141 MVS update

June 1998

In this issue

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

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
Jaime Kaminski

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

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

Subscriptions and back-issues
A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £325.00 in the UK; $485.00 in the USA and Canada; £331.00 in Europe; £337.00 in Australasia and Japan; and £335.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

Looking inside other address spaces from TSO

INTRODUCTION

As a typical mainframe system programming professional, I find that I have an insatiable desire to know more about what is going on under the covers of any part of a system than is usually easily accessible by various DISPLA- type commands or even the commercially-available monitors.

It is always possible to undertake dumps, but the problem is that, by their very nature, dumps are large and unwieldy. In my experience, I frequently need to examine a very minor fragment, often a mere byte or two, especially in situations where the initial occurrence of a problem has been followed by some dump analysis.

Further, I desired a quick and simple method of looking at another address space while it was executing so that I could examine the contents of an address of interest.

THE SOLUTION

The REXX 'storage' function allows a program to do this kind of thing, but only in the address space of the calling program and, of course, the system common areas. The answer is to use an APF authorized Assembler program to do the actual cross memory access to the desired address space and pass the required data back to a REXX program for processing.

As I wanted to drive the entire process from REXX, I have developed an Assembler program, XMSLOOK, which is called from REXX, reads variables established in the REXX program, and writes the required information directly back into a REXX variable. Before the call to XMSLOOK, three REXX variables are initialized:

- The address space number that you want to examine (SYSAUTH.ASID)
- The address within that address space (SYSAUTH.ADDR)

- The length of the data to be returned (SYSAUTH.LENG).

After the call, the REXX variable XMEMSTOR will contain the desired data or an error message. The program accepts requests for up to 64 bytes of data because this was sufficient for my requirements, but increasing this value is not a problem. Alternatively, the program can be called multiple times. Indeed this is where I have made the most use of this program, because it lends itself to a recursive technique whereby a chain of control blocks can be chased through an executing job with a few lines of REXX code – in exactly the same way that the REXX 'storage' function will for common areas.

As mentioned, XMSLOOK must be authorized or else it will abend with an S047. Under TSO this means that it must be in an APF-defined library and also be defined in the relevant TSO PARMLIB member.

I have included a REXX sample program to illustrate how to set up these variables and call the program. The trickiest part of this is to get the address space number, because, even if you interrogate the same job (say a CICS address space) repeatedly, the address space number can change every time CICS is recycled.

There are various ways to discover the address space number from the job name, perhaps the easiest is just to look at the SDSF DA screen, or RMF monitor 2. But to automate the task, I have included a REXX subprogram called WHATASN, which can be called with a jobname and will return the address space number in the required format or a message informing you if the requested job is not currently executing on the system.

This function could easily be included in the Assembler program if desired, although it suits my purposes to have a REXX program do this because it is a function required by other systems. The method employed is to jump through common storage starting from the CVT.

This points to the ASVT, which is a table of pointers to all the Address Space Control Blocks (ASCBs) in the system, each of which contain pointers to the jobname and address space number of that block.

