = PF3 */

 Return

 If suitefnc = "1" Then

 Call GEN_SUITES

 If suitefnc = "2" Then

 Call DEF_GDGS

 If suitefnc = "3" Then

```

    "ISPEXEC BROWSE DATASET('BQIBI06.OPSREC.BACKUPS')"
  If suitefnc = "4" Then Do
    "ISPEXEC EDIT DATASET('BQIBI06.OPSREC.CONTROL(RECDSKM)'),
      MACRO(DISKUP2)"
  End
  If suitefnc = "5" Then
    Call DISPLAY_AFFINITY
  If suitefnc = "6" Then
    "ISPEXEC BROWSE DATASET('BQIBI06.OPSREC.CONTROL(@*)')"
  If suitefnc = "7" Then
    "ISPEXEC EDIT DATASET('BQIBI06.OPSREC.CONTROL(@*)')"
  End
Return

/* ++++++ */

GEN_SUITES:
/* ===== */
/* This routine does the following: */
/* */
/* a) Allows re-generation of the back-up suites (all of them, */
/* no individual suite option). This can be Background (ie */
/* batch job) or Foreground (ie under TSO). */
/* ===== */
Address "ISPEXEC"

foreback = "F" /* Default = foreground */
"ISPEXEC DISPLAY PANEL(POPSR177)" /* Request initialize type */
If rc = 8 Then Do /* RC8 = PF3 */
  "ISPEXEC SETMSG MSG(MOPR1700)"
  Return
End

If foreback = "B" Then Do /* Run in Background */
  "ISPEXEC FTOPEN TEMP" /* Open ZTEMPF */
  "ISPEXEC FTINCL SOPSR171" /* Do File Tailoring */
  "ISPEXEC FTCLOSE" /* Close temp file */
  Call SUBMIT_JOB /* Go and submit it */
  "ISPEXEC SETMSG MSG(MOPR170R)" /* Say it's subbed */
End

If foreback = "F" Then Do /* Run in Foreground */
  Address "TSO"
  "%GENBKUPS" /* Call the REXX EXEC */
  If rc <> 0 Then Do
    Address "ISPEXEC"
    "ISPEXEC SETMSG MSG(MOPR170P)" /* Something funny... */
  End
  Else Do
    Address "ISPEXEC"
    "ISPEXEC SETMSG MSG(MOPR170Q)" /* It is OK... */
  End
End

```

```

    mainmsg1 = ""
    mainmsg2 = ""
End
End
Return
/* ++++++ */

DEF_GDGS:
/* ===== */
/* This routine does the following: */
/* */
/* a) Allows listing of GDG base(s) for backup suites. */
/* b) Allows easy definition of GDG base(s) for backup suites. */
/* In this option a temporary LIBDEF is set up and a job */
/* is included to do the defines. There is an imbedded in- */
/* clude in the define job which will pull in the defines */
/* from the LIBDEFed library ('BQIBI06.OPSREC.GDGS') for */
/* the requested suite. */
/* ===== */

Address "ISPEXEC"
gdgls = "SHORT"
Do Forever
  gdgfn = ""
  gdgn = ""
  "ISPEXEC DISPLAY PANEL(POPSR178)"
  If rc = 8 Then /* RC8 = PF3 */
    Return

  /* Make sure its a valid b/u suite for this system */

  bit = Right(gdgn,1)
  If Pos(bit,this_systems_backups) = 0 Then Do
    "ISPEXEC SETMSG MSG(MOPR170V)" /* Say it is no good */
    gdgfn = 'X' /* Skip other checks */
  End

  If gdgfn = "1" Then Do /* List GDG Base(s) */
    Address "TSO" /* Display current GDGs */
    type = "VOL" /* Default listing type */
    If gdgls = "LONG" Then
      type = "ALL"
    "LISTC "type" LVL(SYS8."sys".BCK"gdgn)"
  End

  If gdgfn = '2' Then Do /* Define GDG Base(s) */
    suitenam = "BCK"gdgn
    Address "ISPEXEC"
    "ISPEXEC LIBDEF ISPSLIB DATASET ID('BQIBI06.OPSREC.GDGS')"
    "ISPEXEC FTOPEN TEMP" /* Open ZTEMPF */
    "ISPEXEC FTINCL SOPSR172" /* Include def job */

