132

September 1997

MVS update

In this issue

MVS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: stevep_xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

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

Editor

Steve Piggott

Disclaimer

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

MVS Update on-line

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

A REXX framework

PROBLEM ADDRESSED

There are many applications in the MVS environment, especially in the user-productivity field, that are written in REXX. Such applications often make use of existing programs and procedures and excessive run-time can result if the same program is loaded repeatedly. This can occur when the same programs are used in several procedures: loaded programs are deleted when the associated procedure terminates. REXXFWK solves this problem by preloading such programs and then invoking the required main procedure. Programs are deleted only when the highest level procedure terminates, ie the procedure that invokes REXXFWK.

INVOCATION

```
ADDRESS LINK "REXXFWK" /T pgmname .../procname procparms
```

where:

- /T is the trace flag (ie list the loaded programs).
- *pgmname* is the name of a program or list of programs to be loaded. Each *pgmname* entry is delimited by a single blank. The programs should be reusable.
- *procname* is the name of the REXX procedure that is to be invoked: this is the main procedure.
- *procparms* are the parameters to be passed to the main procedure.

SAMPLE INVOCATION

```
ADDRESS LINK "REXXFWK /T TX001 TX002 TX004/TXGO"
```

Loads the TX001, TX002, and TX004 programs, and invokes the TXGO procedure.

PROGRAM CODE

```
REXXFWK  TITLE 'REXX INVOCATION FRAMEWORK'
* Function: Load reusable (REXX-)programs and pass control to REXX
* procedure.
REXXFWK  CSECT
REXXFWK  AMODE 31
REXXFWK  RMODE 24
         BAKR  R14,Ø              save registers
         BASSM R12,Ø              set base register
         USING *,R12
         LA    R13,SA             set internal save area
         MVC   4(4,R13),=C'F1SA'
         MVC   RC,=H'8'           ReturnCode (parm omitted error)
         LM    R2,R3,Ø(R1)
         L     R2,Ø(R2)           A(parm)
         ICM   R3,B'1111',Ø(R3)   L(parm)
         BZ    EXIT               parm missing
* test for '/T ' flag
         CLC   =C'/T ',Ø(R2)
         BNE   NO_TFLAG
         OI    FLAG,X'Ø1'         set option flag
         LA    RØ,3               length of option
         AR    R2,RØ              correct parm start
         SR    R3,RØ              correct parm length
* search for '/'
NO_TFLAG MVC   RC,=H'12'          ReturnCode (parm length error)
         LR    R6,R2              parm start
         LA    R5,Ø(R2,R3)        parm end
         LR    R15,R5             save parm end address
         L     RØ,=X'ØØØØØØ61'
         SRST  R5,R6
         BH    EXIT               no procedure name specified
         LA    R14,1(R5)          addr user parm
         SR    R15,R14            length
         BZ    EXIT               parm omitted
         CH    R15,=AL2(L'PCMD)
         BH    EXIT               parm too long
         ST    R15,PCMDL          parm length
         LA    RØ,PCMD
         LR    R1,R15
         MVCL  RØ,R14
         STM   R5,R6,APARM        parm end, parm start
* load resident modules
LOOP     LR    R4,R5              save parm end
         CR    R6,R5
         BH    LOOPEND
         L     RØ,=X'ØØØØØØ4Ø'    search for blank (delimiter)
         SRST  R5,R6
         LR    R14,R6
```

```
         LR    R15,R5
         SR    R15,R14             length
         O     R15,=X'40000000'
         LA    RØ,PGMNAME
         LA    R1,L'PGMNAME
         MVCL  RØ,R14
         LA    R6,1(R5)            restart address
         LR    R5,R4
         LOAD  EPLOC=PGMNAME       make access module resident
         TM    FLAG,X'Ø1'
         BZ    LOOP                no trace
         TPUT  PGMNAME,PGMNAMEL
         B     LOOP
LOOPEND  DS    ØH
* invoke procedure
         L     R15,CVTPTR          A(CVT)
         USING CVT,R15             CVT-DSECT
         L     R15,CVTTVT          A(TSO Vector Table)
         L     R15,TSVTASF-TSVT(R15)       A(TSO Service Facility)
         CALL  (15),(POPTS,PCMD,PCMDL,PCRC,PTSFRSC,PCAC),VL
* delete loaded modules
         LM    R5,R6,APARM         parm end, parm start
         MVC   PGMMSG,=CL8'deleted'
DLOOP    LR    R4,R5               save parm end
         CR    R6,R5
         BH    DLOOPEND
         L     RØ,=X'00000040'     search for blank (delimiter)
         SRST  R5,R6
         LR    R14,R6
         LR    R15,R5
         SR    R15,R14             length
         O     R15,=X'40000000'
         LA    RØ,PGMNAME
         LA    R1,L'PGMNAME
         MVCL  RØ,R14
         LA    R6,1(R5)            restart address
         LR    R5,R4
         DELETE EPLOC=PGMNAME      remove module
         TM    FLAG,X'Ø1'
         BZ    DLOOP               no trace
         TPUT  PGMNAME,PGMNAMEL
         B     DLOOP
DLOOPEND MVC   RC,=H'Ø'
EXIT     LH    R15,RC
         PR
SA       DS    18F                 user save area
RC       DS    H
APARM    DS    2A
POPTS    DS    ØXL4
         DC    X'ØØ',X'Ø1',X'ØØ',X'Ø1'  -,unauth,no dump,REXX exec
```

```
PCMD     DC    CL256' '
PCMDL    DS    F
PCRC     DS    F      command return code
PTSFRSC  DS    F      TSF reason code
PCAC     DS    F      command ABEND code
FLAG     DC    X'Ø'   option flag
* X'Ø1': trace
PGMNAME  DC    CL9' '
PGMMSG   DC    CL8'loaded'
PGMNAMEL EQU   *-PGMNAME
* DSECT definitions
         IKJTSVT
         CVT   DSECT=YES
         END
```

A S Rudd
Technical Consultant (Germany)

ISPF panel user exits

INTRODUCTION

One of the most powerful products provided by IBM for the MVS/TSO environment is the Interactive System Productivity Facility/Program Development Facility, better known as ISPF/PDF or simply ISPF. ISPF provides a wide variety of functions and services that facilitate the development of powerful interactive applications known as ISPF dialogs.

The list of ISPF services consists of the Browse Interface (also known as BRIF) service, the BROWSE service, the EDIT service, the Edit Interface (also known as EDIF) service, the Edit Recovery for EDIF (also known as EDIREC) service, the Edit Recovery for Edit (also known as EDREC) service, as well as a set of library access services. Additional information about the functionality and purpose of these services can be found in Chapter One of the *ISPF/PDF Services* manual.

ISPF services can be invoked by either compiled programs, CLISTs, or REXX commands. Language support is provided for PL/I, C,

COBOL, FORTRAN, PASCAL, APL2, and Assembler. Again, details on how to invoke ISPF services using any of the above listed computer languages can be found in the *ISPF/PDF Services* manual.

One of the most important aspects of ISPF dialogs is the mechanism used by ISPF to request input data or to display output data. These mechanisms are supported by a set of ISPF functions, one of which is the DISPLAY function.

The DISPLAY function is used to present an ISPF panel or an ISPF table to the user. In simple terms, an ISPF panel is the visual display presented on your monitor at the time the ISPF dialog is requesting or displaying information, for example a menu with a list of options. In complex terms, an ISPF panel is a set of ISPF panel definition statements that will process dialog variables, permitting an ISPF dialog: the interactive input and output of data. ISPF panels are not an ISPF service. However, they are a very important part of an ISPF dialog because they define what is being presented to the user as well as what is to be requested from the user.

Using ISPF panel definition statements, one specifies the different characteristics of the panel, such as what colours will be used, what text will be displayed, which fields will be presented for input, which fields will be presented for output, etc. For complete information on ISPF panel definitions, see Chapter 7 of the *ISPF Dialog Management Guide and Reference*.

InISPF/PDF V3.*x* and V4.*x*, there are twelve different types of ISPF panel definition statement. They are assignment, EXIT, GOTO, IF, ELSE, REFRESH, TOG (toggle), VER(verify), VEDIT, VGET, VPUT, and PANEXIT (panel user exit). In V4.*x*, there are eight built-in functions: TRUNC (truncate), TRANS (translate), PFK (function key), LVLINE (last visible line), ADDSOSI (add shift-out character), DELSOSI (delete shift-out character), ONEBYTE (convert to a one-byte code), and TWOBYTE (convert to a two-byte code). These statements provide extensive and powerful functionality which, in turn, facilitates the development of the ISPF dialogs, since a lot of the validating of variables can be handled by the panel-processing features. Again, refer to Chapter 7 of the *ISPF Dialog Management Guide and Reference* for additional information on the functionality of each one of the above statements, as well as panel-processing features.

An important concept about ISPF panels is how ISPF processes the panel definition statements.

ISPF PANEL EXITS: BUILD YOUR OWN FUNCTION

Although the panel definition statements provide a lot of functionality and features, there will be times when these statements will not be able to accomplish the desired results. It could be that a panel needs to perform a set of complex statements several times (kind of a subroutine or common code) or that, although the panel definition statements can accomplish the desired result, a very large number of panel dialog statements may be required. A specific example of panel dialog statement limitations is when you need to compare two different input fields and perform an operation depending on the result: ISPF panel statements will allow you to validate each one of the fields individually against predetermined values, but it does not provide a built-in way to validate the fields against each other. The ISPF dialog either performs the required checks after receiving control back from the panel, or a panel user exit could be used to perform the necessary checks while the panel is still validating the user's input.

Panel user exits provide a way of significantly extending the processing capabilities of ISPF panels. Special panel processing requirements, such as verification, transformation, formatting, validation, etc, can be easily implemented via a panel user exit, simplifying the panel definition as well as the ISPF dialog code.

Basically, a panel exit is a compiled program. Panel exits can be coded in any of the programming languages that support ISPF services, such as the languages mentioned above. The exit program uses standard MVS Operating System linkage conventions (register 1 points to a list of addressees etc) and must support 31-bit addressing. Be aware that a panel user exit can't use any dialog variables except those passed on the call, nor can it issue requests for any ISPF services or functions. While developing panel user exits, it is a good idea to have robust error handling routines in the code, as well as debugging facilities, such as displaying messages if a debug parameter is turned on.

The PANEXIT panel statement takes several parameters and, in turn, builds a parameter list that is to be passed to the panel user exit. The

parameters to the panel user exit are a list of addresses, described as follows:

1 EXDATA points to the exit data field used in the PANEXIT call. This parameter, a 4-byte fixed format dialog variable, contains an address of an information area to be passed to the exit routine. This parameter is optional. If not specified, ISPF will use binary zeroes.

2 PANEL NAME points to a left-justified eight character storage area containing the name of the panel from which the panel user exit is being invoked.

3 PANEL SECTION points to a one-character storage area that identifies the panel section from which the panel user exit is being invoked. Possible values are I for the)INIT section, R for the)REINIT section, and P for the)PROC section.

4 MSGID points to a left-justified eight-character storage area containing the name of the error message to be issued in the event of the PANEXIT statement failing or an application error. The name of the error message can be updated by the panel user exit.

5 ARRAY DIMENSION points to a four-byte (fullword) storage area containing the number of parameters being passed to the panel user exit on the PANEXIT statement.

6 ARRAY OF NAMES points to an array of eight-character entries, containing the name of each parameter being passed to the panel user exit on the PANEXIT statement.

7 ARRAY OF LENGTHS points to an array of four-byte (fullword) entries, containing the length of each parameter being passed to the panel user exit on the PANEXIT statement.

8 STRING OF VALUES points to a character buffer of dialog variable values mapped by the VARLEN and VARNAME arrays (6 and 7). These values can be updated by the panel user exit.

Panel exits can be loaded in three different ways, depending on how the panel that uses the exit is invoked. If the panel is being loaded from a program that invokes the ISPF service DISPLAY, the panel exit routine could be statically link-edited with the dialog load module or it could

be dynamically loaded into memory before the panel is displayed. In either case, the program passes the address of the loaded panel exit to ISPF. A third way is when the panel lets ISPF load the panel exit routine dynamically, ISPF doing all the work of allocating the memory for the panel user exit as well as loading the exit into the user's TSO address space. This third way is the easiest to implement.

A dialog invokes a panel user exit by issuing the PANEXIT statement. The syntax of the PANEXIT statement is as follows:

```
PANEXIT ((parm1,parm2,,,,parmn),PGM,exit-address,exit-data,MSG=msgidval)
```

or

```
PANEXIT((parm1,parm2,,,,parmn),LOAD,exit-name,exit-data,MSG=msgidval)
```

where

- *parm1*,*parm2*,,,*parmn* are the names of the ISPF dialog variables being passed to the panel exit routine.
- PGM is the keyword that indicates that the exit routine was either loaded by the application itself or was loaded as part of the application load module.
- LOAD is the keyword that indicates that the exit routine is to be dynamically loaded at the time the panel invokes it.
- *exit-address* is the name of a 4-byte, fixed format, dialog variable that contains the address of the exit routine.
- *exit-name* is the name of the panel user exit routine to be dynamically loaded by ISPF at the time the panel invokes it.
- *exit-data* is the name of a 4-byte fixed format dialog variable that contains a value, such as the address of an information area, to be passed to the exit routine.
- MSGID is an optional keyword.
- *msgidval* is an optional parameter that identifies the message to be displayed upon return from the exit routine after an error has been detected. If none is specified, ISPF uses a generic message. This field can be used by the panel user exit to set different error or debugging messages.

AN EXAMPLE: IMS TIMESTAMP VALIDATION PANEL USER EXIT

Now that we have a general idea of what ISPF panel user exits can do, let's have a look at a panel user exit example. An application and ISPF dialog (CLIST) that generates IMS Database Recovery Control commands wants to ensure that all IMS timestamp values entered via panels represent valid dates. Furthermore, there are several panels in the dialog that have multiple IMS timestamps that need to be checked.

IMS timestamps (prior to Release 5.1) do not comply with Year 2000 requirements. Their format is YYJJJHHMMSST (12 digits). The YY represents the last 2 digits of the year, JJJ represents the Julian day of the year (with a maximum value of 365 or 366 depending on whether it is a leap year or not), HH represents the hour of the day (01 to 23), MM represents the minutes (00 to 59), SS represent the seconds (00 to 59), and T represents the tenths of seconds (0 to 9).

Validating an IMS timestamp using ISPF panel statements is difficult to accomplish. A possible alternative is to perform the IMS timestamp validation inside the ISPF dialog (using a program, CLIST, or REXX command), but it would have complicated the dialog logic, since it would need to coordinate the IMS timestamp validation process with the panel I/O process.

Instead of the above approach, I chose to develop an ISPF panel user exit that provided all the required functionality, enhancing both the robustness of the application as well as the user-interface.

A SAMPLE PANEL TO USE THE IMS TIMESTAMP PANEL USER EXIT

```
)ATTR
)CM*************************************************************
)CM*** COMMENTS FOR THE PANEL    ** remove before executing
)CM*************************************************************
)CM This panel is a subset of a working panel.
)CM Please read the comments and then REMOVE them before
)CM using this panel.
)CM
 @  TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(TURQ)
 _  TYPE(INPUT) INTENS(LOW) HILITE(USCORE) COLOR(RED)
 +  TYPE(TEXT) INTENS(LOW) SKIP(ON)
 %  TYPE(TEXT) INTENS(HIGH) SKIP(ON)
 $  TYPE(OUTPUT) INTENS(LOW)
 £  TYPE(INPUT) INTENS(LOW) COLOR(RED)
```

```
)BODY
!-----------------------   D E L E T E   I C   --------------------------------
%OPTION  ===>£ZCMD                          +
+
%
%         DBD ===>_DBDNAME +
+
%         DDN ===>_DDN     +%or AREA ===>_AREA    +
+
%     RECTIME ===>_RECTIME     +     ( Format@YYDDDHHMMSST+)
+
+
)CM *********************************************************************
)CM *** COMMENTS FOR THE INIT SECTION   ** remove before executing  **
)CM *********************************************************************
)CM 1.  The .CURSOR statement will force the cursor to be positioned at the
)CM     ZCMD field upon first display of the panel.
)CM
)CM 2.  All other statements in the INIT section are for initialization of
)CM     the parameters to be passed into the Panel User exit routine. Note
)CM     that the length of the fields matches what is expected by ISPF.
)CM
)INIT
  .CURSOR   = ZCMD
  &EXITMOD  = DB31AITS      /* PANEL USER EXIT NAME               */
  &EXITDATA = 0000          /* PANEXIT REQUIRED VARIABLE.         */
  &EXITFLAG = N             /* DB31AITS PANEL EXIT PARM. LENGTH=1  */
  &EXITFLD  = DUMMYFLD      /* DB31AITS PANEL EXIT PARM. LENGTH=8  */
  &EXITPOS  = 99            /* DB31AITS PANEL EXIT PARM. LENGTH=2  */
)CM *********************************************************************
)CM *** COMMENTS FOR THE RE-INIT SECTION** remove before executing  **
)CM *********************************************************************
)CM 1.  The REFRESH(*) statement indicates to ISPF that every time the panel
)CM     is re-displayed, proceed to refresh all fields in the panel with the
)CM     most current values from the dialog variable pools.
)CM
)CM 2.  The next 4 statements complement Panel User exit logic.  If the exit
)CM     determined that there was an error in the field being checked, it
)CM     would set the EXITFLAG to 'Y'.  It would also have set the variables
)CM     for the CURSOR and CURSOR position field that will allow the panel to
)CM     reposition the cursor at the exact place where the exit detected the
)CM     error.  The final statement sets the attributes of the cursor, so the
)CM     field in error will be easily detected by the user.
)CM
)REINIT
  REFRESH (*)
  IF ( &EXITFLAG = Y )      /* CHECK IF DB31AITS EXIT SET AN ERROR */
     .CURSOR = &EXITFLD
     .CSRPOS = &EXITPOS
     .ATTR (.CURSOR) = 'COLOR(YELLOW) HILITE(REVERSE)'
```

```
  &EXITFLD  = DUMMYFLD      /* RESET LENGTH OF PANEXIT PARAMETER   */
)CM ********************************************************************
)CM *** COMMENTS FOR THE PROC SECTION   ** remove before executing  **
)CM ********************************************************************
)CM 1.  The REFRESH(*) statement indicates to ISPF that every time the panel
)CM     is re-displayed, proceed to refresh all fields in the panel with the
)CM     most current values from the dialog variable pools.
)CM
)CM 2.  The VER statement is set to check that all positions of the field are
)CM     numeric.
)CM
)CM 2.  If the VER statement did not fail (ie did not set the .MSG variable),
)CM     then the panel logic proceeds to prime the EXITFLD variable and then
)CM     invoke the Panel user exit via the PANEXIT statement.
)CM
)CM Parameters passed to the user exit are :
)CM 1) The IMS timestamp to be checked.
)CM 2) The EXITFLAG field that will be set if an error is found.
)CM 3) The name of the panel field being tested.  This field is used by
)CM    the Panel to reposition the cursor on the field in error.
)CM 4) A variable that will contain a numeric value between 1 and 12.  This
)CM    value represents the position within the field in error, where the
)CM    error was detected by the Panel User exit.
)CM
)CM Other parameters used by the PANEXIT statement are :
)CM -  LOAD.  Keyword to indicate that the panel should be loaded by ISPF.
)CM -  EXITMOD variable, preset in the INIT section to DB31AITS value.  This
)CM -          is the name of the load module to be loaded by ISPF to perform
)CM -          the validation of the IMS timestamp.
)CM -  EXITDATA variable, preset in the INIT section to 0000.  This variable
)CM -          is expected by ISPF.  Since is not being used, is preset to 0.
)CM -  MSG=DBRC313A.  This message is preset in the panel just to allocate
)CM -          the storage before loading the panel user exit. The Panel user
)CM -          exit will be setting the appropriate error message. Upon return
)CM -          to the panel, the MSG value will be different from the initial.
)CM
)PROC
  REFRESH (*)
      VER(&RECTIME,NB,PICT,NNNNNNNNNNNN,MSG=DBRC313A)
      IF (.MSG = &Z )
         &EXITFLD = DUMMYFLD   /* PRIME FIELD TO 8 CHARS */
         PANEXIT((RECTIME,EXITFLAG,EXITFLD,EXITPOS),LOAD,
                  &EXITMOD,&EXITDATA,MSG=DBRC313A)
)END
```