XMSLOOK

```
//ASMLINK  EXEC ASMACL,
//         PARM.L='LIST,LET,XREF,MAP,AMODE=31,AC=1'
//C.SYSPUNCH DD DUMMY
//C.SYSIN  DD *
*********************************************************************
**  CROSS MEMORY ACCESS FACILITY.                                  **
**  READ REXX VARIABLES SYSAUTH.ASID, ADDR, LENG AND VALIDATE.     **
**  XMS TO REQD DATA.                                              **
**  WRITE REXX VARIABLE XMEMSTOR.                                  **
*********************************************************************
         TITLE    'CROSS MEMORY ACCESS ROUTINE'
         LCLC     &MODULE
&MODULE  SETC     'XMSLOOK'
&MODULE  CSECT
&MODULE  AMODE    31
&MODULE  RMODE    24
*
         SAVE (14,12)
         USING XMSLOOK,12
         LR    12,15
         LR    14,13
         LA    13,SAVE
         ST    13,8(,14)
         ST    14,4(,13)
*
**  READ SYSAUTH.ASID
*
@ASID    MVC   NAME,=CL12'SYSAUTH.ASID' VARIABLE NAME
         MVC   NL,=F'12'              VARIABLE NAME LENGTH
         MVC   VL,=F'2'               VARIABLE LENGTH
         LINK  EP=IKJCT441,PARAM=(ECR,NP,NL,VP,VL,TK),VL=1
         L     9,VP                   POINTER TO VARIABLE
         MVC   XMASID,Ø(9)            GET VALUE FROM REXX
         MVC   REQASID,XMASID
*
**  READ SYSAUTH.ADDR - PAD TO FULLWORD
*
@ADDR    MVC   NAME,=CL12'SYSAUTH.ADDR'
         MVC   NL,=F'12'
         MVC   VL,=F'4'
         LINK  EP=IKJCT441,PARAM=(ECR,NP,NL,VP,VL,TK),VL=1
         L     8,VP
         MVC   XMADDR,Ø(8)
         L     9,VL
         LA    8,4
@AEQ4    CR    8,9
         BNH   @LENG
         LA    7,XMADDR
         MVC   3(Ø,7),2(7)
         MVC   2(Ø,7),1(7)
```

```
         MVC   1(Ø,7),Ø(7)
         MVI   Ø(7),X'ØØ'
         LA    9,1(9)
         B     @AEQ4
*
**  READ SYSAUTH.LENG - PAD TO FULLWORD
*
@LENG    MVC   NAME,=CL12'SYSAUTH.LENG'
         MVC   NL,=F'12'
         MVC   VL,=F'4'
         LINK  EP=IKJCT441,PARAM=(ECR,NP,NL,VP,VL,TK),VL=1
         L     8,VP
         MVC   XMLENG,Ø(8)
         L     9,VL
         LA    8,4
@LEQ4    CR    8,9
         BNH   @CHK64
         LA    7,XMLENG
         MVC   3(Ø,7),2(7)
         MVC   2(Ø,7),1(7)
         MVC   1(Ø,7),Ø(7)
         MVI   Ø(7),X'ØØ'
         LA    9,1(9)
         B     @LEQ4
*
**  CHECK LENGTH REQUESTED <= 64
*
@CHK64   L     9,XMLENG
         LA    8,65
         CR    8,9
         BH    @XMEM
@LENERR  MVC   XMSTOR,=CL64'XMSLOOK  - REQUESTED LENGTH > 64'
         B     @ERROR
*
**  CROSS MEMORY CALL
*
@XMEM    XR    2,2                   ZERO REG 2
         ESAR  2                     OBTAIN OUR ADDR SP NUMBER
         ST    2,OURASN              SAVE IT
*
         MODESET KEY=ZERO,MODE=PROB  KEY ZERO, PROB STATE
         BAL   14,@INXMEM
*
         XR    1,1                   REG 1 = Ø
         L     8,XMADDR              ADDRESS TO READ FROM
         L     2,XMLENG              LENGTH TO READ
         MVCP  XMSTOR(2),Ø(8),1
*
         BAL   14,@OUTXMEM
         MODESET KEY=NZERO,MODE=PROB KEY USER, PROB STATE
*
**  MOVE RETURNED DATA TO REXX VARIABLE XMEMSTOR
```

```
*
@UPDTRX  MVC   NAME,=CL8'XMEMSTOR'
         MVC   NL,=F'8'
         MVC   VL,XMLENG
         LA    8,XMSTOR
         ST    8,VP
         LINK  EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
         B     @FINISH
*
@ERROR   MVC   NAME,=CL8'XMEMSTOR'
         MVC   NL,=F'8'
         MVC   VL,=F'4Ø'
         LA    8,XMSTOR
         ST    8,VP
         LINK  EP=IKJCT441,PARAM=(ECU,NP,NL,VP,VL,TK),VL=1
*
@FINISH  L     13,SAVE+4
         RETURN (14,12),RC=Ø
*
**  SUBROUTINE - INTO CROSS MEMORY MODE  *************************
*
@INXMEM  LA    2,1                    REG 2 = 1
         AXSET AX=(2)                 AUTH INDEX = 1
         L     2,REQASN               INTO XMEM MODE
         SSAR  2
@INXEND  BR    14
*
**  SUBROUTINE - OUT OF CROSS MEMORY MODE  ************************
*
@OUTXMEM L     2,OURASN               OUT OF XMEM MODE
         SSAR  2
         XR    2,2                    REG 2 = Ø
         AXSET AX=(2)                 AUTH INDEX = Ø
@OUTXEND BR    14
*
**  STORAGE  ******************************************************
*
         DS    ØD
SAVE     DS    18F
XMSLPT   DS    F
OURASN   DS    F
REQASN   DS    ØF
         DC    XL2'Ø'
REQASID  DS    H
ASJOBN   DS    CL8
XMASID   DS    XL2
XMADDR   DS    F
XMLENG   DS    F
XMSTOR   DS    CL64
*
         DS    ØD
NAME     DS    CL12
```

```
NP       DC    A(NAME)
NL       DS    F
VL       DS    F
VP       DS    F
TK       DC    F'Ø'
ECR      DC    A(TSVERETR)
ECU      DC    A(TSVEUPDT)
*
         PRINT NOGEN
         IKJTSVT
         END
//*
//C.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.MODGEN,DISP=SHR
//L.SYSLMOD DD DSN=USER.LINKLIB,DISP=SHR    <------ APF authorized
//L.SYSIN DD *
  NAME XMSLOOK(R)
//
```

XMSLOOK REXX SAMPLE

```
/*------------------------------ REXX --------------------------------*/
/* Function   : Read storage via cross memory                         */
/*--------------------------------------------------------------------*/
injob = 'CICSJOB1'
call WHATASN injob
asn = result
if substr(asn,1,7) = 'WHATASN' then
  do
  say asn injob
  signal no_go
  end
sysauth.asid = asn
sysauth.addr = '9ØØØ'x
sysauth.leng = '1Ø'x
xmemstor = ''
say 'Calling parameters:' sysauth.asid sysauth.addr sysauth.leng
"CALL 'USER.LINKLIB(XMSLOOK)'"
error = substr(xmemstor,1,7)
if error = 'XMSLOOK' then
  do
  say 'Error in XMSLOOK' substr(xmemstor,9,56)
  exit Ø
  end
say 'Data returnd:' xmemstor
say 'Data in hex: ' c2x(xmemstor)
no_go:
exit Ø
```

WHATASN REXX SUBPROGRAM

```
/*------------------------------ REXX --------------------------------*/
/* Function   : Return a job's ASN.                                    */
/*---------------------------------------------------------------------*/
numeric digits 15
arg injob
addr = d2x(16)
cvt  = storage(addr,4)
addr = d2x(c2d(cvt) + c2d(x2c(Ø22c)))
asvt = storage(addr,4)
addr = d2x(c2d(asvt) + c2d(x2c(Ø2Ø4)))
asvu = storage(addr,4)
maxu = c2d(asvu)
addr = d2x(c2d(asvt) + c2d(x2c(Ø21Ø)))
asve = storage(addr,4)
do i = 1 to maxu
  unus = bitor(substr(asve,1,1),'7f'x)
  if unus = 'ff'x then
    nop
    else
    do
    addj = d2x(c2d(asve) + c2d(x2c(ØØac)))
    jbn  = d2x(c2d(storage(addj,4)))
    if jbn = Ø then
      do
      addj = d2x(c2d(asve) + c2d(x2c(ØØbØ)))
      jbn  = d2x(c2d(storage(addj,4)))
      end
    jobn = storage(jbn,8)
    if jobn = injob then
      do
      addn = d2x(c2d(asve) + c2d(x2c(ØØ24)))
      asn  = storage(addn,2)
      signal got_it
      end
    end
  addr = d2x(x2d(addr) + 4)
  asve = storage(addr,4)
end
asn = 'WHATASN  - Requested job not executing'
got_it:
return asn
```