```

```

incrc = rc                                /* Save rc          */
"ISPEXEC FTCL0SE"                          /* Close temp file  */
If incrc <> 0 Then                          /* Couldn't do include*/
  "ISPEXEC SETMSG MSG(MOPR170T)"          /* Tell them...     */
Else Do
  Call SUBMIT_JOB                          /* Sub define job   */
  "ISPEXEC SETMSG MSG(MOPR170U)"          /* Say its subbed   */
  "ISPEXEC LIBDEF ISPSLIB"               /* Reset LIBDEF     */
End
End
End                                         /* Do forever       */

```

```

/* ++++++

```

DISPLAY_AFFINITY:

```

/* ===== */
/* This routine does the following:          */
/*                                           */
/* a) Reads each of the members that start with a '@' in DSN */
/*     'BQIBI06.OPSREC.CONTROL'. These are the 'System Affinity */
/*     Control Files' which show 1) if a back-up suite can be */
/*     run on a system and 2) whether it is taken to BKUPSITE. */
/* b) Formats the members read and displays a panel.          */
/* ===== */

```

Address "TS0"

```

ctl = contrl"@SYS3)"                      /* For MVSSYS3      */
"ALLOC FI(TEMP1) DA('ctl') SHR"
mvscg. = ""
mvscgb = "*NONE*"                         /* Defaults - should */
mvscgc = "*NONE*"                         /* be overwritten... */
"EXECIO * DISKR TEMP1 (Stem mvscg. FINIS"
"FREE FI(TEMP1)"
If mvscg.1 <> "" Then mvscgb = mvscg.1
If mvscg.2 <> "" Then mvscgc = mvscg.2

```

```

ctl = contrl"@SYS1)"                      /* For MVSSYS1      */
"ALLOC FI(TEMP1) DA('ctl') SHR"
mvsf. = ""
mvsfpb = "*NONE*"                         /* Defaults - should */
mvsfpc = "*NONE*"                         /* be overwritten... */
"EXECIO * DISKR TEMP1 (Stem mvsf. FINIS"
"FREE FI(TEMP1)"
If mvsf.1 <> "" Then mvsfpb = mvsf.1
If mvsf.2 <> "" Then mvsfpc = mvsf.2

```

```

ctl = contrl"@SYS2)"                      /* For MVSSYS2      */
"ALLOC FI(TEMP1) DA('ctl') SHR"
mvspr. = ""
mvsprb = "*NONE*"                         /* Defaults - should */
mvsprc = "*NONE*"                         /* be overwritten... */

