

LPREXEC

```

/* REXX */
/*****
/*                                PANEL PANEL PANEL PANEL          */
/* MRKTOOLS- THIS VERSION RUN USING PANEL PANEL PANEL PANEL      */
/* MRKEXECS-                                PANEL PANEL PANEL PANEL */
/*****

```

```

"FREE DDNAME(INFILE)"
"FREE DDNAME(REPTFILE)"
address ispec 'vget (lpropt, dname, cmd, cln, apln, pdsn, saveid,
                  redef, level, acr, dname) shared'
address ispec 'vget (db2ss, db2l, untd, catid, swwid, lprid, lprn,
                  tsoid, jname) shared'

```

```

OPTION = lpropt
CALL MAINPROCESS
RETURN

```

```

MAINPROCESS:
/* This routine the execution to the required paragraph based on the
option selected on the MAIN MENU */

```

```

select
  when OPTION = 1 then
    do
      DSNKEY = 'TORENAME'
      call ALLOCFILE
      call RENAMEBYLIST
    end
  when OPTION = 2 then
    do
      DSNKEY = 'TODELETE'
      call DELPDSMEMBERS
    end
  when OPTION = 3 then
    do
      DSNKEY = 'KNOWATTR'
      call ALLOCFILE
      call KNOWATTR          /*identifies just the attributes*/
    end
  when OPTION = 4 then
    do
      DSNKEY = 'KNOWVSAM'
      call ALLOCFILE
      call KNOWVSAM
    end
  when OPTION = 5 then;      call RENAMEBYLEVEL
  when OPTION = 6 then
    do
      DSNKEY = 'LSTBYLVL'
      call ALLOCFILE
      call LISTBYLEVEL
    end
  when OPTION = 7 then
    do
      DSNKEY = 'KNOWATTR'
      EMPTY_YES_NO = 'Y'
      call ALLOCFILE
      call KNOWATTR          /*this also identifies empty */
    end
  when OPTION = 8 then
    do
      PASSPARM = 'MURALI'
      say ' BEFORE CALLING TESTCALL ' PASSPARM
      call TESTCALL PASSPARM
      say ' AFTER return FROM TESTCALL ' PASSPARM
      pull
    end
end

```

LPREXEC5

```

when OPTION = 09 then
  do
    say 'BEFORE CALLING FRPR'
    pull
    call FRPR
    if RC =0 then
      say 'AFTER CALLING FRPR-SUCCESS'
    ELSE
      say 'AFTER CALLING FRPR-FAILURE'
    pull
  end
when OPTION = 10 then
  do
    say 'BEFORE CALLING TODV1'
    call TODV1
    if RC =0 then
      say 'AFTER CALLING TODV1-SUCCESS'
    ELSE
      say 'AFTER CALLING TODV1-FAILURE'
    pull
  end
when OPTION = 11 then;      call DB2REXX
when OPTION = 12 then;      call HLIST_LEVEL
when OPTION = 13 then
  do
    DSNKEY = 'KNOWMIGR'
    call ALLOCFILE
    call HLIST_DSN
  end
when OPTION = 14 then;      call HEXADECI MAL_TO_DECI MAL
when OPTION = 15 then
  do
    call ALLOCFILE
    call DB2DETAILS
  end
when OPTION = 16 then
  do
    DSNKEY = 'BINDJCL'
    call ALLOCFILE
    call BINDJCL
  end
when OPTION = 17 then
  do
    call LOADEXEC
  end
when OPTION = 18 then
  do
    call DBCOMPARE
  end
when OPTION = 19 then
  do
    DSNKEY = 'DLTLST'
    call ALLOCFILE
    call DELETEBYLIST
  end
when OPTION = 21 then
  do
    DSNKEY = 'RMMCOPY'
    call ALLOCFILE
    call RMMCOPY
  end
when OPTION = 32 then;
  do
    call BTS0 cmd
  end
when OPTION = 33 then;      call SCANPDS
when OPTION = 34 then;      call
when OPTION = 35 then;      call DISPLAY_INFORMATION

```

```

                                LPREXEC
when OPTION = 36 then;         call DISPLAY_LINKLIST
when OPTION = 37 then;         call DISPLAY_APFLIST
when OPTION = 38 then;         call REAL_NAME
when OPTION = 39 then
    call PROFDUMP dname
when OPTION = 40 then;         call SHOWMVS_PARA
WHEN OPTION = XX then
do
    zedsmg = 'Enter a valid value'
    address ISPEXEC "SETMSG MSG(ISRZ001)"
end
end
return

ALLOCFILE:
if TRACE_ON = 'Y' then TRACE ?R
else TRACE 0
/* This routine allocates the user INPUT file and copies into the
file in the standard name MRK7.LPRT00LS."DSNKEY".FILE.DTPSF . This
also allocates a output report file to check the results and erros
in the name MRK7.LPRT00LS."DSNKEY".REPORT.DTPSF */

/***** change the input file here *****/
if (OPTION \= 6 & OPTION \= 15 & OPTION \=16 & OPTION \=17) then
do
    FILE_NAME = dname

/* ALLOCATING THE ENTERED INPUT INFILE */
"ALLOCATE DDNAME(INFILE)
DSNAME(' FILE_NAME' ) OLD"
if RC \= 0 then
    say 'ALLOCATE ERROR ON ENTERED INPUT DATASET ' FILE_NAME
else
do
    "EXECIO * DISKR INFILE (FINIS STEM LCLIST."
    if RC \= 0 then
do
    say 'ERROR READING "FILE_NAME"' RC = "RC"
EXIT
end
end
end

/* ALLOCATING THE OUTPUT REPORT FILE */

if OPTION \= 12 & OPTION \= 16 & OPTION \=21 then /* not required */
"ALLOCATE DDNAME(REPTFILE)
DSNAME(' MRK7. "TSOID".LPRT00LS."DSNKEY".REPORT.DTPSF') NEW
SPACE(1 1) TRACKS RECFM(F,B) LRECL(133) DSORG(PS)"
else /* for BINDJCL in DTD prefix */
if OPTION = 16 then
"ALLOCATE DDNAME(REPTFILE)
DSNAME(' DTD. MRK7. "TSOID".LPRT00LS."DSNKEY".JCL.DTPSF') MOD
UNIT(TEMPDA) SPACE(1 1) TRACKS RECFM(F,B) LRECL(80) DSORG(PS)"

if RC \= 0 then
do
    say 'ALLOCATE ERROR ON OUTPUT DATASET'
    call EXIT
end

return

RENAMEBYLIST:
/*****
/* RENAMES THE DATASET IN THE LIST PROVIDED IN THE INPUT DATASET */