Patrick Mullen
MVS Systems Consultant (Canada)

Dynamic LPA

LINK PACK AREA OVERVIEW

The Link Pack Area (LPA) is an area of virtual storage containing re-enterable routines such as TYPE 3 and 4 SVCs, access methods, and other re-enterable read-only system and user programs, which can be used concurrently by all tasks in the system. During system initialization, the contents supervision Resource Initialization Module (RIM) creates the MLPA, FLPA, PLPA, the initial LPA queue, and the LPA directory. Contents supervision uses the RMODE specification in the partitioned dataset (you cannot used PDSEs in the LPALST concatenation) directory entry for a module, to determine whether to load the module in the LPA above or below the 16MB line. This loads modules with RMODE ANY into virtual storage above the line, and modules with RMODE 24 below the line. At the end of the LPA initialization, the following CVT fields are set:

- CVTPLPAS, CVTPLPAE, CVTEPLPS, and CVTEPLPE with the low and high-end addresses of the PLPA and EPLPA.
- CVTFLPAS, CVTFLPAE, CVTEFLPS, and CVTEFLPE with the low and high-end addresses of the FLPA and EFLPA
- CVTMLPAS, CVTMLPAE, CVTEMLPS, and CVTEMLPE with the low and high-end addresses of the MLPA and EMLPA

LINK PACK AREA (LPA)

The LPA contains:

- LPA directory – a record of every program in the PLPA. The directory is created during nucleus initialization and consists of LPA Directory Entries (LPDEs) for each entry point in the PLPA modules. LPDEs for major entry points contain a CDE (Contents Directory Entry) and a compressed extent list. LPDEs for alias entry points contain the name of a related major entry point instead of a compressed extent list. After the last LPA module has been loaded into the LPA, the Link Pack Area initialization routines allocate storage for the LPA directory, move the directory to the next address lower than the PLPA, and place a pointer to the

directory in field CVTLPDIR. The directory available flag, CVTDICOM, is then set to enable the system to use the LPA directory.

- Modified LPA – containing modules that are to be temporarily appended to the PLPA as additions to or replacements for existing modules. The MLPA modules are represented by CDEs on the LPA queue and are used in preference to identified copies of modules in the PLPA.
- Fixed LPA – an optional extension to the LPA. The FLPA is used to improve system performance or to satisfy a module's time dependencies. If a fixed LPA is present, contents supervision searches it before the pageable or modified LPA. Fixed LPA modules are represented by CDEs on the LPA queue and are used in preference to identical paged copies of modules in the PLPA.
- LPA Queue – a record of all FLPA, MLPA, and possibly some currently active PLPA modules. The elements on the queue are Contents Directory Entries (CDEs), one per entry point. All PLPA modules are represented by LPDEs in the LPA directory. Contents supervision does not build or queue CDEs for PLPA modules. Contents supervision creates CDEs on the LPA queue when required to identify alternate entry points for LPA modules. The LPA queue resides in subpool 245.
- LPALST Table (LPAT) – a list that names the datasets included in the LPALST concatenation. The ordering of the entries in the LPAT corresponds to the order in which the datasets have been concatenated. Once built, the LPAT cannot be changed and has read-only access. The LPALST concatenation can have up to 255 extents. If the maximum number of extents is exceeded, the system truncates the LPALST concatenation.
- Quick Start Records – during LPA cold start initialization, the quick start record (QSRCD), the extended quick start record (EQSRD), and the quick start record extensions (XQSRD) are created and written to the PLPA dataset. The QSRCD contains the fields CVTVVMDI, CVTPLPAS, and CVTPLPAE. The EQSRD contains all the PLPA XPT (External Page Table) information, and the XQSRD contains the CVTEPLPS, CVTEPLPE, and CVTRWNS (starting virtual address of the read/write nucleus). During subsequent warm starts, the QSRCD, EQSRD, and

XQSRDs are read from the PLPA dataset to rebuild the PLPA and the EPLPA.

A recent enhancement to the LPA creation is the addition of the SYSLIB statement to the PROGxx PARMLIB member. This allows you to change the default system datasets placed at the beginning of the LPALST concatenation. The following example places SYS9.LPALIB at the beginning of the LPALST concatenation:

```
SYSLIB(SYS9.LPALIB)
```