```

```

"EXECIO * DISKR TEMP1 (Stem mvspr. FINIS"
"FREE FI(TEMP1)"
If mvspr.1 <> "" Then mvsprb = mvspr.1
If mvspr.2 <> "" Then mvsprc = mvspr.2

Address "ISPEXEC"

"ISPEXEC DISPLAY PANEL(POPSR179)" /* Display values */
Return
/* ===== */
/* DISKUP : Edit Macro */
/* */
/* Update the Disk Master Listing to create a Disk */
/* Map. Saves the updated map as 'DISK2'. */
/* ===== */
Address "ISPEXEC"
"ISREDIT MACRO"
"ISPEXEC VGET (recd0 mapsz)" /* Get counts... */
Do a = 2 to recd0
  "ISPEXEC VGET (rcad"a" rcvl"a" rcsr"a)" /* Get each record */
  Interpret "ad = rcad"a
  Interpret "v1 = rcvl"a
  Interpret "sh = rcsr"a
  Select
  When sh = "Y" & Left(v1,2) = "MV" then /* Shared and Spare */
  "ISREDIT CHANGE ' @ad" ' '*SPARE ' FIRST" /* Change in master list*/
  When sh = "Z" & Left(v1,2) = "MV" then /* Shared and Spare */
  "ISREDIT CHANGE ' @ad" ' '@SPARE ' FIRST" /* Change in master list*/
  When sh = "Y" & Left(v1,2) <> "MV" then /* Shared and non-Spare */
  "ISREDIT CHANGE ' @ad" ' '*v1"' FIRST" /* Change in master list*/
  When sh = "Z" & Left(v1,2) <> "MV" then /* Shared and non-Spare */
  "ISREDIT CHANGE ' @ad" ' '@v1"' FIRST" /* Change in master list*/
  When sh = "N" & Left(v1,2) = "MV" then /* Non-Sh and Spare */
  "ISREDIT CHANGE ' @ad" ' ' SPARE ' FIRST" /* Change in master list*/
  When sh = "N" & Left(v1,2) <> "MV" then /* Non-Sh and non-Spare */
  "ISREDIT CHANGE ' @ad" ' ' v1"' FIRST" /* Change in master list*/
  Otherwise
  Do
  Say ad v1 sh 'INCORRECT SHR FLAG SET IN RECDISK CONTROL'
  Exit 99
  End
End
End

thedata = ""Date()" AT "Time()""
"ISREDIT CHANGE @@DATE@@ "thedata" FIRST" /* Edit in dates... */
thedata = ""Substr(Date("m"),1,3)""
"ISREDIT CHANGE @M@ "thedata" FIRST"
"ISREDIT REPLACE DISK2 1 "mapsz /* Replace "DISK2" mamber */
"ISREDIT CAN"
Return

```

```

/* ===== */
/* */
/* DISKUP2 : Edit Macro */
/* */
/* Update an entry in 'BQIBI06.OPSREC.CONTROL(RECDSKM)' */
/* that was previously a spare to show its new volid */
/* following initialization. */
/* */
/* The previous volid of MVccuu will be changed to */
/* the newly initialized volid, and the Suite Name, */
/* BKUPSITE flag and Recovery String will all be */
/* changed to '?'s. When the back-up suites are gen- */
/* erated these '?'s will be flagged as errors, so */
/* ensuring that the correct values are inserted. */
/* */
/* ===== */
Address "ISPEXEC"
"ISREDIT MACRO"
"ISREDIT TABSLINE = <1,-,7,-,15,-,19,-,26,-,29,->"
"ISPEXEC VGET (opcu opvl rcdev rcsuite rccom rcrec)" /* Get vars */
from = "'opcu" MV"opcu" "rcdev" "rcsuite" "rccom" "rcrec"'
to = "'opcu" "opvl" "rcdev" "???" ? "????????'"

"ISREDIT CHANGE" from to "FIRST"
"ISREDIT SAVE" /* Save changes */
/*
"ISREDIT END" /* Comment out "END" to leave us in Edit */
*/
Return

/* ===== */
/* DISKUP3 : Edit Macro */
/* */
/* Set correct TABS positions when editing 'RECDSKM'. */
/* ===== */
Address "ISPEXEC"
"ISREDIT MACRO"
"ISREDIT TABSLINE = <1,-,7,-,15,-,19,-,26,-,29,->"
/*
"ISREDIT END" */
Return

/* ===== */
/* DISKUP4 : Edit Macro */
/* */
/* Update an entry in 'BQIBI06.OPSREC.CONTROL(RECDSKM)' */
/* that was previously not a spare to be marked as a */
/* spare (following initialization). Reset eveything */
/* including any comments. */
/* */
/* The previous volid will be changed to MVccuu. */
/* ===== */