```

LPREXEC5

```

/*****
do I = 1 TO LCLIST.0
  CURRENTDSN = WORD(LCLIST. I, 1)

/*  FIRST_PART is IF QUALIFIER IS ADDED IN THE BEGGINING */
/*  MID_PART   is IF QUALIFIER IS ADDED IN THE MIDDLE   */
/*  END_PART   is IF QUALIFIER IS ADDED IN THE end      */
/*  FINAL_PART is FINAL DSN AFTER CONCATENATING THESE THREE */
/*  SUBSTR(CURRENTDSN, STARTPOS, NO OF CHARACTERS)      */

FIRST_PART = 'CSS8.V7F.LPRTTOOLS.'
MID_PART   = ''
END_PART   = SUBSTR(CURRENTDSN, 10, 11)

FINAL_PART = FIRST_PART || MID_PART || END_PART

DUMMY = OUTTRAP("LCT.", "*")
"RENAME 'CURRENTDSN' 'FINAL_PART'"

if RC \= 0 then
  do
    REPT_LINE = ' ERROR RENAMING ' || CURRENTDSN
    push REPT_LINE
    "EXECIO 1 DISKW REPTFILE"
  end
end
DUMMY = OUTTRAP(OFF)
"FREE DDNAME(INFILE)"
"FREE DDNAME(REPTFILE)"
return

DELPDSMEMBERS:
/*****
/* DELETE THE MEMBERS INSIDE THE INPUT PDS */
/*****

TO_BE_DELETED = pdsname
if LEFT(TO_BE_DELETED, 9) \= 'CSS8.V7F.' |,
  RIGHT(TO_BE_DELETED, 9) \= 'BACKUP.DTPSF' then
  do
    say "CHANGE THE PROGRAM IF YOU WANT TO DELETE OTHER THAN",
      "CSS8.V7F OR BACKUP.DTPSF."
    say "PRESS ANY KEY TO CONTINUE"
    pull
/*  return */
  end

say "THE DSN YOU ENTERED IS : " TO_BE_DELETED "PLEASE CONFIRM Y/N"
pull CONFIRM
UPPER CONFIRM
if CONFIRM = 'Y' then
  do
    DUMMY_VAR = OUTTRAP("D_S_N.", "*", "CONCAT")
    "LISTDS 'TO_BE_DELETED' MEMBERS"
    if RC \= 0 then
      do
        say ' ERROR LISTING THE DATASETS: ' RC
        EXIT
      end
    ELSE
      READ_LINES = D_S_N.0
      DUMMY_VAR = OUTTRAP("OFF")
      say "READ LINES" READ_LINES

    if WORD(D_S_N. 3, 4) = 'PO' then
      do I = 7 TO READ_LINES
        PDS_MEMBER = TO_BE_DELETED || '(' || STRIP(VALUE(D_S_N. "I")) || ')'

```

```

                                LPREXECs
        "DELETE ' "PDS_MEMBER" "
    end I
ELSE
    say ""TO_BE_DELETED" IS NOT A PDS"
end
ELSE
    NOP
return

DBCMPARE:
/* REXX */
ADDRESS ISPEXEC
"ISREDIT MACRO"
ADDRESS
ADDRESS ISPEXEC
"LIBDEF ISPLIB DATASET ID(' CSS8.V7F. "LPRN". ISPLIB' )"
"LIBDEF ISPSLIB DATASET ID(' CSS8.V7F. "LPRN". ISPSLIB' )"
VGET ZDATE
VGET ZTIME
"ISREDIT (XDSN)=DATASET"
"ISREDIT (XMEM)=MEMBER"
/* Iprn      = MVSVAR(' SYSNAME' ) */
' FTOPEM TEMP'
' FTINCL DBCOMPAR'          /* printskl is available in CSS8.V7F.DV.ISPSLIB' */
' FTCLOSE'
ADDRESS ISPEXEC "EDIT DATASET(' &ZTEMPF' )"
RETURN 0

DELETEBYLIST:
/*****
/* DELETES DATASETS FROM A LIST PROVIDED IN THE INPUT DATASET */
*****/
CONFIRM = ' '
DUMMY = OUTTRAP("LCT. ", "")
do I = 1 TO LCLIST.0
    CURRENTDSN = WORD(LCLIST. I, 1)

    if CONFIRM = 'B' then          /* CONFIRMATION BYPASSED */
        NOP
    else
        do
            SAY 'DATA SET IS ; PRESS Y TO CONFIRM B TO BYPASS ' CURRENTDSN
            PULL CONFIRM
            UPPER CONFIRM
        end

        IF CONFIRM = 'Y' | CONFIRM = 'B' then
            do
                "DELETE ' "CURRENTDSN" "

                if RC \= 0 then
                    do
                        REPT_LINE = ' ERROR DELETING ' || CURRENTDSN
                        push REPT_LINE
                        "EXECIO 1 DISKW REPTFILE"
                    end
                end
            end
        end
    DUMMY = OUTTRAP(OFF)
    "FREE DDNAME(INFILE)"
    "FREE DDNAME(REPTFILE)"
return
KNOWVSAM:
/*****
/*
/* TO KNOW THE VSAM FILES IN THE GIVEN DATASET list
/* FOR VSAM FILES SYSDSORG WILL BE VS
*****/

```

LPREXEC5

```

/*****
FL = 'DATASET NAME          DSORG          '
     ' STATUS              RECORDS          '
push FL
"EXECIO 1 DISKW REPTFILE"
FL = '-----'
push FL
"EXECIO 1 DISKW REPTFILE"
do I = 1 TO LCLIST.0
  DSN_STATUS = ''
  VOLNAME    = ' '
  DSORG      = ' '
  LRC = 20
  CURRENTDSN = WORD(LCLIST.I,1)
  LRC = LISTDSI(''CURRENTDSN'' PREALLOC DIRECTORY RECALL)
  if LRC <= 4 then
  do
    NO_RECORDS = ''
    if SYSDSORG = 'VS' then
    do
      DUMMY = OUTTRAP("LCT.", "*")
      "LISTCAT ENTRIES('CURRENTDSN') ALL"
      DUMMY = OUTTRAP(OFF)
      if WORD(LCT.1,1) = 'AIX' then DSN_STATUS = 'ALTERNATE INDEX'
      else call KNOWVSAM_EMPTY
    end
    PADDSN = INSERT(' ', CURRENTDSN, 48)
    PADDSORG = INSERT(' ', SYSDSORG, 5)
    PADSTATUS = INSERT(' ', DSN_STATUS, 20)
    push PADDSN PADDSORG PADSTATUS NO_RECORDS
    "EXECIO 1 DISKW REPTFILE"
  end
  ELSE
  if LRC \> 12 then
  do
    DUMMY = OUTTRAP("LCT.", "*")
    "LISTCAT ENTRIES('CURRENTDSN') "
    DUMMY = OUTTRAP(OFF)
    BASE = WORDPOS('GDG BASE', LCT.1)
    if BASE = 1 then
    do
      PADDSN = INSERT(' ', CURRENTDSN, 48)
      PADDSORG = 'GDG'
      push PADDSN PADDSORG
      "EXECIO 1 DISKW REPTFILE"
    end
  end
  else
  do
    PADDSN = INSERT(' ', CURRENTDSN, 48)
    PADDSORG = 'ERROR - CHECK WHETHER DATASET IS AVAILBLE '
    push PADDSN PADDSORG
    "EXECIO 1 DISKW REPTFILE"
  end
end
"EXECIO 0 DISKW REPTFILE ( FINIS"
"FREE DDNAME(INFILE)"
"FREE DDNAME(REPTFILE)"
return

```

KNOWVSAM\_EMPTY:

```

/* below routine is to identify the empty VSAM datasets */
/* IR_LOCN is to assign IMBED and REPLICATE location */
/* RECS_LOCN is to assign NO OF RECORDS location */
/* LASTPOSTN is to identify the last position of '-' and add 1 */