THE PANEL USER EXIT

The panel user exit is coded in Assembler and is named DB31AITS. It can be assembled and link-edited using any standard JCL. Ensure

that the resulting load module uses 31-bit addressing. The load module needs to be placed in a library that can be accessed by the ISPLLIB DDNAME, by the STEPLIB of the TSO procedure you use, or in a library that is part of your LINKLIST. No other special requirements are needed.

```
*-------------------------------------------------------------------*
*    MODULE NAME = DB31AITS ( IMS TIME STAMP PANEL USER EXIT ).     *
*-------------------------------------------------------------------*
*    THIS IS AN ISPF PANEL EXIT ROUTINE.  ITS MAIN PURPOSE IS TO    *
*    CHECK THAT THE TIMESTAMP OBSERVES THE FOLLOWING RULES :        *
*    1 - VALID YEARS ARE 90 THRU 99.                                *
*    2 - VALID DAYS ARE 001 THRU 365.  ALSO CHECKS FOR LEAP YEARS.  *
*    3 - VALID HOURS ARE 00 THRU 23.                                *
*    4 - VALID MINUTES ARE 00 THRU 59.                              *
*    5 - VALID SECONDS ARE 00 THRU 59.                              *
*    6 - VALID FRACTIONS ARE 1 THRU 9.                              *
*    IF AN ERROR IS FOUND, THIS EXIT WILL SET THE ERROR MESSAGE TO  *
*    BE DISPLAYED, AND IT WILL ALSO SET THE POSITION WITHIN THE     *
*    FIELD WHERE THE ERROR WAS FOUND.  PANEL LOGIC WILL CONTROL THE *
*    PLACEMENT AND DISPLAY OF THE CURSOR WHEN AN ERROR IS FOUND.    *
*-------------------------------------------------------------------*
*    THE FOUR PARAMETERS USED BY THIS ROUTINE ARE AS FOLLOWS:       *
*    1 - (INPUT).  - IMS TIMESTAMP, 12 CHARS, FORMAT YYDDDHHMMSST.  *
*    2 - (OUTPUT)  - A FLAG THAT GETS SET TO 'Y' WHEN AN ERROR      *
*        WAS FOUND, AND GETS SET TO 'N' WHEN NO ERROR WAS FOUND.    *
*    3 - (OUTPUT)  - A NAME ( 8 CHARS ) THAT CORRESPONDS TO THE FIELD *
*        WITH THE IMS TIMESTAMP THAT IS BEING CHECKED.              *
*    4 - (OUTPUT)  - A TWO-DIGIT  VALUE THAT TELLS WHERE IN THE FIELD *
*        AN ERROR WAS FOUND.  POSSIBLE VALUES ARE 01,03,06,08, AND 10 *
*    TOTAL LENGTH OF PARAMETERS IS 23 CHARACTERS.                   *
*-------------------------------------------------------------------*
*    THIS ASSEMBLER PROGRAM FOLLOWS THE ISPF CONVENTIONS FOR PANEL  *
*    EXITS AS EXPLAINED IN THE ISPF DIALOG MANAGEMENT GUIDE AND     *
*    REFERENCE, CHAPTER 7.                                          *
*    A DIALOG INVOKES THE PANEL USER EXIT BY ISSUING THE PANEXIT    *
*    STATEMENT FROM A PANEL'S PROC, INIT, OR REINIT SECTION.  ISPF  *
*    INVOKES THE EXIT ROUTINE USING A CALL (BALR 14,15).  OS STANDARD *
*    LINKAGE CONVENTIONS MUST BE USED.                              *
*    ISPF USES THE STANDARD PARAMETER LIST FORMAT TO PASS PARAMETERS. *
*    REGISTER ONE POINTS TO A LIST OF ADDRESSES WHERE EACH ADDRESS  *
*    POINTS TO A DIFFERENT PARAMETER.                               *
*    REG1 --->  ADDR1     EXDATA                                    *
*               ADDR2     PANNAME                                   *
*               ADDR3     PANSECT                                   *
*               ADDR4     MSGID    ( MESSAGE ID TO BE USED BY ISPF ) *
*               ADDR5     ARAYDIM ( VARNAME AND VARLEN DIMMENSION ) *
*               ADDR6     VARNAME ( ARRAY OF NAMES )                *
*               ADDR7     VARLEN   ( ARRAY OF LENGTHS )             *
```

```
*              ADDR8     VARVAL  ( ARRAY OF VALUES )                *
*    RETURN CODES SET IN THE PANEL USER EXIT ROUTINE, RECOGNIZED BY   *
*    ISPF ARE AS FOLLOWS :                                           *
*    Ø  : SUCCESSFUL OPERATION                                       *
*    8  : EXIT DEFINED FAILURE.  ISPF SETS THE .MSG CONTROL VARIABLE  *
*         AND DISPLAYS OR REDISPLAYS THE PANEL WITH THE MESSAGE.     *
*    2Ø : SEVERE ERROR IN THE EXIT ROUTINE.                          *
*-----------------------------------------------------------------*
DB31AITS CSECT
DB31AITS AMODE 31
*** STANDARD CSECT ENTRY INSTRUCTIONS.
         STM   R14,R12,12(R13)         SAVE REGS
         BALR  R12,Ø                   ESTABLISH BASE REGISTER
         USING *,R12                   SET UP CSECT ADDRESSABILITY
         B     AROUND
         DC    CL9'DB31AITS_'
         DC    CL9'&SYSDATE._'
         DC    CL8'&SYSTIME'
AROUND   DS    ØH                      ALIGN INSTRUCTIONS
         ST    R13,SAVEAREA+4          SAVE REGISTER 13
         LA    R13,SAVEAREA            KEEP ADDRESS OF PARMS
***      CLEAR THE MESSAGE AREA AND THE RETURN CODE
         L     R2,12(R1)               LOAD R3 WITH ADDRESS OF MSGID
         USING MSGID,R2                SET ADDRESSABILITY
         MVC   MSGID(8),BLANKS         CLEAR MESSAGE AREA
         SR    R15,R15                 CLEAR ERR COND
         ST    R15,SAVER15             STORE IT ON SAVER15 AREA
**       MAP THE PARAMETERS PASSED USING A LOCAL DSECT
         L     R9,28(R1)               LOAD ADDRESS OF VARVAL
         USING IMSTIMES,R9             SET ADDRESABILITY
**       CHECK THAT SOME PARAMETER WAS PASSED TO THE EXIT
         L     R3,16(R1)               LOAD ADDRESS OF ARAYDIM
         L     R3,Ø(R3)                LOAD ARAYDIM VALUE
         MVC   MSGID(8),DBRC313H       PRELOAD ERROR MESSAGE
         LTR   R3,R3                   CHECK VALUE OF ARAYDIM
         BNP   ERRFOUND                IF ZERO, EXIT
**       CHECK THE LENGTH OF THE PARAMETERS PASSED TO THE EXIT
         L     R4,24(R1)               LOAD ADDRESS OF VARLEN
**       CHECK FOR THE TIMESTAMP LENGTH
**       AFTER, CHECK IF ADDITIONAL PARAMETERS WERE SPECIFIED
         L     R5,Ø(R4)                LOAD FIRST LEN VALUE
         LA    R6,12                   WILL CHECK FOR 12 CHARACTERS
         MVC   MSGID(8),DBRC313J       PRELOAD ERROR MESSAGE
         CR    R5,R6                   COMPARE REGISTERS
         BNE   ERRFOUND                IF NOT EQUAL, THEN ERROR
         MVC   MSGID(8),DBRC313I       PRELOAD ERROR MESSAGE
         BCTR  R3,Ø                    SUBTRACT 1 FROM ARAYDIM
         LTR   R3,R3                   SET CONDITION CODE
         BZ    ERRFOUND                IF NO MORE PARMS, ERROR
**       CHECK FOR THE ERROR FLAG LENGTH
```

```
**      AFTER, CHECK IF ADDITIONAL PARAMETERS WERE SPECIFIED
        L     R5,4(R4)                LOAD SECOND LEN VALUE
        LA    R6,1                    WILL CHECK FOR 1 CHARACTER
        MVC   MSGID(8),DBRC313K       PRELOAD ERROR MESSAGE
        CR    R5,R6                   COMPARE REGISTERS
        BNE   ERRFOUND                IF NOT EQUAL, THEN ERROR
        MVC   MSGID(8),DBRC313I       PRELOAD ERROR MESSAGE
        BCTR  R3,Ø                    SUBTRACT 1 FROM ARAYDIM
        LTR   R3,R3                   SET CONDITION CODE
        BZ    ERRFOUND                IF NO MORE PARMS, ERROR
**      CHECK FOR THE OUTPUT EXIT FIELD NAME VARIABLE
**      IF OKAY, RESET VARIABLE WITH NAME OF TIMESTAMP FIELD
**      AFTER, CHECK IF ADDITIONAL PARAMETERS WERE SPECIFIED
        L     R5,8(R4)                LOAD THIRD LEN VALUE
        LA    R6,8                    WILL CHECK FOR 8 CHARACTERS
        MVC   MSGID(8),DBRC313L       PRELOAD ERROR MESSAGE
        CR    R5,R6                   COMPARE REGISTERS
        BNE   ERRFOUND                IF NOT EQUAL, THEN ERROR
        L     R5,2Ø(R1)               LOAD ADDRESS OF VARNAMES
        MVC   ITSFLD(8),Ø(R5)         MOVE THE FIELD NAME
        MVC   MSGID(8),DBRC313I       PRELOAD ERROR MESSAGE
        BCTR  R3,Ø                    SUBTRACT 1 FROM ARAYDIM
        LTR   R3,R3                   SET CONDITION CODE
        BZ    ERRFOUND                IF NO MORE PARMS, ERROR
**      CHECK FOR THE OUTPUT EXIT CURSOR POSITION PARM
        L     R5,12(R4)               LOAD FOURTH LEN VALUE
        LA    R6,2                    WILL CHECK FOR 2 CHARACTERS
        MVC   MSGID(8),DBRC313M       PRELOAD ERROR MESSAGE
        CR    R5,R6                   COMPARE REGISTERS
        BNE   ERRFOUND                IF NOT EQUAL, THEN ERROR
***     PRELOAD THE ERROR FLAG WITH YES
        MVC   ITSFLAG(1),YESVALUE
***     CHECK THE YEARS.
        LA    R4,9Ø                   LOAD MINIMUM VALUE FOR YEARS
        LA    R5,99                   LOAD MAXIMUM VALUE FOR YEARS
        MVC   MSGID(8),DBRC313B       PRELOAD MESSAGE
        MVC   ITSPOS(2),YYPOS         SET POSITION WITHIN FIELD
        MVC   WRKYY,ITSYY             MOVE ITSYY TO WORK AREA
        NI    WRKYY+1,X'CF'           MASK YEAR1 SIGN
        PACK  PACKAREA(8),WRKYY(2)    PACK THE YEAR
        CVB   R7,PACKAREA             CONVERT TO BINARY
        CR    R7,R4                   CHECK IF LESS THAN LOWER LIMIT
        BM    ERRFOUND                IF LESS, GO TO ERRFOUND
        CR    R7,R5                   CHECK IF MORE THAN UPPER LIMIT
        BP    ERRFOUND                IF MORE, GO TO ERRFOUND
***     DIVIDE NUMBER OF YEARS BY 4 TO DETERMINE IF LEAP YEAR
        SR    R6,R6                   CLEAR R6 FOR DIVISION
        D     R6,FOUR                 DIVIDE YEARS BY FOUR
        LTR   R6,R6                   DO WE HAVE ANY LEAP YEAR ?
        BZ    ISALEAP                 YES, WE HAVE A LEAP YEAR
```

```
         MVC   MSGID(8),DBRC313C       PRELOAD MESSAGE FOR NON-LEAP
         LA    R5,365                  NO,  USE 365 DAYS FOR UPPER L
         B     CHECKDDD                GO TO CHECKDDD.
ISALEAP  LA    R5,366                  YES, USE 366 DAYS FOR UPPER L
         MVC   MSGID(8),DBRC313D       PRELOAD MESSAGE FOR LEAP DAYS
CHECKDDD EQU   *
***      CHECK DAYS
         LA    R4,1                    SET LOWER LIMIT
         MVC   ITSPOS(2),DDDPOS        SET POSITION WITHIN FIELD
         MVC   WRKDDD,ITSDDD           COPY DAYS TO WORK AREA
         NI    WRKDDD+2,X'CF'
         PACK  PACKAREA(8),WRKDDD(3)   PACK DDD TO PACKAREA
         CVB   R7,PACKAREA             CONVERT DDD TO BINARY
         CR    R7,R4                   CHECK IF LESS THAN LOWER LIMIT
         BM    ERRFOUND                IF LESS, GO TO ERRFOUND
         CR    R7,R5                   CHECK IF MORE THAN UPPER LIMIT
         BP    ERRFOUND                IF MORE, GO TO ERRFOUND
***      CHECK HOURS.
         SR    R4,R4                   SET LOWER LIMIT
         LA    R5,23                   SET UPPER LIMIT
         MVC   MSGID(8),DBRC313E       PRELOAD MESSAGE
         MVC   ITSPOS(2),HHPOS         SET POSITION WITHIN FIELD
         MVC   WRKHH,ITSHH             COPY HOURS TO WORK AREA
         NI    WRKHH+1,X'CF'
         PACK  PACKAREA(8),WRKHH(2)    PACK TO PACKAREA
         CVB   R7,PACKAREA             CONVERT TO BINARY
         CR    R7,R4                   CHECK IF LESS THAN LOWER LIMIT
         BM    ERRFOUND                IF LESS, GO TO ERRFOUND
         CR    R7,R5                   CHECK IF MORE THAN UPPER LIMIT
         BP    ERRFOUND                IF MORE, GO TO ERRFOUND
***      CHECK MINUTES
         SR    R4,R4                   SET LOWER LIMIT
         LA    R5,59                   SET UPPER LIMIT
         MVC   MSGID(8),DBRC313F       PRELOAD MESSAGE
         MVC   ITSPOS(2),MMPOS         SET POSITION WITHIN FIELD
         MVC   WRKMM,ITSMM             COPY HOURS TO WORK AREA
         NI    WRKMM+1,X'CF'
         PACK  PACKAREA(8),WRKMM(2)    PACK TO PACKAREA
         CVB   R7,PACKAREA             CONVERT TO BINARY
         CR    R7,R4                   CHECK IF LESS THAN LOWER LIMIT
         BM    ERRFOUND                IF LESS, GO TO ERRFOUND
         CR    R7,R5                   CHECK IF MORE THAN UPPER LIMIT
         BP    ERRFOUND                IF MORE, GO TO ERRFOUND
***      CHECK SECONDS
         SR    R4,R4                   SET LOWER LIMIT
         LA    R5,59                   SET UPPER LIMIT
         MVC   MSGID(8),DBRC313G       PRELOAD MESSAGE
         MVC   ITSPOS(2),SSPOS         SET POSITION WITHIN FIELD
         MVC   WRKSS,ITSSS             COPY HOURS TO WORK AREA
         NI    WRKSS+1,X'CF'
```

```
         PACK  PACKAREA(8),WRKSS(2)    PACK TO PACKAREA
         CVB   R7,PACKAREA             CONVERT TO BINARY
         CR    R7,R4                   CHECK IF LESS THAN LOWER LIMIT
         BM    ERRFOUND                IF LESS, GO TO ERRFOUND
         CR    R7,R5                   CHECK IF MORE THAN UPPER LIMIT
         BP    ERRFOUND                IF MORE, GO TO ERRFOUND
***      NO ERRORS WERE FOUND, CLEAR CONTROL VARIABLES AND EXIT
         MVC   MSGID(8),BLANKS         NO ERRORS FOUND, CLEAR MSGID
         MVC   ITSFLAG(1),NOVALUE      NO ERRORS FOUND, CLEAR FLAG
         B     EXITPROG
*------------------------***------***------------------------*
ERRFOUND LA    R15,8                  SET RETURN CODE TO ERR COND
         ST    R15,SAVER15
EXITPROG L     R13,SAVEAREA+4         RESTORES R13
         L     R15,SAVER15            RETURN CODE IN ERROR
         ST    R15,16(R13)            OVERLAY OLD R15 VALUE
         LM    R14,R12,12(R13)        RESTORES REGISTERS
         BR    R14                    RETURN TO START
SAVEAREA DS    18F
SAVER15  DS    1F
FOUR     DC    1F'4'
PACKAREA DS    1D
WRKZONE  DS    CL8
WRKYY    DS    CL2
WRKDDD   DS    CL3
WRKHH    DS    CL2
WRKMM    DS    CL2
WRKSS    DS    CL2
WRKT     DS    CL1
BLANKS   DC    CL8' '
***      NAME OF MESSAGES TO BE DISPLAYED IN THE EVENT OF ERROR
DBRC313A DC    CL8'DBRC313A'          INCORRECT TIMESTAMP FORMAT
DBRC313B DC    CL8'DBRC313B'          YEARS OUT OF RANGE
DBRC313C DC    CL8'DBRC313C'          DAYS OUT OF RANGE - NON-LEAP YEAR
DBRC313D DC    CL8'DBRC313D'          DAYS OUT OF RANGE - LEAP YEAR
DBRC313E DC    CL8'DBRC313E'          HOURS OUT OF RANGE
DBRC313F DC    CL8'DBRC313F'          MINUTES OUT OF RANGE
DBRC313G DC    CL8'DBRC313G'          SECONDS OUT OF RANGE
DBRC313H DC    CL8'DBRC313H'          ERROR 1 - NO PARAMETERS FOUND
DBRC313I DC    CL8'DBRC313I'          ERROR 2 - NOT ENOUGH PARAMETERS
DBRC313J DC    CL8'DBRC313J'          ERROR 3 - PARM 1 INCORRECT LENGTH
DBRC313K DC    CL8'DBRC313K'          ERROR 4 - PARM 2 INCORRECT LENGTH
DBRC313L DC    CL8'DBRC313L'          ERROR 5 - PARM 3 INCORRECT LENGTH
DBRC313M DC    CL8'DBRC313M'          ERROR 6 - PARM 4 INCORRECT LENGTH
***      CONSTANTS
YESVALUE DC    CL1'Y'
NOVALUE  DC    CL1'N'
YYPOS    DC    CL2'01'       POSITION IN FIELD WHERE YEAR BEGINS
DDDPOS   DC    CL2'03'       POSITION IN FIELD WHERE DDD  BEGINS
HHPOS    DC    CL2'06'       POSITION IN FIELD WHERE HH   BEGINS
```

```
MMPOS     DC    CL2'Ø8'        POSITION IN FIELD WHERE MM BEGINS
SSPOS     DC    CL2'1Ø'        POSITION IN FIELD WHERE SS BEGINS
          DSECT
MSGID     DS    CL8            MESSAGE ID
          DSECT
IMSTIMES  DS    ØCL23          EXPECTED DATA FORMAT
ITSYYDDD  DS    ØCL5           YEAR DAYS IN JULIAN FORMAT
ITSYY     DS    CL2            JULIAN YEAR
ITSDDD    DS    CL3            JULIAN DDD
ITSHMST   DS    ØCL7           HHMMSSH HOUR, MINUTES, SECS, 1ØTH/SEC
ITSHH     DS    CL2            HH HOURS
ITSMM     DS    CL2            MM MINUTES
ITSSS     DS    CL2            SS SECONDS
ITST      DS    CL1            1ØTHS OF SECONDS
ITSFLAG   DS    CL1            ERROR FLAG 'Y' OR 'N'
ITSFLD    DS    CL8            FIELD NAME
ITSPOS    DS    CL2            CURSOR POSITION WITHIN THE FIELD
          END
```