LPA CONTROL BLOCK OVERVIEW

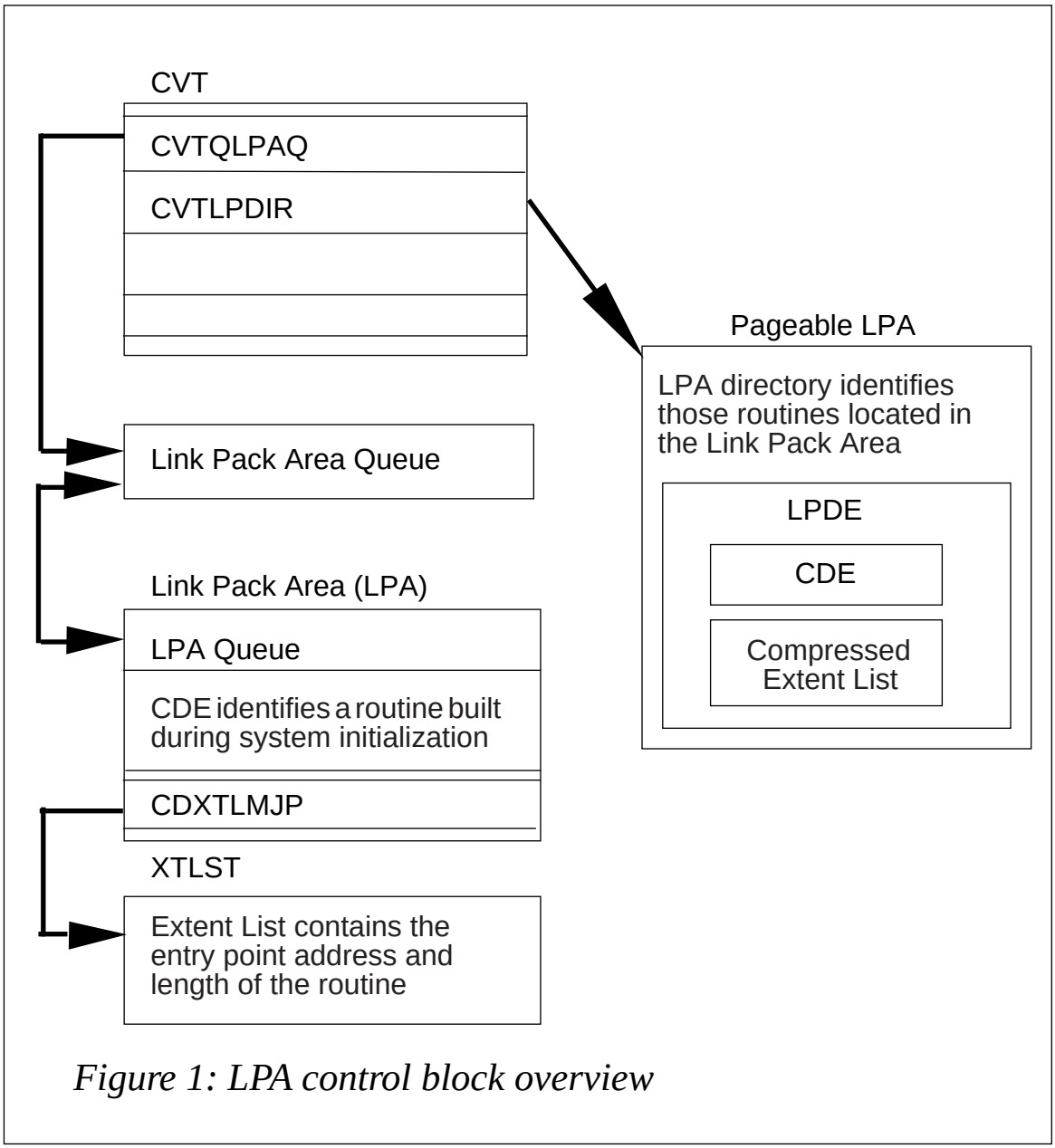

Figure 1: LPA control block overview

DYNAMIC LPA

- OS/390 Version 2 Release 4 introduced the Dynamic LPA Facility (DLPA). Prior to this release of OS/390 or any release of MVS, a vendor product or an installation-written utility would have been required to dynamically add a new or replacement LPA module (or as a last resort, an IPL would have been required). The standard technique employed by most vendors is to load a replacement module into common storage and then alter the Link Pack Directory Entry (LPDE) to point to the new module (see Figure 1).
- Dynamic LPA and PROGxx – IBM have employed the PROGxx PARMLIB member to define the Dynamic LPA Facility. A new statement, LPA, has been defined and is used to specify:
  - Modules that are to be added to the LPA following an IPL.
  - Modules that are to be deleted from the LPA following an IPL.
  - Threshold value for the minimum amount of CSA storage that must still be available after an ADD operation.

The syntax for the LPA statement is as follows:

```
LPA ADD,MODNAME(modname,modname...) | MASK(mask)
        DSNAME(dsname | LNKLST | LINKLIST)
        [FIXED|PAGEABLE]
        [PAGEPROTPAGE]

LPA DELETE, MODNAME(modname)
        FORCE(YES)
        [CURRENT|OLDEST]

LPA CSAMIN
        (below,above)
```

Where:

- MODNAME – specifies a 1 to 8 character LPA module name or alias. If modname is not specified, MASK must be specified.
- MASK – specifies a 1 to 8 character mask that is to be applied to all the members of the specified dataset. If mask is not specified, MODNAME must be specified.

- DSNAME – specifies a 1 to 44 character dataset name, which contains the module(s) or alias(es). When modname is specified, DSNAME(LNKLST) or DSNAME(LINKLIST) can also be specified. This will cause the system to use its normal search sequence instead of a particular dataset.
- CSAMIN – specifies a minimum amount of CSA and ECSA that must remain after a module is added to LPA. The default value is set to (0,0). It is probably advisable to increase the ECSA values in PARMLIB before implementing the Dynamic LPA facility.
- FIXED/PAGEABLE – specifies whether the modules are to be placed in fixed or pageable storage.
- PAGEPROTPAGE – specifies that only full pages occupied by the module should be page protected. For example, if a 9K module is loaded onto a new page, it will occupy two full pages and a partial page (8K on the first two pages and 1K on the third page). If PAGEPROTPAGE is not specified, all three pages will be page protected thus causing an increase in storage utilization. When PAGEPROTPAGE is specified, it is always possible that a storage overlay at the beginning or end of the load module can occur.
- FORCE(YES) – specifies that the system can have no knowledge of any code that is currently executing within the specified module. Use this parameter with caution.
- CURRENT/OLDEST – CURRENT specifies that the current copy of the load module is to be deleted. OLDEST specifies that the oldest dynamic copy is to be deleted.

OPERATOR COMMAND INTERFACE

The following operator commands control the Dynamic LPA facility:

```
SET PROG=xx
```

SET PROG=xx causes the PROGXX parmlib member containing the LPA definitions to take effect.

```
SETPROG LPA
```

SETPROG LPA provides an operator interface to add or delete Dynamic LPA modules. This interface can be used instead of creating a PROGxx PARMLIB member. The command syntax is as follows:

```
SETPROG LPA,{ADD,[MODNAME=(modname...,modname) | MASK=mask}
             {    ,DSNAME=[dsname | LNKLST]               }
             {    [,FIXED] [,PAGEPROTPAGE]                }
             {                                            }
             {DELETE,MODNAME=(modname...,modname)         }
             {    FORCE=YES [CURRENT | OLDEST]            }
             {                                            }
             {CSAMIN=(below,above)                        }

D PROG,LPA[,MODNAME=modname]
         [,CSAMIN         ]
```

Where:

– MODNAME= displays the current entry point and load point/ length information for the requested LPA module.