```

```

if SUBSTR(WORD(LCT.32,1),1,8) = 'SHROPTNS' then

```

LPREXECS

```

do
/* if only data component is present */
IR_LOCN      = LCT. 32
RECS_LOCN    = LCT. 35
end
else
if SUBSTR(WORD(LCT. 33, 1), 1, 8) = 'SHROPTNS' then
do
/* if data and index component are present */
IR_LOCN      = LCT. 33
RECS_LOCN    = LCT. 36
end
else
do
/* if data, index and alterante index components are present */
IR_LOCN      = LCT. 34
RECS_LOCN    = LCT. 37
end

DSN_STATUS    = WORD(IR_LOCN, 7) || ' ' || WORD(IR_LOCN, 8)
LASTPOSTN     = LASTPOS('-', WORD(RECS_LOCN, 1)) + 1
NO_RECORDS    = SUBSTR(WORD(RECS_LOCN, 1), LASTPOSTN)

if NO_RECORDS = 0 then DSN_STATUS = DSN_STATUS || ' ' || 'EMPTY'
else                  DSN_STATUS = DSN_STATUS || ' ' || NO_RECORDS

return

RENAMEBYLEVEL:
/*****
/* TO RENAME THE DATASET BY GIVING THE LEVEL */
*****/

TO_BE_RENAMED = level
DUMMY_VAR = OUTTRAP("D_S_N.", "*", "CONCAT")
"LISTCAT LEVEL('TO_BE_RENAMED'*)"
if RC \= 0 then
do
say 'ERROR LISTING THE DATASETS'
EXIT
end
else
READ_LINES = D_S_N. 0
DUMMY_VAR = OUTTRAP("OFF")

say 'READ LINES' READ_LINES
do I = 1 TO READ_LINES BY 2
DSN_TYPE = WORD(VALUE(D_S_N. ""I""), 1)
DSN_HYPHEN = WORD(VALUE(D_S_N. ""I""), 2)
DSN_NAME = WORD(VALUE(D_S_N. ""I""), 3)
if DSN_TYPE = 'NONVSAM' then
do
"RENAME '"CURRENTDSN"' '"FINAL_PART"' "
say 'DSN TYPE : ' DSN_TYPE
say 'DSN NAME : ' DSN_NAME
end
else
NOP
end
return

LISTBYLEVEL:
/*****
/* TO LIST THE DATASET AND TYPE */
*****/

ENTERED_LEVEL = level
FL = 'DATASET NAME' TYPE '

```

LPREXEC5

```

push FL
"EXECIO 1 DISKW REPTFILE"
FL = '-----'
push FL
"EXECIO 1 DISKW REPTFILE"
DUMMY_VAR = OUTTRAP("D_S_N.", "*", "CONCAT")
"LISTCAT LEVEL("ENTERED_LEVEL")"
if RC \= 0 then
do
say 'ERROR LISTING THE DATASETS'
EXIT
end
ELSE
do
READ_LINES = D_S_N.0
end
DUMMY_VAR = OUTTRAP("OFF")

say "NO OF DATASETS IN THIS LEVEL" READ_LINES/2
DO I = 1 TO READ_LINES BY 2
DSN_TYPE = WORD(VALUE(D_S_N. ""I""), 1)
if DSN_TYPE \= 'GDG' then
do
DSN_HYPHEN = WORD(VALUE(D_S_N. ""I""), 2)
DSN_NAME = WORD(VALUE(D_S_N. ""I""), 3)
end
ELSE
do
DSN_HYPHEN = WORD(VALUE(D_S_N. ""I""), 3)
DSN_NAME = WORD(VALUE(D_S_N. ""I""), 4)
end
/* say 'DSN NAME : ' INSERT(' ', DSN_NAME, 48) " TYPE " DSN_TYPE */
FL = INSERT(' ', DSN_NAME, 48) || DSN_TYPE
push FL
"EXECIO 1 DISKW REPTFILE"
end
return

```

WRITEROUTINE:

```

if f :NT_CTR > 64
FL1 = 'DATASET NAME'
FL2 = 'LUME STATUS'
FL3 = 'KEY'
FL = FL1 || FL2 || FL3
push FL
"EXECIO 1 DISKW REPTFILE"
FL1 = '-----'
FL2 = '-----'
FL3 = '-----'
FL = FL1 || FL2 || FL3
push FL
"EXECIO 1 DISKW REPTFILE"
return
KNOWATTR:
/*****
/*
/* TO KNOW THE ATTRIBUTES OF THE DATASETS IN THE LIST
/* DSNAME + DATASET ORGANIZATION + VOLUME + STATUS + RECORD FORMAT +
/* LRECL + BLKSIZE + KEY LENGTH
*****/

FL1 = 'DATASET NAME'
FL2 = 'LUME STATUS'
FL3 = 'KEY'
FL = FL1 || FL2 || FL3
push FL
"EXECIO 1 DISKW REPTFILE"

```

LPREXEC5

```

FL1 = '-----'
FL2 = '-----'
FL3 = '-----'
FL = FL1 || FL2 || FL3
push FL
"EXECIO 1 DISKW REPTFILE"

```

```

DO I = 1 TO LCLIST.0
  DSN_STATUS = ''
  VOLNAME     = ''
  DSORG       = ''
  RECFM       = ''
  LRECL       = ''
  BLKSIZE     = ''
  KEYLEN      = ''
  LRC=20
  CURRENTDSN = WORD(LCLIST.I, 1)
  call DSNIDENT

  if OPTION = '7' then          call EMPTYIDENT

  /* if we are interested just ATTRIBUTE identification */
  PADDSN      = INSERT(' ', CURRENTDSN, 48)
  PADDSORG    = INSERT(' ', DSORG, 7)
  PADVOLUME   = INSERT(' ', VOLNAME, 8)
  PADSTATUS   = INSERT(' ', DSN_STATUS, 30)
  PADRECFM    = INSERT(' ', RECFM, 5)
  PADLRECL    = INSERT(' ', LRECL, 5)
  PADBLKSZ    = INSERT(' ', BLKSIZE, 7)
  PADKEYLN    = INSERT(' ', KEYLEN, 5)
  push PADDSN PADDSORG PADVOLUME PADSTATUS PADRECFM PADLRECL,
        PADBLKSZ PADKEYLN
  "EXECIO 1 DISKW REPTFILE"
end
"FREE DDNAME(INFILE)"
"FREE DDNAME(REPTFILE)"
return

```

EMPTYIDENT:

```

if EMPTY_YES_NO = 'Y' then
do
  if LRC = 0 then
  do
    DUMMY = OUTTRAP("DUMMY_TRACE.", "*")
    "ALLOCATE DDNAME(TEMPFILE)"
    DSNNAME(' "CURRENTDSN" ') SHR"
    if RC \= 0 then say ' ALLOCATION FAILED FOR ' CURRENTDSN
    "EXECIO * DISKR TEMPFIL (FINISSTEM LTEMP."
    DUMMY = OUTTRAP("OFF")

    if RC \= 0 then      DSN_STATUS = 'READ FAILED -INVALID FORMAT'
    else
      if LTEMP.0 = 0 then DSN_STATUS = 'EMPTY'
      else                DSN_STATUS = 'NOT EMPTY ' || LTEMP.0

    "FREE DDNAME(TEMPFILE)"
  end
end
return

```

PDSMEMBER:

```

SAY "INSIDE PDSMEMBER"
MEMNAME = (CURRENTDSN, '(')
DUMMY = OUTTRAP("LCT.", "*")
"LISTDS (' "CURRENTDSN" ') MEMBERS
DUMMY = OUTTRAP(OFF)