ISPF MESSAGE MEMBER

The ISPF messages used by this panel user exit need to be stored in member DBRC31 in a PDS allocated to the ISPMLIB DDNAME of your TSO procedure, or a LIBDEF statement could be used to map to it. The messages are as follows:

```
DBRC313A 'INCORRECT TIMESTAMP' .ALARM=YES
'TIMESTAMP FORMAT MUST BE YYDDDHHMMSST.  PLEASE TRY AGAIN'
DBRC313B 'YEAR OUT OF RANGE' .ALARM=YES
'YEAR MUST BE BETWEEN 9Ø AND 99.  PLEASE TRY AGAIN'
DBRC313C 'DAYS OUT OF RANGE' .ALARM=YES
'FOR A NON-LEAP YEAR, DAYS MUST BE BETWEEN ØØØ AND 365. TRY AGAIN'
DBRC313D 'DAYS OUT OF RANGE' .ALARM=YES
'FOR A LEAP YEAR, DAYS MUST BE BETWEEN ØØØ AND 366. TRY AGAIN'
DBRC313E 'HOURS OUT OF RANGE' .ALARM=YES
'HOURS MUST BE BETWEEN ØØ AND 23.  PLEASE TRY AGAIN'
DBRC313F 'MINUTES OUT OF RANGE' .ALARM=YES
'MINUTES MUST BE BETWEEN ØØ AND 59.  PLEASE TRY AGAIN'
DBRC313G 'SECONDS OUT OF RANGE' .ALARM=YES
'SECONDS MUST BE BETWEEN ØØ AND 59.  PLEASE TRY AGAIN'
DBRC313H 'DB31AITS ERROR 1' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : NO PARAMETERS WERE FOUND BY EXIT. CHECK'
DBRC313I 'DB31AITS ERROR 2' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : INSUFFICIENT PARAMETERS SPECIFIED. CHECK'
DBRC313J 'DB31AITS ERROR 3' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : INCORRECT LENGTH FOR PARAMETER 1. CHECK'
DBRC313K 'DB31AITS ERROR 4' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : INCORRECT LENGTH FOR PARAMETER 2. CHECK'
```

```
DBRC313L 'DB31AITS ERROR 5' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : INCORRECT LENGTH FOR PARAMETER 3. CHECK'
DBRC313M 'DB31AITS ERROR 6' .ALARM=YES
'DB31AITS PANEL EXIT ERROR : INCORRECT LENGTH FOR PARAMETER 4. CHECK'
```

CONCLUSION

I hope you find this information about ISPF panel user exits interesting and useful. There are many practical applications for them, and knowing how to work with them is a good ISPF trick to have.

Antonio L Salcedo (USA)

A multi-target copy edit macro

The purpose of this edit macro is to copy a block of lines in a single copy operation to multiple places thus reducing edit time significantly when copying the same lines repetitively. The only requirement is that the target locations must be identified uniquely either as all lines having the same keyword or by a specific line interval. The block will be copied to a position (A)fter or (B)efore the target location as selected or implied in a parameter to ZMCOPY.

The target range is from the current cursor position (outside the copy block) and extends to the indicated or implied limit. The 'copy from' range is indicated by C or CC line commands.

The parameters are:

1 Positional and required: keyword or numeric interval.

2 - 9 Optional (can be indicated in any order):

- Copy direction: down or up – the default is down.
- Target position: after or before – the default is after.
- Limit: a label – the defaults are .ZL if the direction is down and .ZF if the direction is up.

- Left boundary: numeric value – the default is left bounds.
- Right boundary: numeric value – the default is right bounds.
- Tracing CLIST: debug – the default is no CLIST tracing.
- Optional parameters: can be used for other find keywords like prefix, word, NX etc. If the keyword contains numeric characters and an optional parameter contains the word 'repeat' then the copy block is copied after/before every number of lines as indicated in 'keyword'.

If no parameters are supplied, help information will be shown.

```
PROC Ø DEBUG(NEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOCONLIST NOSYMLIST
ERROR DO
 SET &RET = &LASTCC
 RETURN
END
SET &RET = Ø
IF &STR(&DEBUG) = DEBUG THEN DO
 CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
END
IF &SYSISPF = &STR(NOT ACTIVE) THEN DO
 WRITE SORRY ONLY EXECUTABLE UNDER ISPF
 EXIT CODE(16)
END
IF &SYSNEST = NO THEN DO
 ISREDIT MACRO (KEYWORD,P1,P2,P3,P4,P5,P6,P7,P8,P9) NOPROCESS
END
ISPEXEC CONTROL ERRORS RETURN
IF &STR(&SYSNSUB(1,&KEYWORD)) = &STR() THEN DO
 SET &ZEDLMSG = ENTER SEARCH KEYWORD TO MULTICOPY AFTER.
 ISPEXEC SETMSG MSG(ISRZØØØ)
 ISPEXEC CONTROL DISPLAY LINE START(4)
 WRITE EDIT MACRO TO MULTI-COPY ONE OR MORE LINES AFTER OR BEFORE ALL
 WRITE LINES CONTAINING A SPECIFIC KEYWORD OR WITH A SPECIFIC INTERVAL.
 WRITE THE TARGET RANGE IS FROM THE CURRENT CURSOR POSITION (OUTSIDE
 WRITE +
 THE COPY-BLOCK) AND EXTENDS TO THE INDICATED OR IMPLIED LIMIT.
 WRITE THE COPY-FROM RANGE IS INDICATED BY C OR CC LINE COMMANDS.
 WRITE PARAMETERS:
 WRITE 1) POSITIONAL AND REQUIRED: KEYWORD OR NUMERIC INTERVAL.
 WRITE 2 - 9) OPTIONAL AND CAN BE INDICATED IN ANY ORDER:
 WRITE DIRECTION      : DOWN OR UP; DEFAULT DOWN
 WRITE POSITION       : AFTER OR BEFORE; DEFAULT IS AFTER
 WRITE LIMIT          : A LABEL; DEFAULT IS BOTTOM OR TOP
 WRITE LEFT BOUNDARY : NUMERIC VALUE; DEFAULT IS LEFT BOUNDS
```

```
 WRITE RIGHT BOUNDARY: NUMERIC VALUE; DEFAULT IS RIGHT BOUNDS
 WRITE TRACING CLIST : DEBUG; DEFAULT IS NO CLIST TRACING
 WRITE OPTIONAL PARAMETERS: CAN BE USED FOR OTHER FIND KEYWORDS
 WRITE LIKE PREFIX, WORD, NX ETC.
 WRITE IF KEYWORD CONTAINS NUMERIC AND AN OPTIONAL PARAMETER CONTAINS
 WRITE THE WORD "REPEAT" THEN THE COPY BLOCK IS COPIED AFTER/BEFORE
 WRITE EVERY NO OF LINES AS INDICATED IN "KEYWORD".
 ISPEXEC CONTROL DISPLAY LINE START(1)
 EXIT CODE(16)
END
IF &SYSNEST = NO THEN DO
 ISPEXEC CONTROL ERRORS RETURN
 ISREDIT PROCESS RANGE C CC
 IF &RET > Ø THEN DO
  SET &ZEDLMSG = ENTER COPY FROM AND TO LINE COMMANDS AS C OR CC.
  ISPEXEC SETMSG MSG(ISRZØØØ)
  EXIT CODE(16)
 END
END
SET &P = P
SET &Q = Ø
DO WHILE &Q < 9
 SET &Q = &Q + 1
 SET &PARMHIT = NO
 SET &D = &STR(&SYSNSUB(2,&&P&Q)
 SET &C = &STR(&SYSNSUB(1,&D))
 IF &STR(&SYSNSUB(1,&C)) NE &STR() THEN DO
  IF &AFTER = &STR() THEN DO
   IF &STR(&SYSNSUB(1,&C)) = AFTER | &STR(&SYSNSUB(1,&C)) = BEFORE +
   THEN DO
    SET &AFTER = &C
    SET &PARMHIT = YES
   END
  END
  IF &DIRECTN = &STR() THEN DO
   IF &STR(&SYSNSUB(1,&C)) = DOWN | &STR(&SYSNSUB(1,&C)) = UP THEN DO
    SET &DIRECTN = &C
    SET &PARMHIT = YES
   END
  END
  IF &LIMIT = &STR() THEN DO
   IF &SUBSTR(1:1,&STR(&SYSNSUB(1,&C))) = &STR(.) THEN DO
    SET &LIMIT = &C
    SET &RET = Ø
    ISREDIT (L) = LINENUM &LIMIT
    IF &RET > 4 THEN DO
     SET &ZEDLMSG = SPECIFY LIMIT AS VALID LABEL (.LABELNAME)
     ISPEXEC SETMSG MSG(ISRZØØØ)
     EXIT CODE(16)
    END
```

```
    SET &PARMHIT = YES
   END
  END
  IF &BEGCOL = &STR() THEN DO
   IF &DATATYPE(&STR(&SYSNSUB(1,&C))) = NUM THEN DO
    SET &BEGCOL = &C
    SET &PARMHIT = YES
   END
  END
  IF &ENDCOL = &STR() AND &PARMHIT NE YES THEN DO
   IF &DATATYPE(&STR(&SYSNSUB(1,&C))) = NUM THEN DO
    SET &ENDCOL = &C
    SET &PARMHIT = YES
   END
  END
  IF &REPEAT = &STR() AND &PARMHIT NE YES THEN DO
   IF &STR(&SYSNSUB(1,&C)) = REPEAT AND +
   &DATATYPE(&STR(&SYSNSUB(1,&KEYWORD))) = NUM THEN DO
    SET &REPEAT = YES
    SET &PARMHIT = YES
   END
  END
  IF &STR(&DEBUG) NE DEBUG AND &PARMHIT NE YES THEN DO
   IF &STR(&SYSNSUB(1,&C)) = DEBUG THEN DO
    SET &DEBUG = &STR(&C)
    SET &PARMHIT = YES
    CONTROL MSG NOFLUSH LIST CONLIST SYMLIST
   END
  END
  IF &OPT1 = &STR() AND &PARMHIT NE YES THEN DO
   SET &OPT1 = &STR(&SYSNSUB(1,&C))
   SET &PARMHIT = YES
  END
  IF &OPT2 = &STR() AND &PARMHIT NE YES THEN DO
   SET &OPT2 = &STR(&SYSNSUB(1,&C))
   SET &PARMHIT = YES
  END
  IF &OPT3 = &STR() AND &PARMHIT NE YES THEN DO
   SET &OPT3 = &STR(&SYSNSUB(1,&C))
   SET &PARMHIT = YES
  END
 END
END
ISREDIT (FROM) = LINENUM .ZFRANGE
ISREDIT (TO) = LINENUM .ZLRANGE
SET &LCYNO = &TO - &FROM + 1
IF &DIRECTN = &STR() THEN DO
 SET &DIRECTN = DOWN
END
SET &UE = &STR(<)
```

```
SET &ADD = &STR(+)
SET &PREV = NEXT
IF &DIRECTN = UP THEN DO
 SET &UE = &STR(>)
 SET &ADD = &STR(-)
 SET &PREV = PREV
END
IF &AFTER = &STR() THEN DO
 SET &AFTER = AFTER
END
IF &L = &STR() AND &DIRECTN = DOWN THEN DO
 ISREDIT (L) = LINENUM .ZLAST
 SET &BOTTOM = YES
END
IF &L = &STR() AND &DIRECTN = UP THEN DO
 SET &L = 1
END
SET &MAXLINE = &L
IF &MAXLINE >= &FROM AND &MAXLINE <= &TO THEN DO
 SET &ZEDLMSG = &STR(LIMIT CANNOT BE WITHIN FROM-TO RANGE)
 ISPEXEC SETMSG MSG(ISRZØØØ)
 EXIT CODE(16)
END
SET &HIT = Ø
IF &L = Ø THEN DO
 ISREDIT LINE_AFTER .ZFIRST = DATALINE MASKLINE
END
ISREDIT (ROW,COL) = CURSOR
IF &DIRECTN = UP AND &ROW >= &FROM THEN DO
 SET &ROW = &FROM - 1
END
IF &DIRECTN = DOWN AND &ROW <= &TO THEN DO
 SET &ROW = &TO + 1
END
SET &CPYOFS = Ø
IF (&ROW < &FROM AND &DIRECTN = DOWN) | +
(&ROW <= &FROM AND &DIRECTN = UP) THEN DO
 SET &CPYOFS = &LCYNO
END
SET &HITLINE = &ROW
SET &MAXLOOP = &L - &ROW    /* LIMIT LOOP IF RUN AWAY */
IF &MAXLOOP < Ø THEN DO
 SET &MAXLOOP = Ø - &MAXLOOP
END
SET &MAXLOOP = &MAXLOOP + 1
IF &REPEAT = YES AND &DIRECTN = UP THEN DO
 SET &ROW = &ROW - 1
END
SET &LNO = &ROW
DO WHILE &HITLINE &UE &MAXLINE AND &MAXLOOP > Ø
```

```
 SET &MAXLOOP = &MAXLOOP - 1
 ISREDIT CURSOR = &LNO Ø
 IF &DIRECTN = UP THEN DO
  SET &RET = Ø
  ISREDIT CURSOR = &EVAL(&LNO + 1)
 END
 IF &REPEAT NE YES THEN DO
  SET &RET = Ø
  ISREDIT SEEK &STR(&SYSNSUB(1,&KEYWORD)) &STR(&SYSNSUB(1,&PREV)) +
  &STR(&SYSNSUB(1,&BEGCOL)) &STR(&SYSNSUB(1,&ENDCOL)) +
  &STR(&SYSNSUB(1,&OPT1)) &STR(&SYSNSUB(1,&OPT2)) +
  &STR(&SYSNSUB(1,&OPT3))
 END
 IF (&RET = Ø AND &REPEAT NE YES) | +
 (&REPEAT = YES AND &RET = Ø) THEN DO
  SET &HIT = &HIT + 1
  SET &CPYFROM = &FROM
  ISREDIT (ROW,COL) = CURSOR
  SET &LNO = &ROW &ADD 1
  SET &HITLINE = &ROW
  SET &LINEA = &ROW
  SET &N = Ø
  DO WHILE &N < &LCYNO
   SET &N = &N + 1
   ISREDIT LINE_&AFTER &LINEA = DATALINE LINE &CPYFROM
   IF &DIRECTN = DOWN THEN DO
    SET &CPYFROM = &CPYFROM + 1
   END
   ELSE DO
    SET &CPYFROM = &CPYFROM + 2
   END
   SET &LINEA = &LINEA + 1
  END
  SET &FROM = &FROM + &CPYOFS
  IF &DIRECTN = DOWN THEN DO
   SET &MAXLINE = &MAXLINE + &LCYNO
   SET &HITLINE = &HITLINE + &LCYNO
   SET &LNO = &LNO + &LCYNO
  END
  IF &REPEAT = YES AND &DIRECTN = UP THEN DO
   SET &LNO = &LNO - &KEYWORD
  END
  IF &REPEAT = YES AND &DIRECTN = DOWN THEN DO
   SET &LNO = &LNO + &KEYWORD - 1
  END
 END
 ELSE DO
  SET &HITLINE = &MAXLINE
 END
END
```

```
SET &ZEDLMSG = NO OF COPIES PERFORMED: &HIT.
ISPEXEC SETMSG MSG(ISRZ000)
EXIT CODE(1)
```

Nils Plum (Denmark)

A clear screen utility for TSO/E

We have several in-house HLAPI programs around that perform automated sign-on to host ISPF applications from an OS/2 workstation in the wee hours of the morning. They grab some information from the host and port it back to the workstation. The HLAPI was written such that it must see the screens in the correct sequence otherwise the commands that the HLAPI issues to drive ISPF get all messed up (ie when the HLAPI issues a '6' it had better be at the ISPF main menu rather than the TSO ready prompt).

We had a program that did a clear screen; however, it broke somewhere and we didn't have the source (this function is not built into TSO as it is with other systems – Unix for example). The old version would break intermittently, which would leave the HLAPI dangling at a 'page prompt' after TSO issued the LISTBC on the user's behalf. The HLAPI would issue a '6' thinking it was at the main menu and the operation would fail. We needed a program that would work all the time and something we had the source for.