```

```

trace n
Address "ISPEXEC"
"ISREDIT MACRO"

/* Get vars */
"ISPEXEC VGET (opcu opcrvl rcdev rcsuite rccom rcrec rcsh rccm)"
from = "'opcu" "opcrvl" "rcdev" "rcsuite" "rccom" "left(rcrec,8)"
from = from" 0000 "rcsh" "rccm"'"

rcrec = "???????"
If Left(opcu,2) = "04" Then
  rcrec = "HDS"
If Left(opcu,2) = "07" Then
  rcrec = "RVA"
If Left(opcu,1) = "1" Then
  rcrec = "SVA"
comnt = " " /* 34 spaces... */
to = "'opcu" MV"opcu" "rcdev" NONE N "rcrec" 0000"
to = to" "rcsh" "comnt"'"
"ISREDIT CHANGE "from to" FIRST"
"ISREDIT END"

Return
/* ===== */
/* GENBKUPS: Create Full Pack DFDSS backups suites, using the */
/* RECDSKM member as source. */
/* */
/* Also creates the 'DEF GDG' statements for each */
/* of the back-up suites in 'BQIBI06.OPSREC.GDGS', in */
/* the same member name as is used for the back-ups */
/* (ie BCKMVSx, where 'x' is the suite suffix). */
/* */
/* Note that 'BCKMVS@' is a special suite which will */
/* be generated containing any disks which have a '?' */
/* in 'RECDSKM'. This implies that the disk has been */
/* initialized, but that no BCKMVS- suite has yet */
/* been assigned to it. */
/* */
/* If the 'gen_jobs' variable is set to 'Y' then the */
/* actual back-up JCL will be created, allowing a job */
/* scheduler to submit the back-ups, instead of the */
/* Operators via panels. Note that this function is */
/* only valid when a parm of 'B' is passed, implying */
/* we are running in Batch, as the process is quite */
/* slow. */
/* ===== */
/* Trace ir */
Parse Upper Arg parm .

/* ----- */
/* These are valid names which may be used for back-up suites. */
/* ----- */

```

```

bkups = "A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2"
bkups = bkups" 3 4 5 6 7 8 9 @"
gen_jobs = "N" /* Do/do not generate actual jobs */

entries = Words(bkups) /* Number of entries to init */
badones = "N" /* No dodgy ones found (yet) */
df1 = " DEF GDG(NAME(SYS8.&SYS..BCKMVS"
df2 = ") LIMIT(2) SCRATCH NOEMPTY)"
code = 0
If parm = "B" Then Do
  Say ">>> Starting Back-up Suite, GDG Base and Fullpack JCL
Generation:"
  Say ">>> ====="
End
Else Do
  Say ">>> Starting Back-up Suite and GDG Base Generation:"
  Say ">>> ====="
End

Do a = 1 to entries
  suffix = Word(bkups,a) /* Get each suffix */
  Interpret "MVS"suffix".0 = '" /* Initialize all tables... */
  Interpret "MVS"suffix"BKP = 0" /* ...total disks for suite */
End

/* ----- */
/* Read the "BQIBI06.OPSREC.CONTROL(RECDSKM)" member, and */
/* create the relevant data in "BQIBI06.OPSREC.BACKUPS". */
/* ----- */

Address "TS0"
"ALLOC FI(INPUT) DA('BQIBI06.OPSREC.CONTROL(RECDSKM)') SHR"
"EXECIO * DISKR INPUT (Stem inpt. FINIS"
"FREE FI(INPUT)"

/* ----- */
/* Read in the data from "RECDSKM" and split out into relevant */
/* back-up suite stem variables (format is "MVSx.", where 'x' is*/
/* the suite name from the table above)... */
/* ----- */

Do a = 1 to inpt.0
  Parse Upper Var inpt.a addr valid type suite com recstr .
  If suite = "NONE" Then /* Don't create non-existent members! */
    Iterate
  If addr = "*" Then /* Ignore comments cards */
    Iterate
  If addr = "ADDR" Then /* Ignore headers */
    Iterate

  suffix = Right(suite,1)