```

LPREXECS

```

do i = 5 to LCT.1
  STACK_MEM = ' (' LCT."I" ' )'
  say 'stack_mem ' STACK_MEM
  if STACK_MEM = MEMNAME then
    do
      PADDSN = INSERT(' ', CURRENTDSN, 48)
      push PADDSN " MEMBER FOUND IN PDS"
      "EXECIO 1 DISKW REPTFILE"
    end
  end
end
return

DSNIDENT:
/* to know whether the listed dataset is GDG or not */
/* this identifies whter the dataset is in TAPE or not */
/* assumes, 1st position of VOLUME is 0, it is tape, may need to do */
/* modification if VOLUME are going to be different */
/* picks the latest version in the GDG entries and migrates only that*/

DUMMY = OUTTRAP("LCT.", "*")
"LISTCAT ENTRIES('"CURRENTDSN"') "
DUMMY = OUTTRAP(OFF)
BASE = WORDPOS(' GDG BASE' , LCT.1)

if BASE = 0 then /* true - not equal to GDG */
do
  call TAPEIDENT
  if DSN_STATUS \= "TAPE ENTRY" then
  do
    /*LRC = LISTDSI (" "CURRENTDSN" " PREALLOC DIRECTORY RECALL)*/
    LRC = LISTDSI (" "CURRENTDSN" " DIRECTORY NORECALL)
    if LRC = 16 then
    do
      DSN_STATUS = "DATASET NOT FOUND-A"
    end
    else
    if LRC = 9 then DSN_STATUS = "MIGRATED DATASET-A"
    else
    do
      VOLNAME = SYSVOLUME
      DSORG = SYSDSORG
      RECFM = SYSRECFM
      LRECL = SYSLRECL
      BLKSIZE = SYSBLKSIZE
      KEYLEN = SYSKEYLEN

      if DSORG = "PO" then DSN_STATUS = "PDS"
      else
      if DSORG = 'VS' then
      do
        DSN_STATUS = "VSAM"
        DUMMY = OUTTRAP("LCT.", "*")
        "LISTCAT ENTRIES('"CURRENTDSN"') ALL"
        DUMMY = OUTTRAP(OFF)
        if WORD(LCT.1,1) = 'AIX' then DSN_STATUS = 'ALTERNATE INDEX'
        else call KNOWVSAM_EMPTY
      end
      else
      if DSORG = "PS" then DSN_STATUS = "SEQ"
      else DSN_STATUS = "?????"
      end
    end
  end
end
else
if LCT.1 = 0 then DSN_STATUS = "DATASET NOT EXISTS"
else
do
/* gdg */

```

```

                                LPREXECS
Q = LCT.0 - 1                    /* total no of lines - 1 */
if Q = 1 then    DSN_STATUS = "GDG BASE - NO ENTRIES"
else
do
  MODDSN = WORD(LCT.0, 3)        /* picks the last gdg entry */
  CURRENTDSN = MODDSN
  call TAPEIDENT
  if DSN_STATUS \= "TAPE ENTRY" then
  do
    /*LRC = LISTDSI (" "MODDSN" " PREALLOC DIRECTORY RECALL)*/
    LRC = LISTDSI (" "MODDSN" " DIRECTORY NORECALL)
    if LRC = 16 then
    do
      DSN_STATUS = "DATASET NOT FOUND-B"
    end
    else
    if LRC = 9 then DSN_STATUS = "MIGRATED DATASET-B"
    else
    if LRC = 0 then
    do
      VOLNAME      = SYSVOLUME
      DSORG        = SYSDSORG
      DSN_STATUS   = 'GDG ENTRY'
    end
  end
end
end
return

```

TAPEIDENT:

```

DUMMY = OUTTRAP("TAPE_IDENT.", "*")
"LISTCAT ENTRIES('CURRENTDSN') ALL"
DUMMY = OUTTRAP(OFF)
REC_CTR = 0

/* below loop is executed to read the output lines */
/* till we hit the VOLUMES line */
do FOREVER UNTIL REC_CTR >= TAPE_IDENT.0
  REC_CTR = REC_CTR + 1
  if TAPE_IDENT.REC_CTR = 'VOLUMES' then
  do
    VOL_PTR      = rec_ctr + 1
    REC_CTR      = TAPE_IDENT.0        /* to come out of the loop */
    TAPEENTRY    = SUBSTR(WORD(TAPE_IDENT.VOL_PTR, 1), 19, 1)
    VOLNAME      = SUBSTR(WORD(TAPE_IDENT.VOL_PTR, 1), 19, 8)
  end
  else
  do
    VOLNAME      = '-----'
    TAPEENTRY    = '-----'
  end
end
end

```

```

/* assuming if first character of volume is 0, it is a tape */
if TAPEENTRY = 0 then DSN_STATUS = "TAPE ENTRY"

return

```

DB2REXX :

```

/* To test the DB2 connectivity thru REXX */

```

```

NODE_NAME = sysvar(SYSNODE)
OWNER     = 'SYSIBM'

```

LPREXECS

```
Q= " SELECT A. NAME           "
    "      , A. CREATOR      "
    "      , A. TYPE         "
    "      , A. DBNAME       "
    "      , A. TSNAME       "
    " FROM "OWNER".SYSTABLES A "
    " WHERE A. CREATOR       = 'SYSIBM' "
```

```
call CALLDB2 Q
```

```
say 'NAME           CREATOR   TYPE      DBNAME      TSNAME ' ,
say '-----'
```

```
do row = 1 to rt.0
  NAME           = SUBSTR(RT. 1. row, 1, 25, ' ')
  CREATOR        = SUBSTR(RT. 2. row, 1, 10, ' ')
  TYPE           = SUBSTR(RT. 3. row, 1, 07, ' ')
  DBNAME         = SUBSTR(RT. 4. row, 1, 12, ' ')
  TSNAME         = RT. 5. row
```

```
LN = NAME CREATOR TYPE DBNAME TSNAME
say LN
```

```
end
EXIT 0
```

```
DB2DETAILS:
/* To get the report on DB2 databases and tablespaces and programs */
```

```
APPN_SYSTEM = 'RFS'
NODE_NAME   = sysvar(SYSNODE)
OWNER       = 'SYSIBM'
```

```
Q= " SELECT A. NAME           "
    "      , A. CREATOR      "
    "      , A. TYPE         "
    "      , A. DBNAME       "
    "      , A. TSNAME       "
    " FROM "OWNER".SYSTABLES A "
    " WHERE A. DBNAME       LIKE '"APPN_SYSTEM"' "
    " AND   A. DBNAME       = PARMDBNAME "
```

```
call CALLDB2 Q
```

```
say 'DBNAME   TSNAME   TYPE      TABLE NAME           ' ,
say '-----'
```

```
DBPREVNAME = ' '
TSPREVNAME = ' '
do row = 1 to rt.0
  NAME           = SUBSTR(RT. 1. row, 1, 25, ' ')
  CREATOR        = SUBSTR(RT. 2. row, 1, 10, ' ')
  TYPE           = SUBSTR(RT. 3. row, 1, 07, ' ')
  DBNAME         = SUBSTR(RT. 4. row, 1, 12, ' ')
  TSNAME         = RT. 5. row
```

```
if DBPREVNAME = DBNAME then
do
  if TSPREVNAME = TSNAME then
do
  if REPT_FIRST_TIME = 'T'
    LN = "-----"
    push LN
    "EXECIO 1 DISKW REPTFILE"
    LN = DBNAME TSNAME TYPE NAME
    push LN
    "EXECIO 1 DISKW REPTFILE"
```

```

                                LPREXECs
    el se
        LN = '                ' TYPE NAME
        push LN
        "EXECIO 1 DISKW REPTFILE"
    end
    el se
    do
        TSPREVNAME = TSNAME
        LN = '                ' TSNAME TYPE NAME
        push LN
        "EXECIO 1 DISKW REPTFILE"
    end
    el se
    do
        DBPREVNAME          = DBNAME
        TSPREVNAME          = TSNAME
        LN = '-----'
        push LN
        "EXECIO 1 DISKW REPTFILE"
        LN = DBNAME TSNAME TYPE NAME
        push LN
        "EXECIO 1 DISKW REPTFILE"
    end
end
EXIT 0

CALLDB2:
/* Used inside DB2REXX paragraph */

arg SQLQUERY

if sysvar(SYSENV)='FORE' then "ALLOC DD(SYSOUT) DSN(*) SHR REUSE"
rt. = ''
call rxdb2 db2ss, SQLQUERY, 'RT.'
CALLDB2_RC=rc

if sysvar(SYSENV)='FORE' then "FREE DD(SYSOUT)"

if CALLDB2_RC > 4 then
do
    say ' * * * Call error ' CALLDB2_RC
    exit 20
end

if sql ca. sql code = 100 then
do
    say ' * * * No rows.'
    say ' enter Y If you want to exit'
    pull no_row_exit
    UPPER no_row_exit
    if no_row_exit = 'N' then
        return CALLDB2RC
    else
        exit 100
end

if sql ca. sql code \= 0 then
do
    say ' * * * sql error ' sql ca. sql code
    say result' returned by RXDB2 module.'
    say 'Sql ca. Sql code' returned by DB2. - query was unsuccessful.'
    say space(Sql ca. Sql msg)
    exit 20
end

return CALLDB2RC