I looked in the *TSO System Programmers Guide* and found the SETLINE macro and used this, but when doing so we ran into the same problem with intermittent failures. In short, we had a timing problem between SVC93 (PRESS ENTER WHEN *** APPEARS) and SVC94 issued out of the clear screen utility. Both of these SVCs are issued asynchronously so there is no guarantee which will execute first. We played with this for some time and finally added a TCLEARQ to clear output buffers and all was well. The following simple program came out of our HLAPI episode and works well with TSO/ISPF . No special requirements on the BIND are required other than it is put somewhere the system can find it (eg LINKLIST library).


```
CLRSCRN   CSECT                          START
HOUS_KEEP DS    ØH                     LINKAGE
          STM   14,12,12(13)             "
          LR    12,15                    "
          USING CLRSCRN,12               "
          LA    11,SAVEAREA              "
          ST    13,4(11)                 "
          ST    11,8(13)                 "
          LR    13,11                    "
CLEAR     TCLEARQ OUTPUT                 CLEAR OUTPUT BUFFER
          STLINENO LINE=1,CLEAR=YES      CLEAR SCREEN
RETURN    DS    ØH                       RETURN TO CALLER
          L     13,SAVEAREA+4            "
          LM    14,12,12(13)             "
          SR    15,15                    "
          BR    14                       "
SAVEAREA  DS    18F                      SAVE
          END   CLRSCRN
```


Four IMS/DB utilities

INTRODUCTION

The four programs that comprise this set of IMS/DB utilities were designed to facilitate various automated IMS procedures. For some time I have utilized a variety of 'ye olde legacy modules', which manually plough through various control blocks to provide a variety of IMS/DB information. These were prone to becoming redundant if the layouts of any of the appropriate control blocks were altered as part of an IMS version change. They were also somewhat tricky to maintain.

Given that the information provided is integral to some automated functions I use, I decided to write new programs to get the data I needed. These modules use IBM macros to map control blocks, which should, hopefully, ensure that they do not become redundant. I also wrote a batch of programs to provide completely new IMS

information, to expand upon some existing automated procedures, and to help our operations staff.

The four programs included here all work upon the 'load control block then parse it' principal. I use them for various purposes. For example, I use these four programs and others to build a picture of how all our on-line systems inter-relate in terms of IMS and DB2 resources.

DBPCBLST

This program lists the database PCBs and corresponding (database level) PROCOPTS contained within the PSB(s) specified as input. This utility is handy for operations staff. For example, by executing it from a REXX EXEC, a handy TSO 'line command' can be created to list databases referenced by a PSB.

Scheduling staff who may not be familiar with IMS can then readily identify databases accessed by PSBs in a batch job. This easily enables them to see which database resources need to be attributed to the job within their scheduling system. Similarly, operators looking at the JCL of a job in SDSF can use this command to easily find out database information for scheduling purposes etc.

This program will accept parameter input or control card input from the SYSIN DD. The presence of a parameter will override SYSIN processing.

The program LOADs the PSBs specified in sequence, from the PSBLIB specified in the execution JCL/REXX. It maps the PSB using IMS macro DFSIPSB, and parses out the database PCBs and database level PROCOPTs contained therein. These are then written to the SYSPRINT DD.

It is worth noting that PL/I PSBs are laid out differently from COBOL or Assembler PSBs. If you look at the code labelled 'DBLOCATE' you will see how this is handled.

DBPCBLST should be assembled with attributes AMODE24 and RMODE24

```
     TITLE  'DBPCBLST - READ PSB LOAD MODULE AND OUTPUT DBS USED'
*********************************************************************
```

```
*    AUTHOR : MARTIN SYMMERS                                          *
*    PURPOSE:                                                         *
*    THIS PROGRAM WILL LIST THE DATABASES ACCESSED BY THE PSB(S)      *
*    PROVIDED AS INPUT.                                               *
*    INPUT CAN BE PROVIDED AS A PARM OR AS CONTROL CARD DATA.         *
*    NOTE THAT PARM INPUT WILL OVERRIDE ANY CONTROL CARD PRESENT.     *
*    MACROS USED:                                                     *
*    MVS -                                                            *
*    DCB, OPEN, CLOSE, LOAD                                           *
*    IMS -                                                            *
*    DFSIPSB                                                          *
*    REGISTER USAGE:                                                  *
*    RØ  - START ADDRESS OF LOADED PSB                                *
*    R1  - PARM POINTER                                               *
*    R2  - PARM LENGTH                                                *
*    R3  - USED AS POINTER WITHIN LOADED PSB                          *
*    R4  - USED TO INCREMENT R3 WITH OFFSET TO FIRST DATABASE PCB     *
*    R7  - DATABASE PCB ADDRESS                                       *
*    R8  - SAVE RETURN CODE - PGM WILL EXIT WITH LAST NON-ZERO CODE   *
*    R9  - PARM COUNTER - USED TO DECIDE WHETHER WE'VE HAD ANY INPUT  *
*    R1Ø - DATABASE ADDRESS POINTER                                   *
*    R11 - PL1 DOPE VECTOR / DB POINTER                               *
*    R12 - ADDRESSABILITY                                             *
*    R13 - SAVEAREA                                                   *
*    R14 - LINKAGE                                                    *
*    R15 - ENTRY POINT ADDRESS / RETURN CODE                          *
*    OUTPUT:                                                          *
*    COLUMNS:                                                         *
*    1-8: PSBNAME   9-12: SPACES   13-2Ø: DBDNAME   21-24: SPACES     *
*    25-28: PROCOPT   29-8Ø: SPACES                                   *
*    RETURN CODES:                                                    *
*     Ø - ALL MODULES INPUT PROCESSED SUCCESSFULLY                    *
*     4 - PROBLEM LOADING ONE OR MORE MODULES INPUT                   *
*     8 - AT LEAST ONE MODULE INPUT CONTAINED NO DATABASE PCBS        *
*    12 - AT LEAST ONE MODULE INPUT CONTAINED AN INVALID DB PCB AND   *
*         MAY NOT BE A VALID PSB                                      *
*    16 - NO INPUT SUPPLIED                                           *
*    NOTES:                                                           *
*    PROGRAM WILL EXIT WITH RETURN CODE FROM LAST 'NO ZERO' SITUATION *
*    ENCOUNTERED. MESSAGE WILL BE PUT TO SYSPRINT FROM ANY 'ERROR'    *
*    OCCURRENCE.                                                      *
*    PL/I PSBS HAVE A DIFFERENT LAYOUT THAN ASSEMBLER / COBOL MODULES *
*    - THIS IS BECAUSE OF THE INCLUSION OF PL/I DOPE VECTORS.         *
*    AS A RESULT, DATABASE PCB INFO IS FOUND AT DIFFERENT OFFSETS     *
*    DEPENDING ON PSB TYPE.                                           *
**********************************************************************
DBPCBLST CSECT
         BAKR  R14,RØ               SAVE REGS INTO LINKAGE STACK
         LR    R12,R15              ADDRESSABILITY
         USING DBPCBLST,R12         ESTABLISH ADDRESSABILITY
```

```
         LA    R13,SAVEAREA          SET UP SAVE AREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         L     R1,Ø(R1)              GET PARM
         XR    R2,R2                 ZERO
         XR    R3,R3                 WORK
         XR    R4,R4                 REGISTERS
         XR    R7,R7
         XR    R8,R8
         XR    R9,R9
         XR    R1Ø,1Ø
         XR    R11,11
         XR    R15,R15
         MVC   PSBIN,BLANKS          INITIALIZE
         MVC   PSBNAME,BLANKS        VARIABLES
         MVI   WORKTODO,1            INIT WORKTODO LOOP FLAG
         LH    R2,Ø(R1)              GET PARM LENGTH
         C     R2,=F'Ø'              GOT NO PARM?
         BE    FILEOPEN              NO PARM - GOTO OPEN FILES
         MVI   WORKTODO,Ø            PARM INPUT NO NEED TO LOOP
         BCTR  R2,RØ                 DECREMENT 1 FROM R2
         EX    R2,GETPSB             GET PSBIN (PARM)
         MVC   PSBNAME,PSBIN         MOVE PSBIN TO PSBNAME
FILEOPEN OPEN  (PSBLIB,(INPUT))      OPEN PSB LIBRARY
         OPEN  (SYSPRINT,(OUTPUT))   OPEN OUTPUT FILE
         CLC   PSBNAME,BLANKS        HAVE WE GOT A PSB?
         BNE   LOADUP                YES - GO AND LOAD IT UP
         OPEN  (SYSIN,(INPUT))       OTHERWISE OPEN SYSIN FILE
GETREC   GET   SYSIN,CARD            GET A SYSIN RECORD
         CLC   PSBIN,BLANKS          IS IT BLANK?
         BE    NOPARM                YES - GOTO NOPARM ROUTINE
         MVC   PSBNAME,PSBIN         NO - GET PSB NAME
         A     R9,=F'1'              INCREMENT PARM COUNTER
LOADUP   LOAD  EPLOC=PSBNAME,DCB=PSBLIB,ERRET=PSBERR    LOAD UP PSB (RØ
*                                    POINTS TO START ADDRESS)
PSBTEST  LR    R3,RØ                 SAVE START ADDRESS POINTER
         USING PPSB,R3               ESTABLISH PSB ADDRESSABILITY
         CLC   PPSBDBOF,=X'FFFF'     TEST FIRST DBPCB OFFSET. IF
*                                    X'FFFF', WE HAVE NO
         BE    NODBD                 DATABASE PCBS IN THIS PSB
         MVI   PL1,Ø                 INITIALIZE PL/I PSB FLAG
         TM    PPSBCODE,PPSBPL1      CHECK IF THIS IS A PL/I PSB
         BNO   PSBREAD               NO, GO READ IT
         MVI   PL1,1                 YES - SO SET FLAG
PSBREAD  MVC   DBDNAME,BLANKS        BLANK DBDNAME
         MVC   PROCOPT,BLANKPRO      BLANK PROCOPT
         LA    R1Ø,PPSBLIST          LOAD ADDR OF START OF PCB LIST
         AH    R1Ø,PPSBDBOF          ADD OFFSET FIRST DB PCB PTR
DBLOCATE L     R7,Ø(,R1Ø)            GET DBPCB ADDRESS
         LTR   R7,R7                 CHECK FOR ZEROS - INVALID ADDR
         BZ    PCBERR                GOT ZEROS - GOTO ERROR ROUTINE
```

```
         USING PBPCB,R7              AND ESTABLISH ADDRESSABILITY
         CLI   PL1,1                 IS THIS IS A PL/I PSB??
         BNE   GETDBD                NO, GO GET DBD INFO
         L     R11,Ø(,R7)            Y - DBPCB ADDR IS  PL/I DOPE ADDR
         LA    R7,Ø(R11,R7)          PUT IT IN R11, THEN ADD TO R7 TO
*                                    GET ACTUAL DBDNAME ADDRESS
GETDBD   MVC   DBDNAME,PBPCBDBD     GET DBD NAME...
         MVC   PROCOPT,PBPCBPRO      AND PROCOPT
OUTPUT   PUT   SYSPRINT,PSBOUT       OUTPUT TO SYSPRINT
         TM    Ø(R1Ø),X'8Ø' TEST FOR X'8Ø' IN FIRST BYTE - LAST DB FLAG
         BO    CHECK                 ON A MATCH GO TO CHECK
         A     R1Ø,=F'4'             ELSE INCREMENT 1Ø BY +4 - NEXT PTR
         B     DBLOCATE              AND GO BACK & FIND NEXT PCB
PSBERR   C     R8,=F'4'              MODULE LOAD ERROR
         BNL   ERRMSG1
         LA    R8,4
ERRMSG1  MVC   LDERRPSB,PSBNAME
         PUT   SYSPRINT,LDERROR
         B     CHECK
NODBD    C     R8,=F'8'              NO DBDS IN PSB ERROR
         BNL   ERRMSG2
         LA    R8,8
ERRMSG2  MVC   ERRORPSB,PSBNAME
         PUT   SYSPRINT,ERROR
         B     CHECK
PCBERR   C     R8,=F'12'             INVALID DB PCB ADDRESS ERROR
         BNL   ERRMSG3
         LA    R8,12
ERRMSG3  MVC   PCBERPSB,PSBNAME
         PUT   SYSPRINT,DBPCBERR
         B     CHECK
NOPARM   C     R9,=F'Ø'              IF R9 = F'Ø' WE'VE HAD
         BNE   CLOSIN                NEITHER PARM NOR
         PUT   SYSPRINT,PARMERR      CONTROL CARD
         LA      R8,16               INPUT - ERROR
         B        CLOSIN
CHECK    CLI   WORKTODO,1            CHECK IF WORKTODO FLAG SET
         BNE    CLEANUP              NO? - LET'S GET OUT OF HERE
         DELETE EPLOC=PSBNAME        YES - RELEASE LOADED PSB
         B     GETREC                AND GO GET NEXT CARD
CLOSIN   CLOSE (SYSIN)               CLOSE SYSIN DATASET
CLEANUP  DELETE EPLOC=PSBNAME        RELEASE LOADED PSB
         CLOSE (PSBLIB)              CLOSE FILES
         CLOSE (SYSPRINT)            CLOSE FILES
         LR    R15,R8                LOAD SAVED RETURN CODE
         PR    ,                     EXIT WITH RC
GETPSB   MVC   PSBIN(Ø),2(R1)        GET PSBNAME FROM R1 + 2
         LTORG
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,LRECL=8Ø,RECFM=FB
PSBLIB   DCB   DDNAME=PSBLIB,DSORG=PS,MACRF=R
```

```
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,MACRF=GM,EODAD=NOPARM
SAVEAREA DS    18F
WORKTODO DS    CL1
PL1      DS    CL1
CARD     DS    ØCL8Ø
PSBIN    DS    CL8
FILLER   DC    CL72' '
BLANKS   DC    CL8' '
BLANKPRO DC    CL4' '
ERROR    DS    ØCL8Ø
ERRORPSB DS    CL8
ERRORMSG DC    CL37' - NO DATABASES CONTAINED IN THIS PSB'
ERRORFIL DC    CL35' '
LDERROR  DS    ØCL8Ø
LDERRPSB DS    CL8
LDERRMSG DC    CL28' - ERROR LOADING THIS MODULE'
LDERRFIL DC    CL44' '
PARMERR  DS    ØCL8Ø
PRMERMSG DC    CL34'ERROR - NO PARAMETER DATA SUPPLIED'
PRMERFIL DC    CL46' '
DBPCBERR DS    ØCL8Ø
PCBERPSB DS    CL8
PCBERMG1 DC    CL32' - INVALID DB PCB ADDRESS FOUND:'
PCBERMG2 DC    CL3Ø' MODULE MAY NOT BE A VALID PSB'
PCBERFIL DC    CL1Ø' '
PSBOUT   DS    ØCL8Ø
PSBNAME  DS    CL8
         DC    CL4' '
DBDNAME  DS    CL8
         DC    CL4' '
PROCOPT  DS    CL4
         DC    CL52' '
         DFSIPSB PPSBASE=Ø
         DFSIPSB PPCBASE=Ø
         END
```

DBRELLST

This program will return any databases 'related' to the database(s) supplied as input, ie indices, logically related databases, etc. It can be used to automatically build associated groups of databases for purposes such as recovery, reorganization etc. For example, I have an automated procedure which uses this data to automatically 'PROHIBIT AUTHorization' on any databases related to the one being reorganized.

This program will accept parameter input or control card input from the SYSIN DD. The presence of a parameter will override SYSIN

processing. The program LOADs the DBD(s) input, from the DBDLIB library specified in the execution JCL/REXX.

The IMS macro IDBD is used to map the DBD module. Then the 'external database reference table' is located, which is looped through to extract all external DBDs which are output to the SYSPRINT DD.