– CSAMIN displays the current CSA and ECSA minimum values.

MONITORING DYNAMIC LPA PROCESSING

IBM has provided an exitname, CSVDYLPA, that can be used as a notification mechanism for ISV products to update their own internal control blocks when a module is added or deleted from the Dynamic LPA facility. The CSVDYLPA exit receives control when a Dynamic LPA service request is issued via:

- The CSVDYLPA macro.
- The SETPROG LPA operator command.
- When an LPA statement within PROGxx is referenced by the SET PROG=xx operator command.

The exit can be installed by using the Dynamic Exit Services macro, CSVDYNEX. The exit receives control in the following environment:

- Supervisor state KEY 0
- Cross memory mode of PASN=HASN=SASN
- AMODE 31
- Primary ASC mode

- Enabled for I/O and external interrupts
- No locks held
- Parameter areas in the primary address space
- Receives control in the Master address space for SETPROG or SET PROG operator commands
- Receives control in the address space that issued the CSVDYLPA REQUEST=ADD request
- ENQ resource SYSZCSV.CSVDYLPA held in exclusive mode.

SMF RECORDING

Whenever a module is added or deleted using the Dynamic LPA facility, an SMF type 90 (system status) subtype-31 record is produced. The Dynamic LPA Management Section in the SMF record contains the following information:

- Whether the LPA request is ADD or DELETE.
- Requestor ID provided via CSVDYLPA.
- Time of activation.
- Console ID of issuer of the LPA request.
- The security product user token of the issuer of the LPA request.
- MODENTRIES which are mapped by the DSECT LPMEA within macro CSVLPRET.

RACF RESOURCES

There are two new RACF resources contained within the FACILITY CLASS for Dynamic LPA. These are:

```
CSVDYLPA.ADD.modname
Access Level Required= UPDATE
```

and:

```
CSVDYLPA.DELETE.modname
Access Level Required= UPDATE
```

UPDATED OS/390 CONTROL BLOCKS FOR DYNAMIC LPA

Two new pointers have been added to the ECVT to address the Dynamic LPA CDEs. The CDEs can be referenced by running the following control block chain:

```
x'10'            ====>   CVT Address
CVTECVT(x'8C')   ====>   ECVT
ECVTDLPF(x'228)  ====>   Address of the first CDE on the Dynamic  LPA queue
ECVTDLPL(x'22C') ====>   Address of the last CDE on the dynamic LPA queue
```

IBM recommends using the following supported interfaces to obtain Dynamic LPA module information:

```
CSVINFO
CSVQUERY
```

The CDE chain that is anchored by ECVTDLPF will eventually join the chain of LPA CDEs that are anchored by CVTQLPAQ.

DYNAMIC LPA API

IBM has provided the CSVDYLPA macro so that Assembler programmers can interface to the Dynamic LPA facility. The CSVDYLPA macro provides the following functions:

- REQUEST=ADD – this function allows you to add one or more modules or aliases to the LPA.
- REQUEST=DELETE – this allows you to delete from the LPA one or more modules or aliases that had previously been added by the Dynamic LPA service.
- REQUEST=QUERYDYN – this allows you to query whether the ADD or DELETE functions are available.

For ADD and DELETE functions, the following macro should be included in the source code to generate the equate symbols for the return and reason codes, and to obtain a mapping of the I/O area provided via the MODINFO area. The MODINFO area is a contiguous array of entries, mapped by the LMPEA (for ADD requests) or LMPED (for delete requests) DSECT:

```
CSVLPRET
```

For ADD or DELETE requests, the caller may hold the following system ENQ resource with exclusive access:

```
     RNAME=   SYSZCSV
     QNAME=   CSVDYLPA
```

While this ENQ resource is held, any other requests to use the CSVDYLPA service to ADD or DELETE LPA modules will be delayed. The example below will dynamically add a module to the LPA:

```
*****************************************************************************
* Check if the Dynamic LPA facility is available
*
*****************************************************************************
               STORAGE OBTAIN ..........
               LA    R3,16(Ø,Ø)
               USING CVT,R3
               TM    CVTOSLV2,CVTDYLPA
               BZ    DYNAMIC_LPA_NOT_AVAILABLE
               CSVDYLPA REQUEST=QUERYDYN,
X
                     DYNFUNC=DLPA_AVAILABLE
               XR    R4,R4
               ICM   R4,B'ØØØ1',DLPA_AVAILABLE
               BZ    DYNAMIC_LPA_NOT_AVAILABLE
*****************************************************************************
* Setup the MODINFO area for module 'REMSLPA'.                              *
* Page Fix the Module                                                       *
* Locate the module using SYS2.APFLIB                                       *
* Issue  CSVDYLPA REQUEST=ADD to dynamically add the LPA module             *
*****************************************************************************
               LA    R6,MODULE_LIST
               USING LMPEA,R6
               XC    MODULE_LIST(LMPEA_LEN),MODULE_LIST
               MVC   LPAMODS#,=F'1'
               MVC   LPMEANAME,=CL8'REMSLPA'
               OI    LPMEAINPUTFLAGSØ,LMPEAFIXED
               MODESET MODE=SUP
               CSVDYLPA REQUEST=ADD,                                      X
                     MODINFOTYPE=MEMBERLIST,                              X
                     MODINFO=MODULE_LIST,                                 X
                     NUMMOD=LPAMODS#,                                     X
                     RETCODE=DRETCODE,                                    X
                     RSNCODE=DRSNCODE,                                    X
                     APFREQUIRED=YES,                                     X
                     DSNAME=APFLIB,                                       X
                     SECMODCHECK=NO,                                      X
                     REQUESTOR=REMSID,                                    X
                     MF=(E,DYNLPA,COMPLETE)
               LTR   R15,R15
               BNZ   DLPA_ADD_FAILED
*****************************************************************************
* Process the CSVDYLPA REQUEST=ADD as required
```

```
*
*********************************************************************
*
DYNAMIC_LPA_NOT_AVAILABLE  EQU *
DLPA_ADD_FAILED        EQU *
RETURN   EQU   *
               MODESET MODE=PROB
               STORAGE RELEASE ..........
               XR      R15,R15
               PR
REMSID   DC    CL16'REMS_DLPA_EX'
APFLIB   DC    CL44'SYS2.APFLIB'
               LTORG
WORKAREA DSECT
DLPA_AVAILABLE DS X
LPAMODS#       DS F
DRETCODE       DS F
DRSNCODE       DS F
MODULE_LIST    DS CL(LPMEA_LEN)
               CSVDYLPA MF=(L,DYNLPA,ØF)
WORKLEN  EQU   *-WORKAREA
               CSVLPRET
               CVT     DSECT=YES
               END     ADDLPA
```