```

```

                                LPREXECS
RxDB2:; Signal Off Error; Signal Off Failure
Parse Arg RxDB2SQLSysId, RxDB2SQLStmt, RxDB2SQLStem
Address "LINK" "DBM103"
Return Rc
/* use this when you use level like 'CSS8.V7F' */

HLIST_LEVEL:

/* executes the tso command HLIST to list the attributes of the
migrated dataset information */

dsname='WELZ492'

"HLIST LEVEL('"Dsname"' ) MCD outdataset('css8.v7f.rout' )"
return

/* use this when you use complete dataset name' */
HLIST_DSN:

DUMMY = OUTTRAP("LCT.", "**")

do I = 1 TO LCLIST.0

    CURRENTDSN = WORD(LCLIST.I, 1)
    "HLIST DATASETNAME('"CURRENTDSN"' ) MCD outdataset('css8.v7f.rout' )"

end
DUMMY = OUTTRAP(OFF)

"FREE DDNAME(INFILE)"
return

DISPLAY_INFORMATION:

/* to display important information */

user      = sysvar(sysuid)
pref      = sysvar(syspref)
logon_proc = sysvar(sysproc)
date_normal = date()
date_julian = date('j')
curr_time = time()
ispf      = sysvar(sysispf)
env       = sysvar(sysenv)
cpu_time  = sysvar(syscpu)
hsm_level = sysvar(SYSHSM)
if hsm_level = '' then
    hsm_level = ' Not Active '
racf_level = sysvar(SYSLRACF)
if racf_level = '' then
    racf_level = ' Not active '
tsoe_level = sysvar(SYSTSOE)
jes_level  = sysvar(SYSJES)
node_name  = sysvar(SYSNODE)

say "Your TSO user ID is           =====>" user
say "Your prefix as defined in the prfoile =====>" pref
say "Your logon procedure name is   =====>" logon_proc
say "Current Date is                =====>" date_normal
say "Current Date -   in julian format =====>" date_julian
say "Current Time is                =====>" curr_time
say "Whether ISPF dialog manager services are available "
say "for the EXEC (ACTIVE OR NOT NOT ACTIVE " ispf
say "Whether this EXEC is runniNG FOREground/BACKground " env
say "Cpu time consumed              " cpu_time
say "HSM level installed            " hsm_level
say "RACF level installed           " racf_level

```

```

                                LPREXECS
say "TSOE level installed" " tsoe_level
say "JES level installed" " jes_level
say "SYSTEM NODE NAME" " node_name
say "=====
say " the cpu information" "
say "=====

x = SYSCPUS('CPUS.')
if x = 0 then
  do
    say 'Number of online CPUs is ' CPUS.0
    do i = 1 to CPUS.0
      say 'CPU' i ' has CPU info ' CPUS.i
    end
  end
else
  say ' function performed is not successful '

pull " " /* wait */
return

HEXADECIMAL_TO_DECIMAL:
/* to convert the hexadecimal value to decimal */
pull hex
dec = x2d(hex)
say "Hexadecimal " hex " = Decimal " dec

pull " " /* wait */
return

DISPLAY_LINKLIST:
/* to display link list libraries */
/* CVT is Communciation Vector Table */

/* TRACE INT */

/* to obtain the content of 4 bytes at storage location with address */
/* HEX'10' (DECIMAL 16). This is the address of the CVT table */

CVT_ADDR_CHAR = STORAGE(10, 4)
CVT_ADDR_HEX = C2X(CVT_ADDR_CHAR)
CVT_ADDR_DEC = C2D(CVT_ADDR_CHAR)

/* in the CVT at offset 1244(decimal) or 4DC (hexa decimal) is the */
/* address of the link-list table. */

PTR_LINKLIST_TABLE_ADDR_DEC = CVT_ADDR_DEC + 1244
PTR_LINKLIST_TABLE_ADDR_CHAR = D2C(PTR_LINKLIST_TABLE_ADDR_DEC, 4)
PTR_LINKLIST_TABLE_ADDR_HEX = C2X(PTR_LINKLIST_TABLE_ADDR_CHAR)
LINKLIST_TABLE_ADDR_CHAR = STORAGE(PTR_LINKLIST_TABLE_ADDR_HEX, 4)
LINKLIST_TABLE_ADDR_HEX = C2X(LINKLIST_TABLE_ADDR_CHAR)

/* bytes 5 through 8 of A contains number of entries in the */
/* LINKLIST_TABLE */

A = STORAGE(LINKLIST_TABLE_ADDR_HEX, 8)
NUMBER_OF_ENTRIES_CHAR = SUBSTR(A, 5, 4)
NUMBER_OF_ENTRIES_DEC = C2D(NUMBER_OF_ENTRIES_CHAR)

say 'Number of LINKLIST LIBRARIES = ' NUMBER_OF_ENTRIES_DEC
say ''
say 'The following are the LinkList libraries defined to the system'
say '
MAX_LENGTH_POSSIBLE = (NUMBER_OF_ENTRIES_DEC * 45) + 8
LINKLIST_TABLE_DATA = STORAGE(LINKLIST_TABLE_ADDR_HEX, MAX_LENGTH_POSSIBLE)
START_POSITION = 10
do I = 1 to NUMBER_OF_ENTRIES_DEC