DBRELLST should be assembled with attributes AMODE24 and RMODE24

```
DBRELLST  TITLE 'IDENTIFY RELATED DATABASES'
*********************************************************************
*   FUNCTION: IDENTIFY DATABASES WHICH ARE RELATED TO THE DATABASE(S) *
*   INPUT, EG INDICES, LOGICAL RELATIONSHIPS ETC                      *
*   MACROS USED:                                                      *
*   MVS - OPEN, CLOSE, GET, PUT, USING                                *
*   IMS - IDBD                                                        *
*   REGISTER USAGE:                                                   *
*   RØ  - START ADDRESS OF LOADED DBD                                 *
*   R1  - PARM                                                        *
*   R2  - PARM HANDLING                                               *
*   R3  - COUNT OF CONTROL CARDS PROCESSED                            *
*   R4  - STORE EXTERNAL DB REFERENCE COUNT                           *
*   R5  - NOT USED                                                    *
*   R6  - ADDRESSABILITY WITHIN LOADED DBD                            *
*   R7  - ADDRESSABILITY WITHIN LOADED DBD                            *
*   R8  - STORE RETURN CODE                                           *
*   R9  - NOT USED                                                    *
*   R1Ø - NOT USED                                                    *
*   R11 - NOT USED                                                    *
*   R12 - BASE ADDRESSABILITY                                         *
*   R13 - SAVE AREA                                                   *
*   R14 - RETURN ADDRESS                                              *
*   R15 - RETURN CODE                                                 *
*  RETURN CODES:                                                      *
*   Ø - NO COMPRESSION ON ANY DATABASES SPECIFIED ON CONTROL CARD      *
*   4 - NO CONTROL DATA SUPPLIED                                      *
*   8 - ERROR LOADING DBD                                             *
*********************************************************************
DBRELLST CSECT
         BAKR  R14,RØ
         LR    R12,R15
         USING DBRELLST,R12          BASE ADDRESSABILITY
         LA    R13,SAVEAREA          SET UP SAVE AREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         L     R1,Ø(R1)              GET PARM
         SR    R2,R2                 ZERO WORK REGISTERS
         SR    R3,R3                 ZERO WORK REGISTERS
         SR    R4,R4
```

```
         SR    R6,R6
         SR    R7,R7
         SR    R8,R8
         MVI   PARMFLAG,Ø            INIT FLAG THAT INDICATES PARM DATA
         LH    R2,Ø(R1)              GET PARM LENGTH
         LTR   R2,R2                 GOT PARM?
         BZ    FILEOPEN              NO: GOTO OPEN FILES ROUTINE
         MVI   PARMFLAG,1            WORK WITH PARM INPUT -
*                                    NO NEED TO LOOP THRU SYSIN
         MVC   DBNAME,INITDBD        INITIALIZE DBD VARIABLE
         BCTR  R2,RØ                 DECREMENT 1 FROM R2
         EX    R2,GETDBD             GET DBDNAME FROM PARAMETER
FILEOPEN OPEN  (DBDLIB,(INPUT))      OPEN FILES
         OPEN  (SYSPRINT,(OUTPUT))
         CLI   PARMFLAG,1            USING PARM DATA?
         BE    LOADUP                YES, SKIP SYSIN STUFF
         OPEN  (SYSIN,(INPUT))
GETREC   GET   SYSIN,CARD            GET CONTROL CARD
         CLC   CARD,BLANKS           GOT NO DATA ON IT?
         BE    SETCODE               NO! JUMP TO SET BAD CODE
         A     R3,=F'1'              OTHERWISE INCREMENT COUNTER
LOADUP   LOAD  EPLOC=DBDNAME,DCB=DBDLIB,ERRET=LOADERR          LOAD DBD
         LR    R6,RØ                 SAVE START ADDRESS PTR
         USING DBDDSECT,R6           ESTABLISH ADDRESSABILITY
         MVC   TARGET,DBDNAME        MOVE DB TO OUTPUT VAR
         PUT   SYSPRINT,BLANKS       PRINT BLANK LINE
         PUT   SYSPRINT,TARGETDB     PRINT HEADING
GETPREFX L     R7,APREFIX            LOAD PREFIX SECTION...
         USING PREFIX,R7             & USE FOR ADDRESSABILITY
         LH    R4,PRENODBD           SAVE EXT DB COUNT IN R4
         LTR   R4,R4                 IS EXTERNAL DB COUNT Ø
         BNZ   LOADEXDB              NO-GO AND LOAD EX DB TBL
         MVC   RELATE,NONE           Y-FILL IN 'NONE' MESSAGE
         PUT   SYSPRINT,RELATEDB     ...AND PRINT IT
         B     UNLOCK                GO & RELEASE LOADED DBD
LOADEXDB EQU *
         DROP  R7                    ELSE DROP PFX ADDRESSING
         L     R7,AEXTDBD            LOAD EXT DBD TABLE
GETEXDBD EQU *
         USING EXTDBD,R7             EXT DBD TABLE ADDRESSING
         MVC   RELATE,EXTDBNM        GET EXTERNAL DB NAME
         PUT   SYSPRINT,RELATEDB     ...AND PRINT IT OUT
         LA    R7,EXDBDSZ(R7)        LOAD ADDR OF EXT DBD
         BCT   R4,GETEXDBD           & LOOP TILL ALL EXT DB
*                                    REFERENCES PROCESSED
UNLOCK   DELETE EPLOC=DBNAME         RELEASE LOADED DBD
         CLI   PARMFLAG,1            USING PARM DATA?
         BE    XITPOINT              Y- NO NEED TO LOOP BACK
         B     GETREC                N-GET NEXT CONTROL CARD
LOADERR  LA    R8,8                  ERROR LOADING DBD - SET CC
```

```
         B     CLSEFILE              AND GO LEAVE
SETCODE  LTR   R3,R3                 CHECK IF CONTROL CARD CNT IS Ø
         BNZ   CLSEFILE              NO - GO TO XITPOINT
         LA    R8,4                  YES - SET BAD CODE
CLSEFILE CLI   PARMFLAG,1            USING PARM INPUT
         BE    XITPOINT              YES - GO TO XITPOINT
         CLOSE SYSIN                 NO - CLOSE SYSIN FILE
XITPOINT CLOSE DBDLIB                CLOSE FILES
         CLOSE SYSPRINT
         LR    R15,R8                LOAD SAVED RETURN CODE
         PR    ,                     BYE BYE
GETDBD   MVC   DBDNAME(Ø),2(R1)      GET DBDNAME
         LTORG
DBDLIB   DCB   DDNAME=DBDLIB,DSORG=PS,MACRF=GM,EODAD=CLSEFILE
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,MACRF=GM,EODAD=CLSEFILE
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,EODAD=CLSEFILE
SAVEAREA DS    18F
PARMFLAG DS    CL1
BLANKS   DS    ØCL8Ø
INITDBD  DC    CL8' '
         DC    CL72' '
CARD     DS    ØCL8Ø
DBDNAME  DS    CL8
         DS    CL72
TARGETDB DS    ØCL8Ø
         DC    CL21'DATABASES RELATED TO '
TARGET   DS    CL8
         DC    CL1' '
         DC    CL1':'
         DC    CL1' '
         DC    CL49' '
RELATEDB DS    ØCL8Ø
RELATE   DS    CL8
         DC    CL72' '
NONE     DC    CL8'NONE    '
         IDBD
         END
```

DBDYNLST

This program will return the DD and dataset names of the datasets which comprise the database(s) input, based on IMS dynamic allocation modules. The return code for this program indicates the number of datasets which comprise the (last) database processed.

This can be used to automate delete/define processes. For example, I store our VSAM cluster definitions by DDNAME in a PDS – the data

from DBDYNLST can be used to automatically fetch these based on the database specified. The return code is used as an easy method of determining iterations of functions which are dependent on how many datasets comprise a database. Similarly, it can be used to conditionally include extra JCL for databases containing more than one dataset groups for example.

This program will accept parameter input or control card input from the SYSIN DD. The presence of a parameter will override SYSIN processing. The program LOADs the DYNALLOC modules for the database(s) input from the DYNALLOC library specified in the execution JCL/REXX code.

The IMS macro DFSMDA is initially used to map the module to find dynamic allocation request block pointers. These point into storage which is mapped by MVS macro IEFZB4D0. This is used to locate dynamic allocation 'text objects'. Macro IEFZB4D2 (dynamic allocation key definition table) is then referenced to determine what these objects are, eg DD name, dataset name etc.

When the DD name (and dataset name) are identified they are copied into the relevant output fields, and written to SYSPRINT. Note that the dynamic allocation module for the IMS monitor is bypassed as it redefines a Request Block pointer count field, which is used to control a loop within this program.

DBDYNLST should be assembled with attributes AMODE24 and RMODE24

```
DBDYNLST   TITLE 'DATABASE DATASET INFO FROM DYNALLOC MODULE'
*********************************************************************
*   PROGRAM: DBDYNLST                                               *
*   FUNCTION: IDENTIFY DATABASE DD NAME & DATASET NAME              *
*   MACROS USED:                                                    *
*   MVS - OPEN, CLOSE, GET, PUT, USING, IEFZB4D0, IEFZB4D2          *
*   IMS - DFSMDA                                                    *
*   REGISTER USAGE:                                                 *
*   R1  - MVS                                                       *
*   R2  - SAVE PARMETER; INDEXING DYNALLOC REQUEST BLOCK POINTER    *
*   R3  - PARMETER LENGTH                                           *
*   R4  - BASE ADDRESSABILITY - DYNALLOC MODULE                     *
*   R5  - HOLDS NUMBER OF REQUEST BLOCK POINTERS                    *
*   R6  - HOLDS REQUEST BLOCK POINTER                               *
*   R7  - BASE ADDRESSABILITY - DFSMS REQUEST BLOCK                 *
```

```
*   R8  - REQUEST BLOCK TEXT UNIT POINTER                              *
*   R9  - REQUEST BLOCK TEXT UNIT ADDRESSABILITY                       *
*   R1Ø - TEXT DATA                                                    *
*   R11 - INDEXING TEXT UNIT POINTER                                   *
*   R12 - BASE ADDRESSABILITY                                          *
*   R13 - SAVEAREA                                                     *
*   R14 - MVS                                                          *
*   R15 - MVS                                                          *
*   RETURN CODES:                                                      *
*   VALUE OF RETURN CODE CORRESPONDS TO NUMBER OF DATASETS THAT        *
*   COMPRISE (LAST) DATABASE SPECIFIED.                                *
*   ABEND CODES:                                                       *
*   U1875 - ERROR LOADING DBD OR NO DBD SPECIFIED.                     *
**********************************************************************
DBDYNLST CSECT
         BAKR  R14,RØ
         LR    R12,R15
         USING DBDYNLST,R12          BASE ADDRESSABILITY
         LA    R13,SAVEAREA          SET UP SAVE AREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         L     R2,Ø(R1)              SAVE PARM
         SR    R3,R3                 ZERO WORK REGISTERS
         SR    R4,R4
         SR    R5,R5
         SR    R6,R6
         SR    R7,R7
         SR    R8,R8
         SR    R9,R9
         SR    R1Ø,R1Ø
         SR    R11,R11
         MVI   SYSINFLG,1          INTIALIZE 'GOT SYSIN?' FLAG TO 'YES'
GET_PARM LH    R3,Ø(R2)             CHECK IF WE HAVE PARM INPUT
         LTR   R3,R3
         BZ    OPEN_FILES          NO? GO OPEN FILES
         MVI   SYSINFLG,Ø          YES - SET 'GOT SYSIN' TO 'NO'
         MVC   DBNAME,CLEARDBD     CLEAR DBNAME VARIABLE
         BCTR  R3,RØ               DECREMENT 1 FROM R3 FOR EX
         EX    R3,MOVE_PARM        GET PARM
OPEN_FILES OPEN (DYNALLOC,(INPUT)) OPEN DYNALLOC DD
         OPEN  (SYSPRINT,(OUTPUT)) OPEN SYSPRINT DD
         CLI   SYSINFLG,Ø          HAVE WE 'GOT SYSIN' ??
         BE    LOAD_DYNALLOC_MODULE  NO - GO LOAD MODULE
         OPEN  (SYSIN,(INPUT))     YES - OPEN SYSIN FILE
GET_SYSIN_CARD GET SYSIN,CARD      GET SYSIN CARD
         MVC   DBNAME,DBSYSIN      GET DATABASE FROM SYSIN CARD
         CLI   SYSINFLG,2          CHECK IF THIS IS FIRST CARD
         BE    LOAD_DYNALLOC_MODULE  NO - GO LOAD MODULE
         CLC   DBNAME,CLEARDBD     YES - CHECK WE HAVE A DB
         BE    EXIT_WITH_ABEND     NO DB, MEANS NO INPUT: ABEND
         MVI   SYSINFLG,2          SET FLAG TO INDICATE WE HAVE
```

```
*                                  AT LEAST ONE DB ON SYSIN
LOAD_DYNALLOC_MODULE MVI LASTPASS,Ø  INTIALIZE 'LAST PASS' TO 'NO'
         LOAD  EPLOC=DBNAME,DCB=DYNALLOC,ERRET=EXIT_WITH_ABEND  LOAD DB
         LR    R4,RØ                R4 POINTS TO START OF MODULE
         USING DYNALMBR,R4          ESTABLISH ADDRESSABILITY
         LH    R5,DYNALLPN          GET NO OF RB POINTERS
         ST    R5,DSCOUNT           STORE THEM (DATASET COUNT)
         LA    R6,DYNRBPTR          GET PTR TO 1ST DFSMS REQUEST BLOCK
         SR    R2,R2                ZERO R2 (TO USE AS REQ BLOCK INDEX)
GET_RB_POINTER LA R7,Ø(R2,R6)      GET PTR TO CURRENT REQUEST BLOCK
         L     R7,Ø(R7)             LOAD REQUEST BLOCK
         USING S99RB,R7             ...      AND ESTABLISH ADDRESSABILITY
         L     R7,S99TXTPP          LOAD TEXT UNIT POINTER LIST
         USING S99TUPL,R7           ...MAP IT
         LA    R8,S99TUPTR          GET 1ST TEXT UNIT POINTER
LOCATE_TEXT_UNIT LA R9,Ø(R11,R8)   GET CURRENT TEXT UNIT POINTER
         TM    Ø(R9),X'8Ø'          CHECK IF THIS IS LAST POINTER
         BZ    PARSE_TEXT_UNIT      NO - GO READ TEXT UNIT
         MVI   LASTPASS,1           YES - SET 'LASTPASS' TO YES
PARSE_TEXT_UNIT L R9,Ø(R9)         LOAD TEXT UNIT
         USING S99TUNIT,R9          ...AND ESTABLISH ADDRESSABILITY
         MVC   DYNVAR,CLEARDYN      CLEAR DYNALLOC TEXT VARIABLE
         LH    R1Ø,S99TULNG         GET LENGTH OF TEXT
         LTR   R1Ø,R1Ø              GOT TEXT?
         BZ    OUTPUT               NO - GO AND OUTPUT WHAT WE HAVE
         BCTR  R1Ø,RØ               DECREMENT 1 FROM R1Ø FOR EX
         EX    R1Ø,MOVE_VARIABLE    GET DYNALLOC TEXT OBJECT
DD_CHECK CLC   S99TUKEY,=AL2(DALDDNAM)  CHECK IF TEXT OBJECT IS DD
         BNE   DS_CHECK             NO - GO SEE IF IT IS A D/S
         MVC   DDNAME,DYNVAR        YES - STORE DDNAME
         CLC   DDNAME,IMSMON        IS IT IMS MONITOR MODULE?
         BNE   MOVE_REQ_BLOCK_PNTR  NO - GO UPDATE POINTER
         PUT   SYSPRINT,IMSMONMG    YES - POST MESSAGE
         B     RELEASE_DYNALLOC_MODULE  ...AND GO DUMP IT
DS_CHECK CLC   S99TUKEY,=AL2(DALDSNAM)  CHECK IF TEXT OBJECT IS D/S
         BNE   MOVE_REQ_BLOCK_PNTR      NO - GO UPDATE POINTER
         MVC   DSNAME,DYNVAR            YES - STORE DATASET NAME
         B     OUTPUT                   ...AND GO OUTPUT
MOVE_REQ_BLOCK_PNTR A R11,=F'4'         INCREM  ENT PTR INDEX BY 4
         CLI   LASTPASS,1           CHECK IF THIS IS LAST PASS THRU LOOP
         BNE   LOCATE_TEXT_UNIT     NO - GO FIND NEXT TEXT UNIT
OUTPUT   PUT   SYSPRINT,OUTLINE     OUTPUT
         LA    R2,DYNLN          UPDATE RB IDX WITH LENGTH OF DS SECTION
         SR    R7,R7                ZERO R7 - RB BLOCK
         SR    R11,R11              ZERO R11 - TEXT UNIT INDEX
         BCT   R5,GET_RB_POINTER    GO GET NEXT RB POINTER
RELEASE_DYNALLOC_MODULE DELETE EPLOC=DBNAME  RELEASE LOADED
*                                  DYNALLOC MBR
         CLI    SYSINFLG,Ø          ARE WE USING SYSIN DATA?
         BNE    GET_SYSIN_CARD      YES - GO GET NEXT CARD
```

```
CLOSE_FILES CLI SYSINFLG,Ø          ARE WE USING SYSIN DATA?
         BE    EXIT_POINT          NO - GO TO EXIT_POINT
         CLOSE SYSIN               YES - CLOSE SYSIN FILE
EXIT_POINT CLOSE SYSPRINT          CLOSE SYSPRINT FILE
         CLOSE DYNALLOC            CLOSE DYNALLOC FILE
         L     R15,DSCOUNT         LOAD DATASET COUNT INTO RC REG
         PR    ,                   DEPART...
EXIT_WITH_ABEND CLOSE DYNALLOC     CLOSE DYNALLOC FILE
         ABEND 1875                SUFFER AN ABNORMAL END
MOVE_PARM MVC DBNAME(Ø),2(R2)      GET PARM
MOVE_VARIABLE MVC DYNVAR(Ø),S99TUPAR  GET DYNALLOC TEXT OBJECT
         LTORG
DYNALLOC DCB   DNAME=DYNALLOC,DSORG=PS,MACRF=GM,EODAD=CLOSE_FILES
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,EODAD=CLOSE_FILES
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,MACRF=GM,EODAD=CLOSE_FILES
SAVEAREA DS    18F
IMSMON   DC    CL8'IMSMON '
IMSMONMG DS    ØCL8Ø
         DC    CL36'IMS MONITOR DYNALLOC MEMBER BYPASSED'
         DC    CL44' '
CLEARDBD DC    CL8' '
CLEARDYN DC    CL44' '
DYNVAR   DS    CL44
DSCOUNT  DS    F
SYSINFLG DS    CL1
LASTPASS DS    CL1
CARD     DS    ØCL8Ø
DBSYSIN  DS    CL8
         DS    CL72
OUTLINE  DS    ØCL8Ø
         DC    CL5'DBD: '
DBNAME   DS    CL8
         DC    CL5' DD: '
DDNAME   DS    CL8
         DC    CL6' DSN: '
DSNAME   DS    CL44
         DC    CL4' '
         DFSMDA FUNC=DSECT   .
         IEFZB4DØ
         IEFZB4D2
         END
```

DBCMPCHK

This program is used to determine whether the database(s) provided as input is/are compressed or not. Of any database(s) input, it will output those which are compressed. It will also issue a return code of 4 if it finds one or more compressed databases.

I primarily use this with a single database supplied as a parameter. The return code can be used to automatically include JCL/REXX code to carry out functions pertinent to compressed databases, such as automatically building control cards to expand data, when creating database extracts. This allows uniform JCL/REXX/ISPF skeletons to be created for both compressed and uncompressed databases.

This program will accept parameter input or control card input from the SYSIN DD. The presence of a parameter will override SYSIN processing. The program LOADs the DBD(s) from the DBDLIB library specified in the execution JCL/REXX code. The IMS macro IDBD is used to map the DBD module. The 'segment table' is located and looped through, to check each segment to see whether it is compressed. If any compressed segments are found, a return code of 4 is set and the database name is written to the SYSPRINT DD.