R F Perretta
Senior Systems Programmer (UK)

Customizing the TSO/E Version 2 logon command

INTRODUCTION

With TSO/E Version 2, IBM introduced, amongst other enhancements, two new LOGON exits:

- The authorized logon pre-prompt exit, IKJEFLD1,
- The post-prompt logon exit, IKJEFLD3.

These exits make it much easier to implement modifications to the LOGON command than it was with TSO/E Version 1. In our case it

allowed us to implement a restriction on the CPU time limit for foreground TSO sessions – by default these are not CPU-time limited. We achieved this by adding a TIME(nn) parameter to the LOGON command – where nn is the desired CPU time limit in minutes. Our current implementation contains a default value of 10 minutes and imposes an absolute upper limit of 30 minutes.

Partially as a debugging aid during the development of these exits, a DEBUG parameter was also added; this causes IKJEFLD1 to display the input and rebuilt logon commands on the console, and IKJEFLD3 to do the same with the original (system-built) and rebuilt logon JCL, as will be described below.

IKJEFLD1

IKJEFLD1 is called by the LOGON processor before initiating any dialogue with the user, ie after the VTAM LOGON command has been entered but before the full screen log-on panel is displayed. The processing in our IKJEFLD1 is as follows:

1 An Attention Exit routine is defined via a STAX macro. If an attention is detected, the exit sets a flag that is checked at appropriate points in the mainline code.

2 A command buffer is built in the format required by IKJPARS and the text of the user-entered logon command copied in. This step is required because the command buffer supplied to IKJEFLD1 does not have the header required by IKJPARS.

3 The log-on command is parsed via a call to IKJPARS, using a Parameter Control List (PCL) that contains all the standard log-on command parameters *and* all the locally added ones (for example TIME(nn) and DEBUG in our case).

4 If the TSO user-id was not supplied by the user, it is prompted for via a TPUT/TGET dialogue. This dialogue may be terminated by an attention or a terminal-disconnection; in either case processing jumps to step 12 below.

5 If the DEBUG parameter was specified, the input logon command is displayed on the console, eg:

```
LGN110D logon userid time(15) debug
```

6 The ASVT is scanned for active TSO sessions for the same userid. This can happen if a user is logged-on on one terminal and tries to log on to another – the processing in TSO and JES that prevents duplicate TSO sessions does not come into force until *after* the log-on dialogue has completed. If a duplicate session is found, a message is sent via TPUT to the other session informing it of the (possibly malicious) attempt to log on as that user. The current session is also informed that another session for the user is already active.

7 A Logon Communication Block (LCB) is allocated in subpool 130 and initialized. Its address is stored in the exit-to-exit communication word in the IKJEFLD1 parmlist. The contents of this word will be passed by LOGON to IKJEFLD3.

8 If the DEBUG parameter was specified, the debug flag bit is set in the LCBFLAG1 field.

9 The session CPU time limit is determined, either from the value supplied by the user in the TIME(nn) parameter, or by default. If the user did supply a value, it is capped at a maximum value of 30 minutes. The resulting value is stored as a 4-character EBCDIC string, eg '0010', in the LCBTIME field of the LCB.

10 The log-on command is rebuilt from the parsed values, *excluding* the local parameters. The following points should be noted:

- If the user specified a password (ie entered 'LOGON USERID/PASSWORD'), the password is not placed in the rebuilt command – LOGON will require it to be entered on the log-on panel anyway.
- The IBM defaults of 'NOMAIL' and 'NONOTICES' are overridden by setting the MAIL and NOTICES control switches to ON, unless the user specified 'NOMAIL' and/or 'NONOTICES'. This avoids the necessity of adding the 'MAIL' and 'NOTICES' parameters to the rebuilt command.
- Unless the user specified 'RECONNECT', the 'NORECONNECT' parameter is appended to the rebuilt command. Leaving the RECONNECT control switch set OFF is not sufficient to clear the memory of a prior 'LOGON RECONNECT' request.

11. If the DEBUG parameter was specified, the rebuilt log-on command is displayed on the console, eg:

```
LGN111D LOGON userid NORECONNECT
```

12. The attention flag is checked; if found to be set, the 'Immediate Disconnect' flag is set in the control switch parameter.

13. The Parameter Descriptor List (PDL) is freed and the attention exit removed.

14. IKJEFLD1 cleans up and returns to LOGON.

IKJEFLD3

IKJEFLD3 is called by the LOGON processor after the full screen dialogue with the user has completed. As indicated above, one of the parameters it is passed is the contents of the exit-to-exit communication word, into which IKJEFLD1 has stored the address of the LCB. The processing in IKJEFLD3 is as follows:

1 The address of the LCB is taken from the exit-to-exit communication word. Note that there are certain situations in which it will not be set because some paths through LOGON bypass IKJEFLD1, most often during processing of LOGON commands issued from existing TSO sessions. In the absence of an LCB the same default CPU time limit that was used in IKJEFLD1 will be assumed.

2 If LCB is present, and if the debug flag is set, the supplied log-on JCL is displayed on the console via WTOs:

```
//userid   JOB 'DEPTnnnn',REGION=nnnnK
//procname EXEC procname
```

3 The supplied JOB card is scanned and the account number and REGION value extracted. In our case, the accounting information is in the form of a 4-character department name and a 4-digit account number; this is of course not in the standard JCL format.