```

LPREXEC5

```

DATASET_NAME = ,
              SUBSTR(LINKLIST_TABLE_DATA, START_POSITION, 44)
START_POSITION = START_POSITION + 45
say DATASET_NAME
end
pull
return

DISPLAY_APFLIST:
/* to display APF list libraries */
/* CVT is Communciation Vector Table */

/* to obtain the content of 4 bytes at storage location with address */
/* HEX'10' (DECIMAL 16). This is the address of the CVT table */

CVT_ADDR_CHAR      = STORAGE(10, 4)
CVT_ADDR_HEX       = C2X(CVT_ADDR_CHAR)
CVT_ADDR_DEC       = C2D(CVT_ADDR_CHAR)

/* in the CVT at offset 484(decimal) or 1E4 (hexa decimal) is the */
/* address of the APF table. */

PTR_AUTH_TABLE_ADDR_DEC = CVT_ADDR_DEC + 484
PTR_AUTH_TABLE_ADDR_CHAR = D2C(PTR_AUTH_TABLE_ADDR_DEC, 4)
PTR_AUTH_TABLE_ADDR_HEX = C2X(PTR_AUTH_TABLE_ADDR_CHAR)

AUTH_TABLE_ADDR_CHAR = STORAGE(PTR_AUTH_TABLE_ADDR_HEX, 4)
AUTH_TABLE_ADDR_HEX = C2X(AUTH_TABLE_ADDR_CHAR)

NUMBER_OF_ENTRIES_CHAR = STORAGE(AUTH_TABLE_ADDR_HEX, 2)
NUMBER_OF_ENTRIES_DEC = C2D(NUMBER_OF_ENTRIES_CHAR)

say 'The following are the APF      Libraries defined to the system'
say '-----'
MAX_LENGTH_POSSIBLE = (NUMBER_OF_ENTRIES_DEC * 51) + 2
APF_TABLE_DATA = ,
               STORAGE(AUTH_TABLE_ADDR_HEX, MAX_LENGTH_POSSIBLE)
START_POSITION = 3

say 'VOL-SER      THE NAME OF THE APFLIBRARY'
say '-----'
say ' '

do I = 1 to NUMBER_OF_ENTRIES_DEC

  ENTRY_LENGTH_CHAR = SUBSTR(APF_TABLE_DATA, START_POSITION, 1)
  ENTRY_LENGTH_DEC = C2D(ENTRY_LENGTH_CHAR)
  LENGTH_OF_DSN = ENTRY_LENGTH_DEC - 6
  VOLSER = SUBSTR(APF_TABLE_DATA, START_POSITION+1, 6)

  DATASET_NAME = ,
               SUBSTR(APF_TABLE_DATA, START_POSITION+7, LENGTH_OF_DSN)
  START_POSITION = START_POSITION + 1 + ENTRY_LENGTH_DEC
  say VOLSER ' ' DATASET_NAME
end

pull
return

BINJCL:

L1=' //WELZ492B JOB '""' 001200-1-90492-000399' , ' DV1' "" , MSGCLASS=R,      '
L2=' //      CLASS=B, LINES=850000,      '
L3=' //      REGION=6M, MSGLEVEL=(1, 1), NOTIFY=&SYSUID, TIME=1200      '
L4=' /*JOBPARM ROOM=3D1, PROCLIB=PROC31      '
L5=' //*****'
L6=' //BINSTEP EXEC PGM=IKJEFT1A, DYNAMNBR=20, COND=(4, LT) '
L7=' //STEPLIB DD DSN=' DB2SS' . DBA. DB2. DSNLIB, DISP=SHR      '

```

LPREXEC5

```
L8=' //          DD  DSN=' DB2SS' . DBA. DB2. RUNLIB, DISP=SHR
L9=' //          DD  DSN=' DB2SS' . DBA. DB2. INTEX. LOADLIB, DISP=SHR'
LO=' //SYSTSPRT DD  SYSOUT=*'
LA=' //SYSPRINT DD  SYSOUT=*'
LB=' //SYSOUT   DD  SYSOUT=*'
LC=' //SYSTSIN  DD  *'
LD='   DSN SYSTEM(' DB2SS' )'
```

```
push L1 ; "EXECIO 1 DISKW REPTFILE"
push L2 ; "EXECIO 1 DISKW REPTFILE"
push L3 ; "EXECIO 1 DISKW REPTFILE"
push L4 ; "EXECIO 1 DISKW REPTFILE"
push L5 ; "EXECIO 1 DISKW REPTFILE"
push L6 ; "EXECIO 1 DISKW REPTFILE"
push L7 ; "EXECIO 1 DISKW REPTFILE"
push L8 ; "EXECIO 1 DISKW REPTFILE"
push L9 ; "EXECIO 1 DISKW REPTFILE"
push L0 ; "EXECIO 1 DISKW REPTFILE"
push LA ; "EXECIO 1 DISKW REPTFILE"
push LB ; "EXECIO 1 DISKW REPTFILE"
push LC ; "EXECIO 1 DISKW REPTFILE"
push LD ; "EXECIO 1 DISKW REPTFILE"
```

```
NODE_NAME = sysvar(SYSNODE)
COLL_ID = cln
```

```
if SUBSTR(COLL_ID, 1, 3) = 'SWW' then
do
    OWN_ID = 'DASADC1'
    QUAL_ID = 'DASADC1'
end
else
do
    OWN_ID = 'DSSADC1'
    QUAL_ID = 'DSSADC1'
end
```

```
t_dbrmlib = substr(coll_id, 4, 3)
dbrmlib = "" TPD. " || t_dbrmlib || ". DBRMLIB "
```

```
Q="SELECT 'BIND PACKAGE("COLL_ID") MEMBER(' || NAME || ')
      ' LIBRARY(' || dbrmlib || ')
      ' OWNER("OWN_ID") QUALIFIER("QUAL_ID") CURRENTDATA(YES)
      ' VALIDATE(BIND) EXPLAIN(YES) ACTION(REPLACE)
      ' SQLERROR(NOPACKAGE) ISOLATION(CS)
      ' DEGREE(1) NOREOPT(VARS) KEEP DYNAMIC(NO)
      ' FLAG(I);
FROM SYIBM.SYSPACKAGE
WHERE COLLID=' || COLL_ID || ""
```

```
call CALLDDB2 Q
do row = 1 to rt.0
    FIRST_R = ' ' || RT. 1. row
    push FIRST_R ; "EXECIO 1 DISKW REPTFILE"

    SECOND_R = ' ' || RT. 2. row
    push SECOND_R ; "EXECIO 1 DISKW REPTFILE"

    THIRD_R = ' ' || RT. 3. row
    push THIRD_R ; "EXECIO 1 DISKW REPTFILE"

    FOURTH_R = ' ' || RT. 4. row
    push FOURTH_R ; "EXECIO 1 DISKW REPTFILE"

    FIFTH_R = ' ' || RT. 5. row
    push FIFTH_R ; "EXECIO 1 DISKW REPTFILE"
```

```

                                LPREXECS
SIXTH_R          = '          ' || RT. 6. row
push SIXTH_R     ; "EXECIO 1 DISKW REPTFILE"

SEVENTH_R        = '          ' || RT. 7. row
push SEVENTH_R   ; "EXECIO 1 DISKW REPTFILE"

end
"EXECIO 0 DISKW REPTFILE ( FINIS"
"FREE DDNAME(REPTFILE)"
return

SCANPDS:
unitname = 'VI0'      /* Change this if allocations fail */

Parse Source . . mname .
/*Address isredit
'MACRO (INPARMS)';
If rc>0 Then
  Do; Say mname 'must be invoked as an edit command.' ;Exit 0 ;End
  If inparms='' Then
  Do; Say 'The 'mname' command requires a string to find.' ;Exit 12; End
  If substr(inparms,1,1)="" Then Parse Var inparms "" inparms "" rest
  Else
  If substr(inparms,1,1)=' ' Then Parse Var inparms ' ' inparms ' ' rest
  Else rest=' '
  rest=translate(space(rest))
  If rest\=' ' Then rest=', 'rest
  Upper inparms*/
Call getdsns
Parse Value zdsn.0 zdsn.1 zdsn.2 zdsn.3 With zdsn0 zdsn1 zdsn2 zdsn3
'(MEMBER) = MEMBER'
If member = '' Then
  Do
    zerrsm='Must be partitioned'
    zerrlm=mname' is only valid with partitioned data sets.'
    zerrhm='*'
    zerralm=' YES'
    Address ispexec 'SETMSG MSG(ISRZ002)'
    Exit 12
  End
Address 'TS0'
'ALLOC F(SYSIN) REUSE NEW DEL UNIT('unitname') LRECL(80) RECFM(F B)
SPACE(10,10) TRACK RELEASE DSO(PS) BLKSIZE(80)'
Do queued();Pull;End
Queue 'SRCHFOR ''inparms''rest
Address 'TS0' 'EXECIO 1 DISKW SYSIN (FINIS'
'ALLOC F(NEWDD) DA('strip(zdsn.0 zdsn.1 zdsn.2 zdsn.3)') SHR REUSE'
rdd=' $'right(time(s),7,'0')
'ALLOC F('rdd') DA('strip(zdsn.0 zdsn.1 zdsn.2 zdsn.3)') SHR REUSE'
'ALLOC F(OUTDD) REUSE NEW DEL UNIT('unitname') LRECL(80) RECFM(F B)
SPACE(10,10) TRACK RELEASE DSO(PS)'
supparm=' SRCHCMP, ANYC, LMT0, NOSUMS, NOPRTCC, CKPACKL'
a=1
'ALLOC NEW DEL F($UPDPAN) DSO(PO) DIR(1) SP(3,3) TRACK
REUSE RECFM(F B) BLKSIZE(0) LRECL(80) UNIT('unitname')'
Do Until substr(line,1,7)='/*PANEL'
  line = sourceline(a)
  a=a+1
End
Parse Var line . panel name .
Address ispexec
'LIMIT DATAID(TMPPNL) ENQ(EXCLU) DDNAME($UPDPAN)'
'LMOPEN DATAID('tmppnl') OPTION(OUTPUT)'
Do Until substr(line,1,4)=')END'
  line = sourceline(a)
  'LMPUT DATAID(&TMPPNL) MODE(INVAR) DATALOC(LINE) DATALEN(80)'
  a=a+1
End