DBCMPCHK should be assembled with attributes AMODE24 and RMODE24

```
DBCMPCHK  TITLE 'IDENTIFY DBDS WITH COMPRESSION ON'
*********************************************************************
*   PROGRAM: DBCMPCHK                                               *
*   FUNCTION: IDENTIFY DATABASES WHICH HAVE A COMPRESSION EXIT      *
*   ON AT LEAST ONE SEGMENT.                                        *
*   MACROS USED:                                                    *
*   MVS - OPEN, CLOSE, GET, PUT, USING                              *
*   IMS - IDBD                                                      *
*   REGISTER USAGE:                                                 *
*   R1  - START ADDRESS OF LOADED DBD                               *
*   R2  - PARM HANDLING                                             *
*   R3  - COUNT OF CONTROL CARDS PROCESSED                          *
*   R4  - STORE DBD SEGMENT COUNT                                   *
*   R5  - NOT USED                                                  *
*   R6  - BASE ADDRESSABILITY WITHIN LOADED DBD                     *
*   R7  - ADDRESSABILITY WITHIN LOADED DBD                          *
*   R8  - STORE RETURN CODE                                         *
*   R9  - NOT USED                                                  *
*   R1Ø - NOT USED                                                  *
*   R11 - NOT USED                                                  *
*   R12 - ADDRESSABILITY                                            *
*   R13 - SAVEAREA                                                  *
*   R14 - RETURN ADDRESS                                            *
*   R15 - RETURN CODE                                               *
*   RETURN CODES:                                                   *
*   Ø - NO COMPRESSION ON ANY DATABASES SPECIFIED ON CONTROL CARD   *
*   4 - COMPRESSION EXIT ON ONE OR MORE DATABASES SPECIFIED         *
```

```
*   8 - NO CONTROL DATA SUPPLIED                                    *
*  16 - ERROR LOADING DBD                                           *
*********************************************************************
DBCMPCHK CSECT
         BAKR  R14,RØ
         LR    R12,R15
         USING DBCMPCHK,R12        BASE ADDRESSABILITY
         LA    R13,SAVEAREA        SET UP SAVE AREA
         MVC   SAVEAREA+4(4),=C'F1SA'
         L     R1,Ø(R1)            GET PARM
         SR    R2,R2               ZERO WORK REGISTERS
         SR    R3,R3
         SR    R4,R4
         SR    R6,R6
         SR    R7,R7
         SR    R8,R8
         MVI   PARMFLAG,Ø          INIT FLAG THAT INDICATES PARM DATA
         LH    R2,Ø(R1)            GET PARM LENGTH
         LTR   R2,R2               GOT PARM?
         BZ    FILEOPEN            NO: GOTO OPEN FILES ROUTINE
         MVI   PARMFLAG,1          WORK WITH PARM INPUT -
*                                  NO NEED TO LOOP THRU SYSIN
         MVC   DBDNAME,INITDBD     INITIALIZE DBD VARIABLE
         BCTR  R2,RØ               DECREMENT 1 FROM R2
         EX    R2,GETDBD           GET DBDNAME FROM PARAMETER
FILEOPEN OPEN  (DBDLIB,(INPUT))    OPEN FILES
         OPEN  (SYSPRINT,(OUTPUT))
         CLI   PARMFLAG,1          USING PARM DATA?
         BE    LOADUP              YES, SKIP SYSIN STUFF
         OPEN  (SYSIN,(INPUT))
GETREC   GET   SYSIN,CARD          GET CONTROL CARD
         CLC   CARD,BLANKS         GOT NO DATA ON IT?
         BE    SETCODE             NO! JUMP TO SET BAD CODE
         A     R3,=F'1'            OTHERWISE INCREMENT COUNTER
LOADUP   LOAD  EPLOC=DBDNAME,DCB=DBDLIB,ERRET=LOADERR  LOAD DBD
         LR    R6,RØ               SAVE START ADDRESS POINTER
         USING DBDDSECT,R6         ESTABLISH ADDRESSABILITY
GETSEGNO L     R7,APREFIX          LOAD PREFIX SECTION...
         USING PREFIX,R7           AND USE FOR ADDRESSABILITY
         LH    R4,PRENOSEG         SAVE SEGMENT COUNT IN R4
         LTR   R4,R4               SEG COUNT ZERO?
         BZ    UNLOCK              YES - GO DUMP THIS DBD
         TM    PREACCES,PREGSAM    IS THIS A GSAM DBD?
         BO    UNLOCK              YES - BAIL OUT
         DROP  R7                  ELSE DROP PREFIX ADDRESSING
         L     R7,ASEGTAB          LOAD SEGMENT TABLE
READSEG  EQU *
         USING SEGTAB,R7           USE SEG TABLE ADDRESSING
         TM    SEGPACOP,SEGCPRT    COMPRESSION ON?
         BO    OUTPUT              YES - GO OUTPUT DBD NAME
```

```
         LA    R7,SEGSZE(R7)     ELSE LOAD ADDRESS OF NEXT SEGTAB ENTRY
         BCT   R4,READSEG          AND LOOP
         B     UNLOCK              GO AND RELEASE LOADED DBD
OUTPUT   MVC   FILLER,BLANK72      BLANK OUT FILLER
         PUT   SYSPRINT,CARD       OUTPUT DBD NAME
         LA    R8,4                SET CC4 - COMPRESSION FOUND
UNLOCK   DELETE EPLOC=DBDNAME      RELEASE LOADED DBD
         CLI   PARMFLAG,1          USING PARM DATA?
         BE    XITPOINT            YES, NO NEED TO LOOP BACK
         B     GETREC              GO GET NEXT CONTROL CARD
LOADERR  LA    R8,16               ERROR LOADING DBD - SET CC
         B     CLSEFILE            AND GO LEAVE
SETCODE  LTR   R3,R3               CHECK IF CONTROL CARD CNT IS Ø
         BNZ   CLSEFILE            NO - GO TO EXIT POINT
         LA    R8,8                YES - SET BAD CODE
CLSEFILE CLI   PARMFLAG,1          CLOSE FILES
         BE    XITPOINT
         CLOSE SYSIN
XITPOINT CLOSE DBDLIB              CLOSE FILES
         CLOSE SYSPRINT
         LR    R15,R8              LOAD SAVED RETURN CODE
         PR    ,                   BYE BYE
GETDBD   MVC   DBDNAME(Ø),2(R1)    GET DBDNAME FROM R1 + 2
         LTORG
DBDLIB   DCB   DDNAME=DBDLIB,DSORG=PS,MACRF=GM,EODAD=CLSEFILE
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,MACRF=GM,EODAD=CLSEFILE
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,EODAD=CLSEFILE
SAVEAREA DS    18F
PARMFLAG DS    CL1
BLANKS   DS    ØCL8Ø
INITDBD  DC    CL8' '
BLANK72  DC    CL72' '
CARD     DS    ØCL8Ø
DBDNAME  DS    CL8
FILLER   DS    CL72
         IDBD
         END
```

EXECUTION REXX AND JCL

The following are examples of REXX code and JCL to execute any of the IMS/DB utility modules. Each example has been kept very generic, ie the file allocations allow for invocation of any of the four programs.

Obviously, you can set up REXX or JCL particular to each program. This is advisable, particularly with REXX, as it will allow you to present more user-friendly results. For example, you can trap the return

codes pertinent to each program if they are invoked from a specific EXEC. This allows you to issue user-friendly messages detailing the meaning of the return code. Similarly, output can be formatted more attractively when you know what you are expecting. My generic REXX simply buffers program output and 'SAYs' it to the screen, allowing it to handle the output of different formats.

REXX and JCL should be modified as per the internal comments to specify libraries appropriate to your shop.

IMS/DB utilities execution REXX

```
 /* rexx      : XLOADER
    function  : Execute load module with supplied parm
    author    : Martin Symmers  xx/xx/xx
    ******* change fields indicated by comments *******
*/
 arg pgm parm '/' trace_type
 interpret trace trace_type
 MAIN_PROGRAM:
 call HANDLE_PARM
 call INIT
 call FILES
 call EXECUTE_PROGRAM
 EXIT_POINT:
 "free fi("dbdlib")"
 "free fi("psblib")"
 "free fi("dynalloc")"
 "free fi(sysin) delete"
 "free fi(sysprint) delete"
 x = msg(on)
 EXIT
 HANDLE_PARM:
 do while pgm = ''
   say 'Input program Name:'
   parse upper external pgm .
 end
 do while parm = ''
   say 'Input parameter:'
   parse upper external parm .
 end
 RETURN
 INIT:
 steplib = 'DB.UTIL.LOAD'              /* change to your load library */
 psblib = 'IMSVS.PSBLIB'               /* change to your psb library  */
 dbdlib = 'IMSVS.DBDLIB'               /* change to your dbd library  */
 dynalloc = 'IMSVS.DYNALLOC'        /* change to your dynalloc library */
 RETURN
```

```
FILES:
x = msg(off)
"alloc fi(steplib) da('"steplib"') shr reuse"
"alloc fi(sysin) dummy"
"alloc fi(sysprint) unit(disk) space(10,10) tracks release
 recfm(f b) lrecl(80) blksize(27920) new"
"alloc fi(psblib) da('"psblib"') shr reuse"
"alloc fi(dbdlib) da('"dbdlib"') shr reuse"
"alloc fi(dynalloc) da('"dynalloc"') shr reuse"
RETURN
EXECUTE_PROGRAM:
"call '"steplib"("pgm")' '"parm"'"
ret = rc
say 'RETURN CODE ret' from MODULE 'pgm
say 'RETURN CODE ret' from MODULE 'pgm
say 'RETURN CODE ret' from MODULE 'pgm
say
"execio * diskr sysprint ( stem output. finis"
if output.0 > 0 then do
   do q = 1 to output.0
      say output.q
   end
end
RETURN
```

IMS/DB utilities execution JCL

```
//DBXEPP JOB (DB),MS-DBA,MSGCLASS=X,CLASS=I,NOTIFY=&SYSUID
//*
//* EXECUTE IMS DB UTILITIES
//*
//ENVIRON  SET UTIL=DBPCBLST,             <= MODULE TO BE EXECUTED
//             PARM=                      <= PARAMETER
//*
//STEP010  EXEC  PGM=&UTIL,PARM=&PARM
//STEPLIB  DD  DSN=DB.UTIL.LOAD,          <= CHANGE TO YOUR LOAD LIBRARY
//             DISP=SHR
//DBDLIB   DD  DSN=IMSVS.DBDLIB,          <= CHANGE TO YOUR DBD LIBRARY
//             DISP=SHR
//PSBLIB   DD  DSN=IMSVS.PSBLIB,          <= CHANGE TO YOUR PSB LIBRARY
//             DISP=SHR
//DYNALLOC DD  DSN=IMSVS.DYNALLOC,        <= CHANGE TO YOUR DYNALLOC LIB
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*,
//             DCB=(LRECL=133)
//SYSIN    DD  *
PSB1
PSB2
etc
/*
```

CONCLUSION

I hope you find some of these programs useful and interesting to experiment with – the control blocks processed contain plenty more data which can be accessed with very little modification to my code, if it is of use to you.

Martin Symmers
DBA (UK)

IPL information

Has anyone ever said to you "Quick, I need to know when the last IPL was"? Well I have and that's why I wrote this small but handy REXX program. This program will allow you to quickly access information about when and how the last IPL occurred – information like IPL date and time, IPL load parm, IPL address, device type, IPL address and more, for example:

```
================================================================================
                    I P L   I N F O R M A T I O N              SYSID: SYSB
================================================================================

  Last IPl'ed on : Friday May 23,1997 (97143)           Time : Ø2:56:33

  IPL Address : Ø255  Volume : B52RES  Device type : 339Ø LoadParm: Ø24FSBM1

  CPU Model : 9672    Version : ØC      Serial # : n21344    SMFid : SYSB

  Sysplex name : CGP1PLEX    MVS Release : JBB5522    MVS Level : SP5.2.2

  Nucleus : 1       I/O Config : Ø6   Real Storage: 256    (MB)




 PF3 Exit , Enter Exit
```

The program should be copied to a library concatenated to your SYSPROC and the panel to a library in your ISPPLIB. You can run IPLINFO interactively from ISPF option 6, from native TSO (ISPF panel not required), or even in batch. Batch mode is helpful when you are running in a Sysplex environment. This program was developed under MVS/ESA 5.2.2, ISPF 4.2, TSO/E 2.5, and REXX 1.3.

IPLINFO REXX EXEC

```
/*REXX*/
/* TRACE ?R */
PARSE ARG PNL
call RETRIEVE_MVS_STORAGE
call FIND_IPL_VOLUME_INFO
call JULIAN_TO_REAL_DATE
call DAY_OF_THE_WEEK
call CONVERT_TO_REAL_TIME
if pnl = 'Y' | pnl = 'y' then call DISPLAY_BY_SAY
   else call DISPLAY_BY_PANEL
exit
DISPLAY_BY_SAY:
clear
say '-------------------- IPL INFORMATION --------------------------'
say 'LoadParm = '@mlpr' sysplex name = '@sple '  sysname = '@sname
say 'mvs level = '@prodn'   mvs release = '@prodi'   smf   = '@smf
say 'nucleus = '@nc', i/o config = '@io' ,cpu (model,version) =
('@mdl','@v')'
say 'cpu/lpar serial # '@ser
say 'IPL address : '@iucb' IPL Volume : '@ivol' device type : 'dev
say 'last IPL on : 'wday' 'month' 'day','year' ('@idte') Time
'@hh':'@mm':'@ss
say '-------------------- End of IPL Info --------------------------'
return
DISPLAY_BY_PANEL:
idate = wday' 'month' 'day','year' ('@idte')'
itime = @hh':'@mm':'@ss
address ISPEXEC "DISPLAY PANEL(IPLINFO)"
return
RETRIEVE_MVS_STORAGE:
@cvt = STORAGE(10,4)                      /* get cvt addr from psa    */
@rlstg = STORAGE(D2X(C2D(@cvt)+856),4)    /* get cvtrlstg from cvt    */
@rlstg = (C2D(@rlstg))%1024               /* real storage (in MB)     */
@ecvt  = STORAGE(D2X(C2D(@cvt)+140),4)    /* get cvtecvt from cvt     */
@sple  = STORAGE(D2X(C2D(@ecvt)+8),8)     /* ecvtsple sysplex name    */
@mlpr  = STORAGE(D2X(C2D(@ecvt)+168),8)   /* ecvtmlpr load parm       */
@sname = STORAGE(D2X(C2D(@cvt)+340),8)    /* cvtsname system name     */
@prodn = STORAGE(D2X(C2D(@cvt)-40),8)     /* cvtprodn sp level        */
```

```
@prodi = STORAGE(D2X(C2D(@cvt)-32),8)    /* cvtprodi fmid for mvs    */
@mdl   = STORAGE(D2X(C2D(@cvt)-6),2)     /* cvtmdl cpu model         */
@mdl   = D2X(C2D(@mdl))                  /* cvtmdl cpu model         */
@smca  = STORAGE(D2X(C2D(@cvt)+197),3)   /* smca from cvt            */
@idte  = STORAGE(D2X(C2D(@smca)+340),4)  /* smcaidte from smca       */
@idte  = C2D(@idte)%16                   /* strip F from date(96001F)*/
@idte  = D2X(@idte)                      /* smcaidte ipl date(96001) */
@itme  = STORAGE(D2X(C2D(@smca)+336),4)  /* smcaitme from smca       */
@smf   = STORAGE(D2X(C2D(@smca)+16),4)   /* smcaside from smca       */
@sysad = STORAGE(D2X(C2D(@cvt)+48),4)    /* cvtsysad ipl volume      */
@ext2  = STORAGE(D2X(C2D(@cvt)+328),4)   /* cvtext2 from cvt         */
@nc    = STORAGE(D2X(C2D(@ext2)+4),1)    /* cvtnucls      nucleus    */
@io    = STORAGE(D2X(C2D(@ext2)+6),2)    /* cvtiocid      i/o config */
@pcca  = STORAGE(208,4)                  /* pcca from psa            */
@cpid  = STORAGE(D2X(C2D(@pcca)+4),12)   /* pccacpid from pcca       */
@v     = substr(@cpid,1,2)               /* cpu version from pccacpid*/
@ser   = 'n'substr(@cpid,4,5)           /* cpu/lpar serial number   */
return
FIND_IPL_VOLUME_INFO:                    /* volume name/ucb          */
@iucb  = STORAGE(D2X(C2D(@sysad)+12),4)  /* ipl volume ucb addr      */
@ivol  = STORAGE(D2X(C2D(@sysad)+28),6)  /* ipl volume name          */
@ityp  = STORAGE(D2X(C2D(@sysad)+19),2)  /* ipl volume devtype       */
@ityp  = D2X(C2D(@ityp))
if @ityp = 'F00' then dev = 3390
   else if @ityp = 'E00' then dev = 3380
        else dev = '????'
i1st = substr(@iucb,1,1);irest = substr(@iucb,2,3)
if i1st > 9 | i1st < 0 then @iucb = '0'irest
return
JULIAN_TO_REAL_DATE:
m.1=31;m.2=28;m.3=31;m.4=30;m.5=31;m.6=30
m.7=31;m.8=31;m.9=30;m.10=31;m.11=30;m.12=31
n.1='January';n.2='February';n.3='March';n.4='April';n.5='May'
n.6='June';n.7='July';n.8='August';n.9='September';n.10='October'
n.11='November';n.12='December'
@i=1
do i = 96400 to 99400 by 1000
   if i > @idte
      then do                        /* lets find the year  */
           @yrsuf = substr(@idte,1,2)   /* '96 or '97 ect.     */
           day = @idte - (i - 400)      /* 96nnn ...get nnn    */
           day2 = day                   /* used for day of week*/
           year = '19'@yrsuf            /* append 96 to '19    */
           if (@yrsuf/4) = (@yrsuf%4) then m.2 = 29 /*leap yr  */
           j = 0; found = n
           do until(found = y)          /* find month          */
              j = j + 1
              if day > m.j then do
                                day = day - m.j
                                end
```

```
                              else do
                                   month = n.j
                                   found = y
                                   end
           end
           leave
      end
end
return
DAY_OF_THE_WEEK:
d.1='Monday';d.2='Tuesday';d.3='Wednesday';d.4='Thursday';d.5='Friday'
d.6='Saturday';d.7='Sunday'
if day2 > 7 then d1 = (day2 - (7 * (day2 % 7)))
            else d1 = day2
if d1 = 0 then d1 = 7
if year = 1997
          then do
               d1 = d1 + 2
               if d1 > 7 then d1 = d1 - 7
               end
          else if year = 1998
               then do
                    d1 = d1 + 3
                    if d1 > 7 then d1 = d1 - 7
                    end
          else if year = 1999
               then do
                    d1 = d1 + 4
                    if d1 > 7 then d1 = d1 - 7
                    end
wday = d.d1
return
CONVERT_TO_REAL_TIME:                    /* this format ==> HH:MM:SS */
@itme  = C2D(@itme)
@hh    = @itme % 360000                  /* ipltime - hour........HH */
if @hh < 10 then @hh = '0'@hh
@mm    = (@itme - @hh * 360000) % 6000   /* ipltime - minute .....MM */
if @mm < 10 then @mm = '0'@mm
@ss    = (@itme - ((@hh * 360000)+(@mm * 6000))) % 100 /*second ..SS */
if @ss < 10 then @ss = '0'@ss
return
```

IPLINFO PANEL

```
)ATTR
  | TYPE(INPUT) CAPS(ON) INTENS(HIGH) JUST(ASIS) PAD(' ')
  ~ TYPE(INPUT) CAPS(ON) INTENS(LOW) JUST(ASIS) PAD(' ')
  _ TYPE(INPUT) CAPS(ON) INTENS(LOW) JUST(ASIS) PAD(' ')
  ! TYPE(OUTPUT) INTENS(LOW) CAPS(OFF)
```

```
  " TYPE(OUTPUT) INTENS(HIGH) JUST(LEFT) PAD(NULLS)
)BODY EXPAND(¢¢)
+==========================¢=¢========================================
%                   I P L   I N F O R M A T I O N          SYSID:!@sname
+==========================¢=¢========================================
%
%  Last IPl'ed on :!idate                               %Time :!itime
%
%  IPL Address :!@iucb%Volume :!@ivol  %Device type :!dev %LoadParm:!@mlpr
%
%  CPU Model :!@mdl  % Version :!@v    % Serial # :!@ser    % SMFid :!@smf
%
%  Sysplex name :!@sple      %MVS Release :!@prodi    %MVS Level :!@prodn
%
%  Nucleus :!@nc    %I/O Config :!@io %Real Storage:!@rlstg%(MB)
%
%
%
%
%
%
%
%
%
% PF3+Exit+,%Enter+Exit
)INIT
)PROC
)END
```

IPLINFO JCL

```
//IPLINFO   JOB (9999,xxx),*** your-job-card ***
//*
//TSO     EXEC  PGM=IKJEFT01,DYNAMNBR=20
//SYSPROC  DD  DSN=** rexx-library-where-iplinfo-lives **,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//ABNLIGNR DD  DUMMY
//SYSTSIN DD   *
 IPLINFO Y
/*
//
```

Garry Milroy
MVS Technical Analyst
Consumers Gas (Canada)

Extracting log information about jobs and tapes

INTRODUCTION

SCANLOG is a utility designed to search the system log for important information. It retrieves meaningful messages from the log and processes them to present a concise report about jobs and tape usage.