```

```

If suffix = "?" Then Do
  suffix = "@"
  badones = "Y"
End
Interpret "MVS"suffx"BKP = MVS"suffx"BKP + 1"
devt = "xxxx"
If type = "93" Then devt = "3390"
If type = "80" Then devt = "3380"
If type = "81" Then devt = "3380"
If type = "82" Then devt = "3380"
Interpret "VOL"suffx".MVS"suffx"BKP = valid addr devt com"
defgdg = df1__suffx"."valid__df2
Interpret "DEF"suffx".MVS"suffx"BKP = defgdg"
End
/* ----- */
/* If any entries have been created for a back-up suite then write */
/* them to the relevant member... */
/* ----- */
Do a = 1 to entries
  suffix = Word(bkups,a) /* Get each suffix */
  Interpret "bkct = MVS"suffx"BKP" /* Get count for this suffix */
  If bkct > 60 Then /* Over 60 in suite is no good*/
    Call TOO_MANY
  Interpret "MVS"suffx".0 = MVS"suffx"BKP"

If bkct <> 0 Then Do /* If any records for suffix: */
  suite = "BCKMVS"suffx
  comnt = ""
  If parm = "B" Then
    comnt = "(FULLPACK JCL)"
  Say Left(">>> Generating "suite", volumes = "bkct,45)__comnt
  "ALLOC FI("suite") DA('BQIBI06.OPSREC.BACKUPS("suite")') SHR"
  "ALLOC FI("gdgdf") DA('BQIBI06.OPSREC.GDGS("suite")') SHR"
  stemid = "VOL"suffx"."
  "EXECIO * DISKW "suite " (Stem "stemid" FINIS"
  stemid = "DEF"suffx"."
  "EXECIO * DISKW "gdgdf " (Stem "stemid" FINIS"
  "FREE FI("suite")"
  "FREE FI("gdgdf")"
  If parm = "B" Then /* Only in batch, and... */
  If gen_jobs = "Y" /* only if flag set */
  Then Do
    x = OUTTRAP("nulls.",20,"NOCONCAT") /* Suppress next msg */
    "DELETE 'BQIBI06.OPSREC.FULLPACK.BCKMVS"suffx'"
    x = OUTTRAP("OFF")
    "ALLOC F(BKJCL) DA('BQIBI06.OPSREC.FULLPACK.BCKMVS"suffx"')
      BLKSIZE(6160) DSORG(PS) UNIT(SYSDA) CYL RECFM(F B) SPACE(2 1)
      LRECL(80) DIR(43) NEW CATALOG"
    "FREE F(BKJCL)"

  Do nn = 1 to bkct /* Create JCL for each volume */
    stemid = "VOL"suffx"."nn