```

```

                                LPREXECs
' LMMADD DATAID(&TMPPNL) MEMBER(' panel name' )'
' LMFREE DATAID(&TMPPNL)'
' LIBDEF ISPLIB LIBRARY ID($UPDPAN) STACK'
' ADDPOP'
' CONTROL DISPLAY LOCK'
' DISPLAY PANEL(POPUP)'
' SELECT PGM(ISRSUPC) PARM(&SUPPARM)'
supercrc=rc
' REMPOP'
' LIBDEF ISPLIB '
Address tso
' FREE F($UPDPAN)'
If supercrc=1 Then
  Do
    ' EXECIO * DISKR OUTDD (FINIS STEM MEM.'
    Address ispexec
    ' CONTROL ERRORS RETURN'
    ' LIMIT DATAID(DATAID) DDNAME(' rdd' ) ENQ(SHR)'
    ' LMOOPEN DATAID(' dataid' )'
    ' CONTROL NONDISPL END'
    zerrsm=' Found in 'mem.0-5' members'
    zerrlm=' The string "' inparms'" was found in 'mem.0-5' members.'
    zerrhm=' *'
    zerralm=' NO'
    Address ispexec ' SETMSG MSG(ISRZ002)'
    option=' DISPLAY) COMMANDS(ANY'
    zcmd = ''
    Do a=6 to mem.0
      Parse Var mem.a member .
      ' LMMDISP DATAID(' dataid' ) OPTION(' option' ) ' ,
      ' MEMBER(' member' )'
      If rc<=8 Then /* ignore bad names for now */
        option=' ADD'
      Else
        ' CONTROL NONDISPL END' /* don't display next attempt*/
    End
    address tso ' FREE F(NEWDD SYSIN OUTDD)'
    zlmember=''
    Do Until di sprc>0
      ' LMMDISP DATAID(' dataid' ) OPTION(DISPLAY) COMMANDS(ANY) ,
      TOP(' zlmember' )'
      di sprc=rc
      If di sprc=0 & zlmember\='' Then
        Do
          service = ''
          SELECT
            When (zllcmd=' V' ) Then service=' VIEW'
            When (zllcmd=' B' ) Then service=' BROWSE'
            When (zllcmd=' S' ) Then service=' EDIT'
            When (zllcmd=' /' ) Then service=' EDIT'
            When (zllcmd=' E' ) Then service=' EDIT'
            Otherwise
              Do
                zerrsm=' Invalid command: ' zllcmd
                zerrlm=' Use B for BROWSE, V for VIEW or E, S or / for EDIT.'
                zerrhm=' *'
                zerralm=' YES'
                Address ispexec ' SETMSG MSG(ISRZ002)'
              End
            End
          If service \= '' Then
            Do
              service ' DATAID(' dataid' ) MEMBER(' zlmember' )'
              If rc>=12 Then ' SETMSG MSG(ISRZ002)'
            End
          End
        End
      End
    End
    ' LMMDISP DATAID(' dataid' ) OPTION(FREE)'

```

LPREXECS

```

' LMFREE DATAID(' dataid')'
End
Else
Do
zerrsm='String not found'
zerrlm='The string "'inparms'" was not found in any members.'
zerrhm='*'
zerralm='YES'
Address ispeexec 'SETMSG MSG(ISRZ002)'
address tso 'FREE F(NEWDD SYSIN OUTDD)'
End
address tso 'FREE F(' rdd')'

Exit 0
getdsns: Procedure Expose zdsn. panel zwidth
zdsn. =''
tfdp = ptr(76+ptr(640+ptr(ptr(24+ptr(112+ptr(132+ptr(540)))))))
panel =storage(d2x(344+ptr(ptr(24+ptr(112+ptr(132+ptr(540)))))),8)
Do a=0 to 3
Parse Value storage(d2x(ptr(140+a*4+tfdp)),46) With d 3 n
If d>'0000' x Then zdsn.a="" "stri p(n)"" "
End
Return
ptr: Return c2d(bitand(storage(d2x(Arg(1)),4),'7FFFFFFF' x))
/*PANEL POPUP
)ATTR
@ type(pt)
+ type(nt)
/ type(text) color(red)
)BODY window(60,15)
+ @Searching Data Set(s)+
+
+String: /&i nparms
+Data sets:%&zdsn0
+ :%&zdsn1
+ :%&zdsn2
+ :%&zdsn3
)END
*/

```

```

REAL_NAME:
ALIAS_NAME=dname
' FREE F(LPRLLIB)'
"ALLOCATE DATASET('CSS8.V7F."LPRN".LOADLIB') FILE(LPRLLIB) SHR"
ADDRESS ISPEXEC "LIBDEF ISPLLIB EXCLLIBR ID(LPRLLIB)"
ADDRESS ISPEXEC "LIBDEF STEPLIB DATASET ID('CSS8.V7F."LPRN".LOADLIB')"
```

```

REAL_DSN_NAME = LPRO01(ALIAS_NAME)
SAY 'REAL_NAME IS ==> ' REAL_DSN_NAME
RETURN
```

```

PROFDUMP:
Parse Arg dsn appl .
' ALLOC F(FOOBAR) DA(' dsn') MOD DEL UNIT(SYSDA) REU'
' ALLOC F(FOOBAR) DA(' dsn') NEW CAT TR SPA(20,20) RELEASE REUSE',
' DSORG(PS) RECFM(V B) LRECL(500) BLKSIZE(0) UNIT(SYSDA)'
Do queued()
Pul l
End /* JUST IN CASE */
Address ispeexec
' CONTROL ERRORS RETURN'
' VGET ZAPPLID'
parse upper value appl zapplid with appl .
tablenam=appl ' PROF'
' TBTOP ' tablenam
' TBSKIP' tablenam 'NUMBER(1) SAVENAME(SAVES)'
say skip rc