If you need to know at what time a job started or ended, how many tape drives it used, how many mounts it requested, whether it abended or was cancelled, how much your tape drives are being used, or what the peak hours of tape utilization are, SCANLOG can help. When you run it, you are presented with the following ISPF screen:

```
*------------------------ Scan Log ------------------------*
|                                                          |
|   File(empty for LOG):                                   |
|                                                          |
|          Start date ( yyyy/mm/dd )..: 1997 / Ø6 / 26     |
|          Start time ( ØØ to 23 )....: 2Ø                 |
|          Period ( 8 12 24 hours )...: 12                 |
|                                                          |
|  Job types to process (mark with any non blank char):    |
|                                                          |
|         Only jobs with tapes........: *                  |
|         All jobs....................:                    |
|         Tso users...................:                    |
|         Started tasks...............:                    |
|                                                          |
*----------------------------------------------------------*
```

On the 'File' field, you can either leave it blank, in which case the active log will be searched, or you can specify a sequential file that holds the log records you want to search. This is useful if you have back-up copies of old logs. The information in the file should be in a format similar to that obtained by the SDSF PRINT ODSN command.

'Start date', by default, is yesterday's date, because the idea behind SCANLOG is to find out what happened last night. You can specify any date you want as long as your input file or the current log contains it.

By default, 'Start time' is 20.00 (8 pm), a typical hour for a batch run, but you can specify any hour from 00 to 23.

'Period' is the number of hours to analyse. Because of built-in scales for the output, the period must be 8, 12, or 24 hours.

You can specify what types of job you wish to include in your report: ie only jobs with tapes, all jobs, TSO users, or started tasks. Specify at least one category, otherwise the report will be blank.

Once you specify your input and hit ENTER, SCANLOG builds a job and submits it. The job has your name as prefix. First it prints the active log to a temporary file (if necessary), or uses the file you supplied as input for the sort steps. These steps filter the relevant messages and the time span to consider.

Finally, the last step processes the output of the sort and presents the results in the form of three sequential files. Their names are *userid*.SCANLOG1, *userid*.SCANLOG2, and *userid*.SCANLOG3, where *userid* is your TSO user-id.

Here is some sample output. It assumes you requested all job categories.

```
TSU  FRT644Ø      97/Ø6/26    2Ø:ØØ:Ø9    2Ø:29:37    Ø
JOB  PXSSFD1  Ab  97/Ø6/26    2Ø:ØØ:15    2Ø:Ø5:15    Ø
JOB  CFRTSAVO     97/Ø6/26    21:ØØ:15    21:1Ø:Ø2    1    Ø775
JOB  ATR665IS Ca  97/Ø6/26    21:ØØ:27    21:Ø5:27    Ø
JOB  TRDFGOLI     97/Ø6/26    21:ØØ:34    21:Ø9:52    Ø
STC  AGTY78       97/Ø6/26    22:Ø2:Ø2    22:54:Ø2    4    Ø77Ø Ø771
JOB  ATR4ØA       97/Ø6/26    22:Ø2:Ø3    22:11:17    Ø
JOB  PUYOPR6      97/Ø6/26    23:21:52    23:49:22    3    Ø772
JOB  SCEWØ2ØP CC  97/Ø6/27    ØØ:22:46    ØØ:26:27    Ø
JOB  SRT4Ø3ØP     97/Ø6/27    Ø1:25:29    Ø2:38:15    Ø
J5B  EUTAXS2      97/Ø6/27    Ø1:35:57    Ø1:56:18    Ø
```

The above is simply a listing of all the jobs. In the first column you see their category, then comes the name, the start date, the start time, the end time (might be the next day), the number of tape mounts the job requested, and the tape units it used.

In the example, started task AGTY78 requested four mounts and used two drives (0770 and 0771). Also, you can see a job that abended

(marked Ab), another that was cancelled (marked Ca), and yet another that ended because of condition codes (marked CC).

```
0775    97/06/26    20:00:15    20:10:02    UYPPSAVO     TRFS.CITRR.SAVE.SYST5
0776    97/06/26    20:02:04    20:03:54    AUDS5520     TRFS.FRE33.DS20.PCOP
0777    97/06/26    21:02:04    21:03:55    AYT66C20     DUI.TRFS.FRE33.DS20
0778    97/06/26    21:02:17    21:19:48    BSEROIE      TRF5.DS23.SAVE.SYST5
0779    97/06/26    21:02:50    21:04:20    REWE4480     TRFS.FRE34.DVC4.PCOP
0773    97/06/26    21:02:50    21:04:21    AFD4PE80     DUP.TRFS.FRE33.DB80
0774    97/06/26    22:02:59    22:35:43    BIIUA33      TRF4.DS90.D98.SAVEF
0772    97/06/26    22:03:00    22:26:11    GTREEVE5     TRF0.DS80.D99.SABGG
0773    97/06/26    22:03:09    22:20:01    PO99FR4      TRF3.DS70.D65.SAVU7
```

The above example is similar to the first, but is taken from the point of view of the mount requests. This file will be empty if no job has tapes associated. Column one shows the unit, then comes the date, the hour the mount was requested, the hour of dismount, the job name that made the request, and the file name associated with it. Note that a mount/dismount has nothing to do with a job start or end. A job can request several mounts/dismounts during its lifetime. Also note that the file name associated with the mount request may not appear.

Figure 1 shows a graphical view of our second example. It helps you to see how much your drives are loaded, what the peak hours are, and so on. It may be useful in helping you to redistribute the batch load should you be getting tight in drive units. It shows along the requested period what the occupation of each unit is, regardless of the jobs involved.

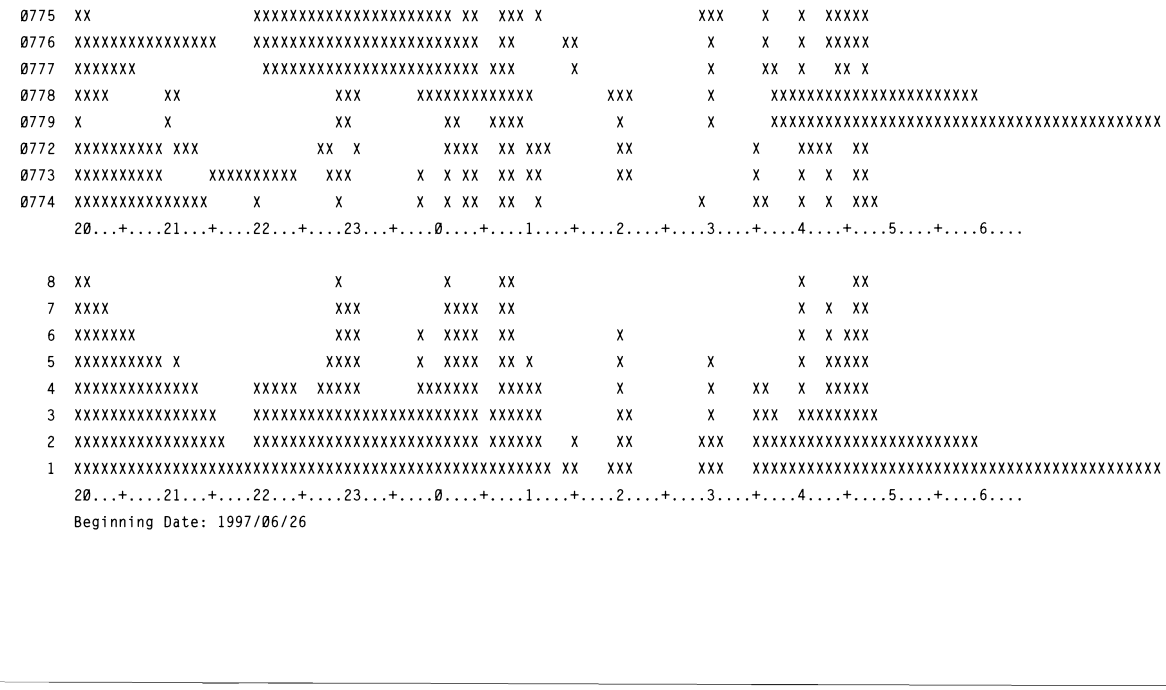
For example, you can see that unit 0776 worked continuously from 20.00 to 21.30 and from 22.00 to 00.30. On the lower part of the graphic, you have the accumulated use of all units: for example, at 20.00, eight drives were in use, while around 21.45 there was only one.

Note that you may have more drives than those that actually appear in the report: if they are not used, and therefore not referred to in any log message, SCANLOG will be unaware of them.

HOW TO INSTALL AND RUN SCANLOG

SCANLOG consists of two files: an ISPF panel, SCANLOG1, that you should place in your ISPPLIB concatenation, and the REXX EXEC, SCANLOG, that should be in your SYSEXEC/SYSPROC

Figure 1: Graphical view of the output



concatenation. You need to set a few variable names at the beginning of the EXEC: pds_exec should hold the name of the PDS where you put SCANLOG: spool, spoolvol, chk1, and chk1vol should hold the names of your spool and checkpoint files and volumes. You can also change the names of the output files if you wish. Note that each run of SCANLOG destroys the previous output files.

When you specify the active log as the input file (by leaving the file field blank, in the ISPF panel), the first step of SCANLOG runs SDSF in batch to print the log. Alternatively, you can go to the log, print it to a file, and declare that file as input. If you do so, use the PRINT ODSN form of the SDSF PRINT command, not the PRINT FILE form, since they produce different results.

Finally, a consideration about jobs whose start or end falls outside the considered time frame. If a job begins before the time but ends inside it, it is ignored; if it starts inside but ends later, it is considered to terminate at the end of the period.

SCANLOG was developed for MVS 5.1, but should run in other versions with little or no modification.