```

```

Interpret "Parse Upper Var "stemid" disvol disadr disdev ."
bkjbnm = "£GOI"suffx"B"Right("Ø"nn,2) /* eg £GOIABØ1! */
suite = "BCKMVS"suffx
usrid = "&USRID"
"ALLOC F(ISPFIL) DA('BQIBIØ6.OPSREC.FULLPACK.BCKMVS"suffx"')
  SHR"
ADDRESS "ISPEXEC"
"FTINCL SOPBK17Ø" /* !!! NOTE this skeleton is */
/* !!! shared with the ISPF */
/* !!! based back-ups system. */

If rc <> Ø Then
  Say ">> Error including skeleton SOPBK17Ø..."
  "FTCLOSE NAME("disvol")"
  ADDRESS "TSO"
  "FREE F(ISPFIL)"
  End /* Do nn = 1 to... */
  End /* If gen_jobs = Y */
  End /* If bkct <> Ø... */
End /* Do a = 1 to... */
/* ----- */
/* A list of ALL of the back-ups is written to member 'COMDBKPS'. */
/* In this list we insert the suite name and the decimal value of */
/* the hex address (this allows us to easily sort into hex order). */
/* This list is a complete list of ALL volumes that are sent to */
/* BKUPSITE, irrespective of whether they are actually restored or */
/* not (those with an 'X' in the BKUPSITE Flag field are not norm- */
/* ally restored). The "FAILMAST" EXEC will later be used to create */
/* a listing of disks required, by MVS system (which excludes those */
/* not normally restored), and a list of ALL disks that we send */
/* backups for. These lists are e-mailed to BKUPSITE so that they */
/* can create the necessary profiles for tests and also have a full */
/* list in case we need to invoke recovery. */
/* ----- */
Do a = 1 to entries
  suffix = Word(bkups,a) /* Get each suffix */
  Interpret "xx = MVS"suffx"BKP" /* Get count for this suffix */
  Interpret "MVS"suffx".Ø = MVS"suffx"BKP"
  If xx <> Ø Then Do w = 1 to xx /* If any records for suffix: */
    Interpret "TEMP = VOL"suffx"."w
    suite = "BCKMVS"suffx
    Parse Var temp vol hexaddr devt comflag .
    If comflag <> "N" Then Do
      sortfld = Right(' 'X2D(hexaddr),6)
      Queue " "vol" "hexaddr" "devt" "suite" "comflag" "sortfld"
    End
  End
End
End
num = Queued()
"ALLOC FI(COMDBKP) DA('BQIBIØ6.OPSREC.BACKUPS(COMDBKPS)') SHR"
"EXECIO "num" DISKW COMDBKP (FINIS"
"FREE FI(COMDBKP)"

```


Return

Skeletons

SOPBK170:

```
)CM *****
)CM * +++                                     +++ *
)CM * +++ BACK-UP VOLUMES FOR OFFSITE RECOVERY...   +++ *
)CM * +++                                     +++ *
)CM *****
//&BKJBNM JOB , '&SUITE BKUP &DISVOL',NOTIFY=&USRID,CLASS=5,
//      MSGLEVEL=(1,1),MSGCLASS=E,REGION=0M
//*
//* BACKUP &DISVOL ( &DISADR )
//*
//&DISVOL EXEC PGM=ADRDUSSU
//SYSPRINT DD SYSOUT=*
//DASD1 DD DISP=SHR,UNIT=&DISDEV,VOL=SER=&DISVOL
//TAPE1 DD DSN=SYS8.&SYS.&SUITE.&DISVOL(+1),
//      DISP=(NEW,CATLG,DELETE),LABEL=EXPDT=990000,
//      UNIT=ROB000,DCB=SYS3.NULL.PATT
//SYSIN DD *
DUMP FULL INDDNAME(DASD1) OUTDDNAME(TAPE1)          WAIT(0,0) -
ADMIN OPTIMIZE(4)
/*
/* DUMP FULL INDDNAME(DASD1) OUTDDNAME(BKUP1) ALLDATA(*) WAIT(0,0) -
//      IF RC GT 0 THEN
//S2 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SE '== JOB &JBNM (SUITE &SUITE &DISVOL) HAS FAILED ==',U(&USRID)
//      ENDIF
//      IF ABEND THEN
//S3 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SE '== JOB &JBNM (SUITE &SUITE &DISVOL) HAS ABENDED ==',U(&USRID)
//      ENDIF
```