```

## LPREXEC

```

Parse Var saves ('saves')
Queue tablenam: TOTAL VARIABLES: words(saves)
Queue copies('-', length(tablenam: TOTAL VARIABLES: words(saves)))
tlen=0
Do While saves \= ''
  Parse Var saves word saves
  'VGET 'word' PROFILE'
  Interpret 'LEN = LENGTH('word')'
  tlen=tlen+l en
  Interpret 'VAL = 'word
  Queue Left(word, 9)right(l en, 5)' 'val
End
Queue 'Total variable lengths: ' tlen
Address tso
' EXECIO 'queued()' DISKW FOOBAR (FINIS'
' FREE F(FOOBAR)'
Do queued()
  Pul l
End /* JUST IN CASE */
say ' Output Will be Stored in ' || sysvar(sysuid) || '.' || dsn
say ' Press Enter '
pul l
return

```

## SHOWMVS\_PARA:

```

"ALLOCATE DATASET('CSS8.V7F."LPRN".LOADLIB') FILE(LPRLLIB) SHR"
ADDRESS ISPEXEC "LIBDEF ISPLLIB EXCLLIBR ID(LPRLLIB)"

```

```

call SHOWMVS /* assembly program */
RETURN
EXIT:
/* EXIT ROUTINGE CLOSE THE ALLOCATED FILES BEFORE EXITING */
"FREE DDNAME(INFILE)"
"FREE DDNAME(REPTFILE)"
"FREE DDNAME(LPRLLIB)"
exit

```

## LOADEXEC:

```

ARG EDIT_FLAG
call bldtbl
ADDRESS ISPEXEC
' FTOPEM TEMP'
' FTINCL LOADSKL' /* loadskl is available in CSS8.V7F.DV.ISPLIB' */
' FTCLOSE'
ADDRESS ISPEXEC "EDIT DATASET('&ZTEMPF')"
RETURN 0

```

## BLDTBL:

```

level_name = 'nfo7.db.' || saveid

if redef = 'Y' then
do
  DUMMY_VAR = OUTTRAP("D_S_N.", "*", "CONCAT")
  say level_name
  "LISTCAT LEVEL('"level_name"')"

  if RC \= 0 then
  do
    say 'ERROR LISTING THE DATASETS'
    EXIT
  end
ELSE
do
  READ_LINES = D_S_N.0
end

```

## LPREXECs

```

DUMMY_VAR = OUTTRAP("OFF")

say 'second'
say "No of datasets under this save_id : " READ_LINES/2

Q= " delete from lpradu1.nfo_tablespace"
call CALLDB2 Q
do I = 1 TO READ_LINES BY 2
  DSN_TYPE = WORD(VALUE(D_S_N. ""I""), 1)
  if DSN_TYPE \= 'GDG' then
    do
      DSN_HYPHEN = WORD(VALUE(D_S_N. ""I""), 2)
      DSN_NAME = WORD(VALUE(D_S_N. ""I""), 3)
    end
  ELSE
    do
      DSN_HYPHEN = WORD(VALUE(D_S_N. ""I""), 3)
      DSN_NAME = WORD(VALUE(D_S_N. ""I""), 4)
    end
  table_space = substr(dsn_name, 18, 8)
  say 'table_space ' table_space

  Q= "INSERT INTO LPRADU1.NFO_TABLESPACES VALUES('"table_space"')"

  call CALLDB2 Q
end
end

Q= "SELECT A.NAME, B.NAME
FROM LPRADU1.NFO_TABLESPACES A , SYSIBM.SYSTABLES B
WHERE B.CREATOR = 'DSSADC1'
AND B.TSNAME = A.NAME "
call CALLDB2 Q
dbname = 'NFOCDB01'
ADDRESS ISPEXEC "TBCREATE NFOTS NAMES(tsname sysdd tname) LIBRARY(ISPTLIB)",
"write replace"
do row = 1 to rt.0
  tsname= rt.1.row
  sysdd = 'SYSR' || right(row, 4, '0')
  tname= rt.2.row
  address ISPEXEC "TBADD NFOTS order"
end
ADDRESS ISPEXEC "TBCLOSE NFOTS"
return

RMMCOPY:

TCNT = 0
F_TIME = 'T'
ptvol = '000000'

"ALLOCATE DDNAME(ISPFIL)
DSNAME('MRK7.TEST.MEDTAPE.COPY') SHR"
if RC \= 0 then
DO
  say 'ALLOCATE ERROR ON ISPFIL ' FILE_NAME
  EXIT
END

do I = 1 TO LCLIST.0
  TDSN = WORD(LCLIST.I, 1)
  TVOL = WORD(LCLIST.I, 2)
  TLBL = WORD(LCLIST.I, 5)
  call BLD_DSN_TBL TVOL
end

ADDRESS ISPEXEC

```

LPREXEC

```

    'TBCLOSE 'TNAME''
    'FTOPEN'
    'FTINCL RMMCSKEL'
    'FTCLOSE NAME('tname')'
EXIT

BLD_DSN_TBL:
arg TVOL
if TVOL = ptvol then
do
    address ISPEXEC "TBADD "TNAME" order"
end
else
do
    if F_TIME = 'T' then
do
        F_TIME = 'F'
end
else
do
        tdsn_swap = tdsn
        ADDRESS ISPEXEC "TBCLOSE "TNAME""
        ADDRESS ISPEXEC
        'FTOPEN'
        'FTINCL RMMCSKEL'
        'FTCLOSE NAME('tname')'
        tdsn = tdsn_swap
    end
    ptvol = TVOL
    TCNT = TCNT + 1
    tname = 'TBL' || TCNT
    ftdsn = tdsn
    ADDRESS ISPEXEC "TBCREATE "tname" NAMES(tdsn tlib) ",
        "LIBRARY(ISPTLIB) write replace"
    /*address ISPEXEC "TBADD "TNAME" order" */
end
RETURN 0
CALLDB2:
/* Used inside DB2REXX paragraph */

arg SQLQUERY
address
if sysvar(SYSENV)='FORE' then "ALLOCATE DD(SYSOUT) DSN(*) SHR REUSE"
rt. = ''
call rxdb2 db2ss, SQLQUERY, 'RT.'
CALLDB2_RC=rc

if sysvar(SYSENV)='FORE' then "FREE DD(SYSOUT)"

if CallDB2_RC > 4 then
do
    say ' * * * Call error 'CALLDB2_RC
    exit 20
end

if sqlca.sqlcode = 100 then
do
    say ' * * * No rows.'
    say ' enter Y If you want to exit'
    pull no_row_exit
    UPPER no_row_exit
    if no_row_exit = 'N' then
        return CallDB2RC
    else
        exit 100
end

if sqlca.sqlcode \= 0 then

```

LPREXECS

```
do
  say ' * * * sql error ' sql ca. sql code
  say result' returned by RXDB2 module.'
  say Sql ca. Sql code' returned by DB2. - query was unsuccessful .'
  say space(Sql ca. Sql msg)
  exit 20
end
```

return CallDB2RC

```
RxDB2: ; Signal Off Error; Signal Off Failure
Parse Arg RxDB2SQLSysId, RxDB2SQLStmt, RxDB2SQLStem
Address "LINK" "DBM103"; Return Rc
```

```
/* use this when you use level like 'CSS8.V7F' */
```