SCANLOG REXX EXEC

```
/*****  REXX MVS  *  Tso > Batch  ********************************/
/*  SCANLOG: Retrieve information from system log about jobs,     */
/*           tapes, started tasks, or tso users                   */
/*********************************************************************/
arg hora scale datgra alljobs .            /* arguments for       */
                                           /* batch cycle only    */
parse source . . progname . . . env .      /* find environment    */
if env = "TSO" then call executa_tso       /* select execution    */
               else call executa_batch     /* cycle: tso or batch */
saida:
exit
/*********************************************************************/
/*                      TSO execution                                */
/*********************************************************************/
executa_tso:
pds_exec= "SIS.EXEC.REXX"                /* this exec's PDS       */
spool   = "SYS1.HASPA002"                /* spool filename        */
spoolvol= "VOLA02"                       /* and volume            */
chk1    = "SYS1.HASPCK03"                /* checkpoint filename   */
chk1vol = "VOLA03"                       /* and volume            */
out1    = userid()".SCANLOG1"            /* output filenames      */
```

```
out2    = userid()".SCANLOG2"
out3    = userid()".SCANLOG3"
jobfile = userid()".LIXO"                /* SDSF print temp file  */
dummyret = display_panel()               /* call display ISPF pan */
parse var dummyret logfile yyyy mm dd hora scale ty1 ty2 ty3 ty4
ADDRESS ISPEXEC 'VERASE (LF YYYY MM DD HO SC A B C D)'
horini = hora":00:00"                    /* format time and date  */
horfim = hora + scale
if horfim > 23 then,
   horfim = horfim - 24
horfim = horfim":00:00"
horfim = right(horfim,8,"0")
horini = right(horini,8,"0")
yy = right(yyyy,2)
datgra = yyyy"/"mm"/"dd
datini  = mm"/"dd"/"yy
datinij = datajul(yy||mm||dd)
if horini < horfim then do
   datfim  = datini
   datfimj = datinij
end
else do
   ddd = right(datinij,3)
   yyy = left(datinij,2)
   ddd = ddd+1
   if (yyy//4 =0 & ddd>366)|,
      (yyy//4¬=0 & ddd>365) then do
      yyy=yyy+1
      ddd=1
   end
   datfimj = yyy||right(ddd,3,"0")
   datfim  = conv_date(datfimj "US")
end
scope = horini datini horfim datfim
dat_hor_ini = datinij" "horini
dat_hor_fim = datfimj" "horfim
incl = " INCLUDE COND=("                 /*format second SORT line*/
if ty2¬="." then alljobs = 1
if ty1¬="."|ty2¬="." then incl = incl"39,3,CH,EQ,C'JOB'"
if ty3¬="." then do
if length(incl) < 16 then incl = incl"39,3,CH,EQ,C'TSU'"
                     else incl = incl",OR,39,3,CH,EQ,C'TSU'"
end
if ty4¬="." then do
if length(incl) < 16 then incl = incl"39,3,CH,EQ,C'STC'"
                     else incl = incl",OR,39,3,CH,EQ,C'STC'"
end
incl = incl")"
if length(incl) > 70 then do
   pos1 = pos(",OR,",incl)+3
```

```
   incl1 = left(incl,pos1)
   incl2 = "               "substr(incl,pos1+1)
end
else do
   incl1 = incl
   incl2 = ""
end
/*********************************************************************/
/*                   Create and submit job                           */
/*********************************************************************/
call alocar
dropbuf
queue "//SDSFLOG1 JOB CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),REGION=2M,"
queue "//   NOTIFY="userid()
queue "//*"
queue "//STEPØ EXEC PGM=IDCAMS"
if logfile = "." then do
 queue "//FICTEMP  DD DISP=(NEW,CATLG),UNIT=3390,DSN="jobfile","
 queue "//         DCB=(RECFM=FB,LRECL=132,BLKSIZE=13200),"
 queue "//         DSORG=PS,SPACE=(TRK,(60,60))"
end
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSIN    DD *"
queue "  DELETE "out1
queue "  DELETE "out2
queue "  DELETE "out3
queue "/*"
queue "//*"
if logfile = "." then do
 queue "//SDSFØ EXEC PGM=SDSF,REGION=2M,PARM='++60,228'"
 queue "//HASPACØØ DD DISP=SHR,DSN="spool","
 queue "//            UNIT=3390,VOL=SER="spoolvol
 queue "//HASPCKPT DD DISP=SHR,DSN="chk1","
 queue "//            UNIT=3390,VOL=SER="chk1vol
 queue "//ISFOUT   DD DUMMY"
 queue "//ISFIN    DD *"
 queue "LOG"
 queue "PRINT ODSN '"jobfile"' * SHR"
 queue "PRINT "scope
 queue "PRINT CLOSE"
 queue "/*"
 queue "//*"
end
queue "//STEP1 EXEC PGM=SORT"
if logfile = "." then,
 queue "//SORTIN   DD DISP=(OLD,DELETE),DSN="jobfile
else,
 queue "//SORTIN   DD DISP=SHR,DSN="logfile
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSOUT   DD SYSOUT=*"
```

```
queue "//SORTOUT  DD DISP=(NEW,PASS),DSN=&&TEMP1,UNIT=3390,"
queue "//         DCB=(RECFM=FB,LRECL=132,BLKSIZE=13200),"
queue "//         SPACE=(TRK,(15,15))"
queue "//SYSIN    DD *"
queue "  SORT FIELDS=COPY"
queue "  INCLUDE COND=((58,6,CH,EQ,C'IEC501',OR,"
queue "                58,6,CH,EQ,C'IEC502',OR,"
queue "                58,6,CH,EQ,C'IEF125',OR,"
queue "                58,6,CH,EQ,C'IEF126',OR,"
queue "                58,6,CH,EQ,C'IEF233',OR,"
queue "                58,6,CH,EQ,C'IEF234',OR,"
queue "                58,6,CH,EQ,C'IEF403',OR,"
queue "                58,6,CH,EQ,C'IEF404',OR,"
queue "                58,6,CH,EQ,C'IEF450',OR,"
queue "                58,6,CH,EQ,C'IEF451',OR,"
queue "                58,6,CH,EQ,C'IEF453'),AND,"
queue "                21,14,CH,GE,C'"dat_hor_ini"',AND,"
queue "                21,14,CH,LE,C'"dat_hor_fim"')"
queue "/*"
queue "//*"
queue "//STEP2 EXEC PGM=SORT"
queue "//SORTIN   DD DISP=(OLD,DELETE),DSN=&&TEMP1"
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSOUT   DD SYSOUT=*"
queue "//SORTOUT  DD DISP=(NEW,PASS),DSN=&&TEMP2,UNIT=3390,"
queue "//         DCB=(RECFM=FB,LRECL=132,BLKSIZE=13200),"
queue "//         SPACE=(TRK,(15,15))"
queue "//SYSIN    DD *"
queue " SORT FIELDS=COPY"
queue incl1
if incl2 ¬="" then queue incl2
queue "/*"
queue "//*"
queue "//STEP3 EXEC PGM=IRXJCL,"
queue "//  PARM='"progname hora scale datgra alljobs"'"
queue "//SYSEXEC  DD DISP=SHR,DSN="pds_exec
queue "//SYSPRINT DD SYSOUT=*"
queue "//SYSTSPRT DD SYSOUT=*"
queue "//FIC      DD DISP=(OLD,DELETE),DSN=&&TEMP2"
queue "//SAIDA1   DD DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(1,1)),"
queue "//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),"
queue "//         DSN="out1",UNIT=3390"
queue "//SAIDA2   DD DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(1,1)),"
queue "//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000),"
queue "//         DSN="out2",UNIT=3390"
queue "//SAIDA3   DD DISP=(NEW,CATLG,DELETE),SPACE=(TRK,(1,1)),"
queue "//         DCB=(RECFM=FB,LRECL=132,BLKSIZE=13200),"
queue "//         DSN="out3",UNIT=3390"
queue ""
"execio * diskw jobe (finis"
```

```
"submit '"jobnome"'"
"free  dd (jobe)"
return
/*********************************************************************/
/*                     Batch execution                               */
/*********************************************************************/
executa_batch:
sym = "X"                                   /* symbol for file 3*/
j0 = 0                                      /* j0 total jobs    */
u0 = 0                                      /* u0 total mounts  */
msg1 = "IEF403"                             /* start job or stc */
msg2 = "IEF404"                             /* end   job or stc */
msg3 = "IEF233"                             /* mount            */
msg4 = "IEC501"                             /* mount comp       */
msg5 = "IEF234"                             /* dismount         */
msg6 = "IEC502"                             /* dismount         */
msg7 = "IEF453"                             /* job abend        */
msg8 = "IEF450"                             /* job cancelled    */
msg9 = "IEF451"                             /* job ended by cc  */
msg10= "IEF125"                             /* tsu logon        */
msg11= "IEF126"                             /* tsu logoff       */
do alfa = 1 to 99999                        /* read loop        */
   execio 1 diskr fic
   if rc¬=0 then leave alfa
   pull linha
   msg = substr(linha,58,6)                 /* message          */
   hor = substr(linha,27,8)                 /* message hour     */
   dat = substr(linha,21,5)                 /* message date     */
   job = substr(linha,39,8)                 /* job number       */
   if job = "        " then,                /* if spaces, create*/
      job = right(alfa,8,"X")               /* a XXX job number */
   call select_message                      /* process message  */
end
execio 0 diskr fic "(finis"
/*********************************************************************/
/*                        Write file 1                               */
/*********************************************************************/
j1 = 0
do p = 1 to j0;
   if carunit0.p¬="" | alljobs = 1 | typejob0.p¬="JOB" then do
      if ¬datatype(left(horafim0.p,2),"W"),
       then do
         hhh = hora+scale-1
         if hhh > 24 then hhh = hhh-24
         hhh=right(hhh,2,"0")
         horafim0.p=hhh||":59:00"
      end
      j1 = j1 + 1
      dataini1.j1 = dataini0.p
      horaini1.j1 = horaini0.p
```

```
      horafim1.j1 = horafim0.p
      nomejob1.j1 = nomejob0.p
      jobabor1.j1 = jobabor0.p
      carunit1.j1 = carunit0.p
      carmoun1.j1 = carmoun0.p
      jobnume1.j1 = jobnume0.p
      typejob1.j1 = typejob0.p
      call trata_horas horaini1.j1 horafim1.j1
      hinidec1.j1 = hini_dec
      hfimdec1.j1 = hfim_dec
   end
end
do k = 1 to j1
   linout = left(typejob1.k,4),
            left(nomejob1.k,8),
            jobabor1.k,
            left(conv_date(datain1.k "EU"),11),
            left(horaini1.k,11),
            left(horafim1.k,11),
            left(carmoun1.k,3),
            carunit1.k
   queue linout
end
queue ""
execio "*" diskw saida1 "(finis"
/*********************************************************************/
/*                          Write file 2                             */
/*********************************************************************/
u2 = 0
do q = 1 to u0                                  /* if no dismount  */
   if ¬datatype(left(hfim0.q,2),"W") then ,     /* found, seek eoj */
    do k = 1 to j1                              /* by jobnumber    */
       if jobnume1.k = jobn0.q then do          /* and supply same */
          hfim0.q = horafim1.k                  /* end hour        */
          leave k
       end
    end
   if ¬datatype(left(hfim0.q,2),"W") then do    /* if still not    */
         hhh = hora+scale-1                     /* found, supply   */
         if hhh > 24 then hhh = hhh-24          /* end of window   */
         hhh = right(hhh,2,"0")
         hfim0.q = hhh||":59:00"
   end
   u2 = u2 + 1
   unit2.u2 = unit0.q
   data2.u2 = data0.q
   hini2.u2 = hini0.q
   hfim2.u2 = hfim0.q
   jobe2.u2 = jobe0.q
   file2.u2 = file0.q
```

```
   call trata_horas hini2.u2 hfim2.u2
   dini2.u2 = hini_dec
   dfim2.u2 = hfim_dec
end
do k = 1 to u2
   linout = left(unit2.k,8) ||,
            left(conv_date(data2.k "EU"),12) ||,
            left(hini2.k,12) ||,
            left(hfim2.k,12) ||,
            left(jobe2.k,12) file2.k
   queue linout
end
queue ""
execio "*" diskw saida2 "(finis"
/*********************************************************************/
/*                         Write file 3                              */
/*********************************************************************/
call graphic_scale
total = 0
x3 = 0
do k = 1 to u2
   ok=0
   do j = 1 to x3
      if unit2.k = cart3.j then do
         ok=1
         leave j
      end
   end
   if ok=0 then x3 = x3+1
   cart3.j = unit2.k
   ini = (dini2.k * divisor)%1 + 1
   fim = (dfim2.k * divisor)%1 + 1
   do a = ini to fim
      tab3.j.a = sym
   end
end
do z = 1 to x3
   tab33 = " "
   do a = 1 to 120
      if tab3.z.a ¬=sym then tab3.z.a = " "
      tab33 = tab33 || tab3.z.a
   end
   linout3.z = cart3.z tab33
end
do a = 1 to 120
   y = 0
   do z = 1 to x3
      if tab3.z.a =sym then do
         total = total + 1
         y = y + 1
```

```
         tab4.y.a = sym
      end
   end
end
do z = 1 to x3
   tab44 = " "
   do a = 1 to 120
      if tab4.z.a ¬=sym then tab4.z.a = " "
      tab44 = tab44 || tab4.z.a
   end
   linout4.z = right(z,length(cart3.1)) tab44
end
if total > 0 then total = total*100%(120*x3)
cab3 = "Beginning date:" datgra copies(" ",36)
cab3 = cab3  "Utilization: "total"%"
do k = 1 to x3
   queue linout3.k
end
queue "      "||cab2
queue " "
queue " "
queue " "
do k = x3 to 1 by -1
   queue linout4.k
end
queue "      "||cab2
queue " "
queue "      "||cab3
queue ""
execio "*" diskw saida3 "(finis"
return
/*********************************************************************/
/*              Subroutines:   Select message                        */
/*********************************************************************/
select_message:
  select
/***************************************************** job start */
   when msg = msg1 |,
        msg = msg10 then do
      j0 = j0 + 1
      carunit0.j0 = ""
      carmoun0.j0 = 0
      datain0.j0 = dat
      horain0.j0 = hor
      jobnume0.j0 = job
      jobabor0.j0 = "  "
      nomejob0.j0 = word(substr(linha,66,8),1)
      nomejob0.j0 = left(nomejob0.j0,8)
      typejob0.j0 = substr(linha,39,3)
   end
```

```
/*********************************************************** job end */
   when msg = msg2 |,
        msg = msg7 |,
        msg = msg9 |,
        msg = msg11|,
        msg = msg8 then do
      do p = j0 to 1 by -1
        if jobnume0.p = job then do
           horafim0.p = substr(linha,27,8)
           if msg = msg7 then jobabor0.p = "Ab"  /* job aborted   */
           if msg = msg8 then jobabor0.p = "Ca"  /* job cancelled */
           if msg = msg9 then jobabor0.p = "CC"  /* ended by CC   */
           leave p
        end
      end
   end
/*************************************************** mount tape */
   when msg = msg3 |,
        msg = msg4 then do
      zzz = substr(linha,68)
      parse var zzz unit "," . "," . "," pa1 "," pa2 "," pa3 "," pa4
      if msg = msg3 then do
         job1 = pa1
         fic1 = left(pa3,20)
      end; else do
         job1 = pa2
         fic1 = left(pa4,20)
      end
      do p = j0 to 1 by -1 ;
        if jobnume0.p = job then do
           carmoun0.p = carmoun0.p + 1
           do x = 1 to words(carunit0.p)
              if word(carunit0.p,x) = unit then do
                 leave p
              end
           end
           carunit0.p = carunit0.p unit
           leave p
        end
      end
      u0 = u0 + 1
      unit0.u0 = unit
      data0.u0 = dat
      hini0.u0 = hor
      jobe0.u0 = job1
      jobn0.u0 = job
      file0.u0 = fic1
   end
/************************************************* dismount tape */
   when msg = msg5 |,
```

```
        msg = msg6 then do
      zzz = translate(word(substr(linha,66),2)," ",",")
      unit= word(zzz,1)
      do q = u0 to 1 by -1 ;
        if unit0.q = unit & job = jobn0.q &,      /* job ok &     */
         ¬(datatype(left(hfim0.q,2),"W")) then do /* no end hour  */
           hfim0.q = hor
           leave q
        end
      end
   end
   otherwise nop
  end                                          /* end select message */
 return
/*********************************************************************/
/*           Convert time(hh:mm:ss)  to decimal format               */
/*********************************************************************/
trata_horas:
 parse arg hh1 ":" mm1 ":" ss1 " " hh2 ":" mm2 ":" ss2
 hini_dec = hh1 + mm1/60 + ss1/3600
 hfim_dec = hh2 + mm2/60 + ss2/3600
 select
   when hini_dec >= hora then hini_dec=hini_dec - hora
   when hini_dec <  hora then hini_dec=hini_dec - hora + 24
   otherwise nop
 end
 select
   when hfim_dec >= hora then hfim_dec=hfim_dec - hora
   when hfim_dec <  hora then hfim_dec=hfim_dec - hora + 24
   otherwise nop
 end
 duracao  = hfim_dec - hini_dec
 if duracao < 0 then duracao = duracao + 24
 duracao  = format(duracao,2,4)
 hini_dec = format(hini_dec,2,4)
 hfim_dec = format(hfim_dec,2,4)
return
/*********************************************************************/
/*                     Select graphic scale                          */
/*********************************************************************/
graphic_scale:
 do a = 0 to 23
    h.a = hora + a
    if h.a > 23 then h.a = h.a - 24
    h.a = left(h.a,2,".")
 end
 select
  when scale=8 | scale = "" then do
   divisor = 15
   cab2=      h.0"...+....+...."h.1"...+....+...."h.2"...+....+...."
```

```
   cab2=cab2||h.3"...+....+...."h.4"...+....+...."h.5"...+....+...."
   cab2=cab2||h.6"...+....+...."h.7"...+....+....hours"
  end
  when scale=12 then do
   divisor = 10
   cab2=      h.0"...+...."h.1"...+...."h.2"...+...."h.3"...+...."
   cab2=cab2||h.4"...+...."h.5"...+...."h.6"...+...."h.7"...+...."
   cab2=cab2||h.8"...+...."h.9"...+...."h.10"...+...."h.11"...+...."
   cab2=cab2||"hours"
   end
  when scale=24 then do
   divisor = 5
   cab2=      h.0"..."h.1"..."h.2"..."h.3"..."h.4"..."h.5"..."
   cab2=cab2||h.6"..."h.7"..."h.8"..."h.9"..."h.10"..."h.11"..."
   cab2=cab2||h.12"..."h.13"..."h.14"..."h.15"..."h.16"..."h.17"..."
   cab2=cab2||h.18"..."h.19"..."h.20"..."h.21"..."h.22"..."h.23"..."
   cab2=cab2||"hours"
  end
  otherwise nop
 end
return
/*********************************************************************/
/*   Julian date conversion to yy/mm/dd (EU)  or  mm/dd/yy (US)   */
/*********************************************************************/
conv_date: procedure
 arg date_in  format
 yy = left(date_in,2) ; ddd = right(date_in,3)
 if yy//4 = 0 then ac = 1 ; else ac = 0
 do mes = 1 to 12
    dddant = ddd
    mesant = mes
    select
       when mes=4|mes=6|mes=9|mes=11 then ndias = 30
       when mes=2 & ac=1             then ndias = 29
       when mes=2 & ac=0             then ndias = 28
       otherwise                          ndias = 31
    end
    ddd = ddd - ndias
    if ddd ¬>0 then leave
 end
 if format = "US" then,
    return right(mesant,2,"0")"/"right(dddant,2,"0")"/"yy
 if format = "EU" then,
    return yy||"/"||right(mesant,2,"0")||"/"||right(dddant,2,"0")
/*********************************************************************/
/*        Convert yymmdd date to julian format(yyddd)                */
/*********************************************************************/
juliana: procedure
arg date_in
aa = left(date_in,2) ; mm = substr(date_in,3,2) ; dd = right(date_in,2)
```

```
if aa//4 = 0 then ac = 1 ; else ac = 0
select
   when  mm = 1   then x = 0
   when  mm = 2   then x = 31
   when  mm = 3   then x = 59 + ac
   when  mm = 4   then x = 90 + ac
   when  mm = 5   then x = 120 + ac
   when  mm = 6   then x = 151 + ac
   when  mm = 7   then x = 181 + ac
   when  mm = 8   then x = 212 + ac
   when  mm = 9   then x = 243 + ac
   when  mm = 10  then x = 273 + ac
   when  mm = 11  then x = 304 + ac
   when  mm = 12  then x = 334 + ac
   otherwise nop
end
j = x + dd
return aa||right(j,3,'0')
/*********************************************************************/
/*                      Alloc job control file                       */
/*********************************************************************/
alocar:
 xx = msg(off)
 jobnome = userid()".JOBTEMP"
 "free  dd (jobe)"
 "alloc da('"jobnome"') dd(jobe) new reuse blksize(8000),
     lrecl(80) recfm(f,b) dsorg(ps) space(1 1) tracks delete "
 if rc¬=0 then do
    say "Error "rc" allocating "jobnome
    signal saida
 end
return
/*********************************************************************/
/*                       Display ISPF panel                          */
/*********************************************************************/
display_panel: procedure
datas = date(s)                        /* default date is yesterday */
yyyy = left(datas,4)                   /* subtract one from today's */
mm   = substr(datas,5,2)
dd   = right(datas,2)-1
if dd = 0 then do
   mm = mm-1
   if mm = 0 then do
      yyyy = yyyy-1
      mm = 12
   end
   if mm=4|mm=6|mm=9|mm=11 then dd=30
   else if mm=2 then do
           if yyyy//4=0 then dd=29
           else dd=28
```

```
         end
         else dd=31
end
sc = 12                                /* default scale is 12 hours */
ho = 2Ø                                /* default start hour is 2Ø  */
a  = "*"
ADDRESS ISPEXEC "ADDPOP ROW(5) COLUMN(4)"
ADDRESS ISPEXEC "DISPLAY PANEL(SCANLOG1)"
if rc¬=Ø then exit
ADDRESS ISPEXEC "REMPOP"
if lf = "" then lf = "."
if a  = "" then a  = "."
if b  = "" then b  = "."
if c  = "" then c  = "."
if d  = "" then d  = "."
ho = right(ho,2,"Ø")
mm = right(mm,2,"Ø")
dd = right(dd,2,"Ø")
return lf yyyy mm dd ho sc a b c d
```

SCANLOG1 ISPF PANEL

```
)ATTR
  _ TYPE(INPUT) CAPS(ON) JUST(LEFT) COLOR(RED)
  # TYPE(INPUT) CAPS(ON) JUST(RIGHT) PAD(Ø) COLOR(RED)
  ! TYPE(OUTPUT) SKIP(ON) COLOR(WHITE)
  ? TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(PINK)
  % TYPE(TEXT) INTENS(HIGH) SKIP(ON) COLOR(YELLOW)
  + TYPE(TEXT) INTENS(LOW) SKIP(ON) COLOR(GREEN)
)BODY WINDOW(6Ø,15)
+
+
?  File(empty for LOG):_LF
+
%        Start date ( yyyy/mm/dd )...:#YYYY+/#MM+/#DD+
%        Start hour ( ØØ to 23 )......:#HO+
%        Period ( 8 12 24 hours )....:#SC+
+
?  Job types to process (mark with any non blank char):
+
%        Only jobs with tapes........:_A+
%        All jobs....................:_B+
%        Tso users...................:_C+
%        Started tasks...............:_D+
+
)INIT
&END = PFK(END)
&ZWINTTL = ' Scan Log '
)PROC
```

```
&scver='8 12 24'
&mmver='1 2 3 4 5 6 7 8 9 10 11 12'
&hover='0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23'
VER(&MM,listv,&mmver)
VER(&HO,listv,&hover)
VER(&SC,listv,&scver)
VPUT (LF YYYY MM DD HO SC A B C D) SHARED
)END
```

Luis Paulo Figueiredo Sousa Ribeiro
System Programmer
Edinfor (Portugal)

January 1995 – September 1997 index

Items below are references to articles that have appeared in *MVS Update* since January 1995. References show the issue number followed by the page number(s). Back-issues of *MVS Update* are available back to Issue 100 (January 1995). See page 70 for details.

Suggested articles for *MVS Update*

From time to time, subscribers contact us suggesting subjects for articles they would like to see covered in future issues of *MVS Update*. While we make every effort to obtain the appropriate material to satisfy these requests, we don't always meet with the success that we would like. In addition, a number of potential authors have expressed their willingness to contribute but are unsure what to write about.

Therefore, partly to inspire prospective authors and partly to see if anyone already has some existing material that might be appropriate, here is a list of subjects that readers have shown a positive interest in recently:

- The Year 2000 and MVS
- Open Edition/MVS
- Sysplex
- MVS internals
- Security
- Back-up and recovery
- User exits
- Performance
- Data sharing
- OS/390
- Tuning.

If you are interested in contributing an article on any of the above subjects or indeed any other MVS-related topic, and would like further details, or if there is an area that you would especially like to see covered in a future issue of *MVS Update*, please contact Xephon on +44 1635 38030 (telephone), stevep_xephon@compuserve.com (e-mail), or write to any of the addresses shown on page 2.

Candle Corporation has introduced MQSecure, a security tool for IBM MQSeries applications. MQSecure ensures that data sent through MQSeries applications or across an MQSeries network is protected using various types of security measure. These include user or machine authentication, which ensures that a message sent is from the system or user who is supposed to have sent it; message validation, which ensures that messages are not intercepted and changed while they are being sent from one system to another; non-repudiation, which prevents a person from denying having sent a message; and message privacy or encryption, which ensures that the content of the message is kept confidential. The product also offers node-to-node or end-to-end security levels.

For further information contact:
Candle Corporation, 2425 Olympic Boulevard, Santa Monica, CA 90404, USA
Tel: (310) 829 5800/(800) 843 3970
Fax: (310 582 4287 or
Candle Service Ltd, 1 Archipelago, Lyon Way, Frimley, Camberley, Surrey, GU16 5ER, UK
Tel: (01276) 681400
Fax: (01276) 681420.

* * *

Version 3.1 of Shadow Direct, its MVS-desktop connectivity product, has been unveiled by Neon Systems Inc. Specific enhancements which have been incorporated in this newest release include full-screen interactive symbolic support for debugging stored procedures, threaded pooling support for improved scalability, dynamic-to-static SQL logging, better usage monitoring, improved diagnostic facilities, and a transaction/data mapping facility that provides turnkey exploitation of existing and new IMS and CICS transactions without any host or client application programming. In addition, the vendor claims that improved data access support will result in up to a 90% reduction in CPU utilization.

For further information contact:
Neon Systems Inc, 14141 Southwest Freeway, Suite 6200, Sugar Land, TX 77478, USA
Tel: (713) 491 4200/(800) 505 NEON
Fax: (713) 242 3880 or
Neon Systems UK Ltd, Third Floor, Sovereign House, 26-30 London Road, Twickenham, Middx, TW1 3RW, UK
Tel: (0181) 607 9911
Fax: (0181) 607 9933.

* * *

MVS Solutions Inc has released Version 5 of ThruPut Manager, its automated batch turnaround management tool, specifically providing enhancements based on exploiting the latest batch technologies being introduced in OS/390 Version 2.

For further information contact:
MVS Solutions Inc, 8300 Woodbine Avenue, 4th Floor, Markham, Ontario, Canada L3R 9Y7
Tel: (905) 940 9404
Fax: (905) 940 5308

xephon