SOPR170:

```
//&USRID.I JOB , 'INIT - &OPCU',CLASS=A,MSGCLASS=E,
//      MSGLEVEL=(1,1),NOTIFY=&USRID
//* ----- ***
//* INITIALIZE THE VOLUME. MUST BE OFFLINE FIRST ***
//* ----- ***
//VARY1 EXEC PGM=MVSCMD,PARM='V &OPCU,OFFLINE'
//WAIT EXEC PGM=WAIT,PARM='2'
//INIT1 EXEC PGM=ICKDSF,REGION=1024K
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
```

```

INIT UNITADDRESS(&OPCU) -
    VTOC(0,1,29) -
    INDEX(2,0,30) -
    OWNERID(SITA) -
    VOLID(&OPVL) -
    VFY(&OPCRVL) -
    PURGE
//          IF RC GT 0 THEN
//S2       EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
SE '== JOB &USRID.I (INIT. &OPCU AS &OPVL) HAS FAILED. ==' ,U(&USRID)
//          ENDIF
//          IF ABEND THEN
//S3       EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
SE '== JOB &USRID.I (INIT. &OPCU AS &OPVL) HAS ABENDED. ==' ,U(&USRID)
//          ENDIF

```

SOPSR171

```

//&USRID.S JOB , 'REGEN BACKUPS', CLASS=A, MSGCLASS=E, REGION=6M,
//      MSGLEVEL=(1,1), NOTIFY=&USRID
//*-----***
//* COMPRESS TARGET LIBRARIES FIRST ***
//*-----***
//COMPRESS EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//IN1      DD DISP=SHR, DSN=BQIBI06.OPSREC.BACKUPS
//OU1      DD DISP=SHR, DSN=BQIBI06.OPSREC.BACKUPS
//IN2      DD DISP=SHR, DSN=BQIBI06.OPSREC.GDGS
//OU2      DD DISP=SHR, DSN=BQIBI06.OPSREC.GDGS
//SYSIN    DD *
COPY INDD=((IN1,R)), OUTDD=OU1
COPY INDD=((IN2,R)), OUTDD=OU2
/*
//* ----- ***
//* RE-GENERATE THE BACK-UP SUITES ***
//* ----- ***
//RUNREXX EXEC PGM=IKJEFT01, DYNAMNBR=200
//SYSTSPRT DD SYSOUT=*
//SYSEXEC DD DSN=BQIBI06.OPSREC.REXX, DISP=SHR
//ISPLOG DD SYSOUT=*, DCB=(RECFM=VA, LRECL=125, BLKSIZE=129)
//ISPPROF DD DSN=&&PROF, DISP=(NEW,PASS), SPACE=(TRK,(1,1,10)),
//          DCB=(RECFM=FB, LRECL=80, BLKSIZE=3120), UNIT=SYSDA
//ISPMLIB DD DISP=SHR, DSN=SYS1.MLIB
//          DD DISP=SHR, DSN=SYS3.MLIB
//ISPLIB DD DISP=SHR, DSN=SYS3.PLIB

```

```

//ISPTLIB DD DISP=SHR,DSN=SYS1.TLIB
//ISPSLIB DD DSN=BQIBI06.OPSREC.SKELS,DISP=SHR
//SYSTSIN DD *
ISPSTART CMD(%GENBKUPS B)
// IF RC GT 0 THEN
//S2 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SE '== JOB &USRID.S GAVE A NON-ZERO RC. PLEASE CHECK. ==',U(&USRID)
// ENDIF
// IF ABEND THEN
//S3 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SE '== JOB &USRID.S HAS ABENDED... PLEASE CHECK. ==',U(&USRID)
// ENDIF

```

SOPSR172:

```

)CM *****
)CM * +++ GENERATE A JOB TO DEFINE THE REQUESTED GDG BASES FOR +++ *
)CM * +++ A BACKUP SUITE. NOTE THAT *ALL* GDGS WILL BE DEFINED, +++ *
)CM * +++ SO THERE WILL NORMALLY BE A NON-ZERO RC FROM THIS JOB. +++ *
)CM * +++ THE ')IM' STATEMENT WILL PULL IN THE DEFINITIONS FROM +++ *
)CM * +++ THE DATASET 'BQIBI06.OPSREC.GDGS'... +++ *
)CM *****
//&USRID.G JOB , 'DEFINE &SUITENAM GDGS', CLASS=A, MSGCLASS=E,
// MSGLEVEL=(1,1), NOTIFY=&USRID, REGION=4M
//* ----- ***
//* DEFINE GDG BASE(S) FOR BACKUP SUITE &SUITENAM ***
//* ----- ***
//DEFINE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
)IM &SUITENAM
// IF ABEND THEN
//S2 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SE '== JOB &USRID.G HAS ABENDED. PLEASE CHECK. ==',U(&USRID)
// ENDIF

```

DISK LISTING

The 'DISKMAST' member is a duplication of the following, incorporating multiple versions of addresses, as required.