4. A new job card is built, incorporating the account number converted to standard JCL format, the TIME parameter from the LCB, the REGION parameter, and also the 'programmer name', obtained from the RACF ACEE. The JCL buffer is rebuilt with the new JOB card (which is now two lines long), and a copy of the

original EXEC card. If the LCB is present and the debug flag is set the new JCL is displayed on the console :

```
//userid   JOB (nnnn,DEPT),'Programmer Name',
//         TIME=(nnnn,OO),REGION=nnnnK
//procname EXEC procname
```

5 The LCB is freed and IKJEFLD3 returns. LOGON then submits the new JCL to the TSOINRDR to establish the TSO session. Note that by adding the programmer name to the JOB card, the 'userid ON TSOINRDR' console message that results will be annotated with this name – a useful feature if the TSO user-ids are not easily recognisable.

IKJEFLD3 is designed to be fail-safe – if at any point a problem is detected, it simply gives up and lets the original, system-built JCL be used.

OPERATIONAL ENVIRONMENT

IKJEFLD1 and IKJEFLD3 may be placed either in the PLPA, via SYS1.LPALIB or a member of the LPA library concatenation, or in the Linklist, via SYS1.LINKLIB or a member of the Linklist concatenation. The choice of location is a trade-off between (possibly) better performance if the modules are in the PLPA, and the flexibility that a Linklist location provides. The latter may be a better choice if it is likely that the default TIME limit will have to be changed on a regular basis – such a change requires the modules to be reassembled and relinked. The LCB macro should be placed in SYS1.MACLIB or another suitable macro library that is accessible when assembling the routines.

The code presented here was originally written for use with TSO/E V2.3 on an MVS/ESA 4.2.2 system, and has since been ported to TSO/E V2.4 on an MVS/ESA 5.1.0 system where no changes were required.