1 DISK USAGE - @M@ 2001

DISK IDENTIFICATION

PUT DOCUMENTATION AND NAMING STANDARDS HERE...

* BEFORE VOLID INDICATES SHARED ONLINE ADDR IN HCD

1 SVA-1 CONFIGURATION (ALL 339Ø-3) CLUSTER Ø
+ SVA-1 CONFIGURATION (ALL 339Ø-3) CLUSTER Ø

Table with columns for CPU-Ø1, CPU-Ø2, CPU-Ø3, CPU-Ø4 and rows for CHP, cache, NVS, and ID. Includes entries like 95ØØ-Ø, 6ØØØ @6ØØØ, etc.

OTHER

Various elements of the system that require explanation:

- WHEREAMI – a simple program that checks the SMFid and generates a return code so that we can check where we are running.
• MVSCMD – a simple program to issue MVS commands.

Grant Carson
Systems Programmer (UK)

© Xephon 2002

MVS news

Amdahl's IT Services unit has announced immediate availability of the Integrated Facility for Linux (IFL) on its Millennium 2000C and 2000E S/390 and z/OS-compatible CMOS servers, and the availability of native FICON in Q1 2002.

IFL, which allows engines to be used exclusively for Linux, VIF and z/VM 4.1 or above for Linux, will also be available on OmniFlex and Millennium 700 and 2000A server.

For further information contact:
Amdahl IT Services, 1250 East Arques Avenue, Sunnyvale, CA 94088, USA.
Tel: (408) 746 6000.
URL: <http://www.amdahlitsservices.com>.

* * *

Serena Software is shipping its ChangeMan ZDD Desktop Development for z/OS and OS/390, which is designed to simplify cross-platform development by eliminating the need for manual file transfers, as well as delivering automated software change management to mainframe applications developed in desktop environments.

The product integrates with ChangeMan ZMF, Application Change Manager for Mainframe Systems, which means software assets remain in the protected environment of ChangeMan ZMF to enable sites to free IT resources dedicated to transfer requirements and help ensure the integrity of SCM processes.

Representing mainframe data and files as a drive letter with hierarchical folders-viewable within Windows Explorer, it's geared to improving the efficiency of the development process by enabling desktop

developers to access mainframe code from their favourite IDE, and to make modifications while utilizing the process management, version control, impact analysis, concurrent development, audit trails, automatic notification, and other features of ChangeMan ZMF.

For further information contact:
Serena Software, 500 Airport Blvd, 2nd Floor, Burlingame, CA 94010-1904, USA.
URL: <http://www.serena.com>.

* * *

Time Machine Software has announced Version 3.4.0 of its SmartProduction performance enhancement tool and its DB2 optional feature, now with over 30 new cases added to the knowledge base, allowing users to detect even more inefficiencies during system analysis.

Specifically, Version 3.4.0 selects all jobs for specified job name prefixes, allowing users to see all matching jobs that require modification. It also provides the ability to limit the view to cases of operations inefficiency.

What's more, it provides the ability to analyse a TSO session and provides for the specification of SYSOUT destinations with the PRINTOFF command in Panel J and Panel D. Cases can be routed to the programmers responsible for correcting the problems detected in their jobs.

For further information contact:
Axios Products, 1373-10 Veterans Highway, Hauppauge, NY 11788, USA.
Tel: (631) 979 0100.
URL: <http://www.tmachine.com/smart.htm>.



xephon