IKJEFLD1

```
         TITLE 'IKJEFLD1: TSO LOGON PRE-PROMPT EXIT'
*********************************************************************
*  SUBROUTINE IKJEFLD1                                              *
*  ———————————                                                      *
```

```
*  This is the TSO LOGON pre-prompt exit.  It is called by the LOGON  *
*  processor before initiating any dialogue with the user. This ver-  *
*  sion provides support  for  the local TIME() and DEBUG parameters  *
*  that we have added to the LOGON command.  The TIME parameter lets  *
*  the user specify  the  CPU  time limit for the session; the DEBUG  *
*  parameter causes this routine, and the post-prompt exit, IKJEFLD3  *
*  to WTO pertinent data to the console.                              *
*                                                                     *
*  The input command is parsed by  a  call to IKJPARS and any syntax  *
*  errors etc corrected  via  the standard IKJPARS prompt mechanism.  *
*  If the user hits the ATTENTION key during this process, the log-on *
*  is aborted.   Otherwise,  the  ASVT is then scanned to see if the  *
*  logging-on user is already logged on; if so both he and the other  *
*  sessions are informed of this fact.   (NB: only specially author-  *
*  ized users can have multiple sessions).                            *
*                                                                     *
*  A Logon Communication Block (LCB) is then created and its address  *
*  stored in the exit-to-exit communication word in the input param-  *
*  eter list. If the user specified a time value it is capped at the  *
*  the maximum allowed value  (currently 3Ø mins)  and stored in the  *
*  LCB. If no time value was specified, the default value of 1Ø mins  *
*  is stored. If DEBUG was specified, a flag is set in the LCB.  The  *
*  LCB is used and then deleted by the post-prompt exit, IKJEFLD3.    *
*                                                                     *
*  Finally,  the input command is rebuilt,   excluding any TIME() or  *
*  DEBUG parameters.  During this process, both the MAIL and NOTICES  *
*  flags are set, overriding the IBM defaults. If the user wishes to  *
*  suppress mail and/or notices,   he must  explicitly  specify  the  *
*  NOMAIL and/or NONOTICES options,  or  set the flags on the log-on  *
*  panel when it is displayed.    A default of 'NORECONNECT' is also  *
*  enforced - the system remembers (via RACF) a 'RECONNECT'  request  *
*  and does not forget it until 'NORECONNECT' is specified.  In this  *
*  case 'RECONNECT' must be added to the rebuilt command - it is not  *
*  enough to leave the reconnect control switch off.                  *
*                                                                     *
*  ENVIRONMENT                                                        *
*                                                                     *
*    State    : Supervisor                                          *
*    Key      : 8                                                   *
*    APF      : Unauthorized                                        *
*    AMODE    : 31                                                  *
*    RMODE    : ANY                                                 *
*    Location : Linklist or PLPA                                    *
*                                                                     *
*  REGISTERS ON ENTRY :                                               *
*                                                                     *
*    R1  - @(STANDARD EXIT PARAMETER LIST)                          *
*          + Ø : @(COMMAND BUFFER PE)                               *
*          + 4 : @(NEW COMMAND BUFFER PE) (Not valid for IKJEFLD1)  *
*          + 8 : @(UPT PE)                                          *
*          +12 : @(ECT PE)                                          *
*          +16 : @(PSCB PE) (Not valid for IKJEFLD1)                *
```

```
*          +2Ø : @(EXIT-TO-EXIT COMMUNICATION WORD PE)                *
*          +24 : @(EXIT REASON CODE PE)                                 *
*          +28 : RESERVED                                               *
*          +32 : RESERVED                                               *
*          +36 : @(CONTROL SWITCH PE)                                   *
*          +4Ø : @(UID PE) )                                            *
*          +44 : @(PASSWORD PE)                                         *
*          +48 : @(ACCOUNT NUMBER PE)                                   *
*          +52 : @(PROCEDURE NAME PE)                                   *
*          +56 : @(REGION PE)                                           *
*          +6Ø : @(JCL BUFFER PE)                                       *
*          +64 : @(NEW PASSWORD PE)                                     *
*          +68 : @(SYSTEM ATTRIBUTE BITS PE)                            *
*          +72 : @(USER ATTRIBUTE BITS PE)                              *
*          +76 : @(GENERIC UNIT PE)                                     *
*          +8Ø : @(CANCEL ECB PE)                                       *
*          +84 : @(PERFORMANCE GROUP NUMBER PE)                         *
*          +88 : @(SYSOUT DESTINATION PE)                               *
*          +92 : @(GROUP DESTINATION PE)                                *
*          +96 : @(SUBMIT HOLD CLASS PE)                                *
*         +1ØØ : @(SUBMIT CLASS PE)                                     *
*         +1Ø4 : @(SUBMIT MSGCLASS PE)                                  *
*         +1Ø8 : @(SYSOUT CLASS PE)                                     *
*         +112 : @(FIRST COMMAND PE)                                    *
*         +116 : @(RBA PE)                                              *
*         +12Ø : @(SECLABEL PE)                                         *
*         +124 : @(CONSOLE PROFILE PE)                                  *
*         +128 : @(PRIMARY LANGUAGE ID PE)                              *
*         +132 : @(SECONDRY LANGUAGE ID PE)                             *
*    R13 - @(SAVEAREA)                                                  *
*    R14 - RETURN ADDRESS                                               *
*    R15 - ENTRY ADDRESS                                                *
*                                                                       *
*  REGISTERS ON RETURN :                                                *
*                                                                       *
*    RØ-R14 - AS AT ENTRY                                               *
*    R15 - RETURN CODE                                                  *
*           Ø : CONTINUE NORMAL PROCESSING                              *
*************************************************************************
         EJECT
IKJEFLD1 CSECT
IKJEFLD1 AMODE 31
IKJEFLD1 RMODE ANY
RØ       EQU   Ø
R1       EQU   1                        * PARM LIST ADDRESS ON ENTRY
R2       EQU   2                        * @(EXIT COMM. PARM ENTRY)
R3       EQU   3                        * WORK REGISTER
R4       EQU   4                        * WORK REGISTER
R5       EQU   5                        * WORK REGISTER
R6       EQU   6                        * WORK REGISTER
R7       EQU   7                        * @(ASCB)
R8       EQU   8                        * @(ASVT ENTRY)
```

```
R9       EQU   9                        * @(ASVT) / @(LCB)
R1Ø      EQU   1Ø                       * @(PPL) / @(PDL)
R11      EQU   11                       * @(PARMLIST)
R12      EQU   12                       * BASE REGISTER
R13      EQU   13                       * SAVEAREA/WORKAREA ADDRESS
R14      EQU   14                       * RETURN ADDRESS
R15      EQU   15                       * ENTRY ADDRESS
*
         USING *,R15                    * ADDRESSABILITY
         B     START                    * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)        * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    C'IKJEFLD1'
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                      * FINISHED WITH R15
         DS    ØF                       * ALIGN TO FULL WORD BOUNDARY
********************************************************************
* ADDRESSABILITY AND LINKAGE - REENTRANT FORM
********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER
         USING IKJEFLD1,R12             * AND DEFINE ADDRESSIBILITY
*
         LR    R11,R1                   * LOAD PARMLIST ADDRESS
         USING PARMLIST,R11             * PARMLIST ADDRESSABILITY
*
         GETMAIN RU,LV=WKALEN,LOC=(BELOW,ANY)
*
         LR    R2,R1                    * @(WORKAREA)
         LA    R3,WKALEN                * L'WORKAREA
         LR    R4,R2                    * @(WORKAREA)
         SR    R5,R5                    * ZERO PAD, LENGTH
         MVCL  R2,R4                    * CLEAR WORKAREA
*
         ST    R13,4(R1)                * STORE HSA ADDRESS
         ST    R1,8(R13)                * STORE LSA ADDRESS
*
         LR    R13,R1                   * R13 = OUR SAVEAREA ADDRESS
         USING WORKAREA,R13             * WORKAREA ADDRESSABILITY
         EJECT
********************************************************************
* MAIN CODE
********************************************************************
* IF THE LOG-ON PROCESSOR FAILED  TO  GET THE SYSIKJUA ENQUEUE FOR THE
* USER-ID (IE USER ALREADY LOGGED ON), FAILED TO GET ANY OTHER RESOURCE
* OR HAS SUFFERED AN ABEND, RETURN IMMEDIATELY.
*
```

```
         L     R15,ACTLSW               * @(CONTROL SWITCH ENTRY)
         USING PARMENT,R15              * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR               * R1 = @(CONTROL SWITCHES)
         DROP  R15                      * FINISHED WITH PARM ENTRY
*
         TM    Ø(R1),X'8Ø'              * USERID ENQUEUE FAILURE ?
         BO    RETURN                   * YES; NOTHING TO DO HERE
*
         TM    Ø(R1),X'2Ø'              * OTHER RESOURCE FAILURE ?
         BO    RETURN                   * YES; NOTHING TO DO HERE
*
         TM    1(R1),X'Ø2'              * PRIOR ABEND ?
         BO    RETURN                   * YES; NOTHING TO DO HERE
         EJECT
*********************************************************************
* SET UP AN ATTENTION EXIT
*********************************************************************
         MVI   ATTNFLAG,X'ØØ'            * CLEAR ATTENTION FLAG
         MVC   STAXLIST(STAXLEN),STAXDUM  * MOVE STAXLIST INTO WORKAREA
         MVC   STAXOBUF(LLGN1Ø5I),LGN1Ø5I * MOVE STAX O/P MSG TO WKAREA
*
         ST    R12,ATTNPARM             * SAVE BASE REG FOR ATTNEXIT
         LA    R1,ATTNFLAG              * SAVE @(ATTNFLAG) ...
         ST    R1,ATTNPARM+4            * ... FOR ATTNEXIT
*
         STAX  ATTNEXIT,
+
               USADDR=ATTNPARM,
+
               OBUF=(STAXOBUF,LLGN1Ø5I),
+
               REPLACE=NO,
+
               MF=(E,STAXLIST)
*
         LTR   R15,R15                  * ATTN EXIT ESTABLISHED ?
         BNZ   STAXERR                  * NO
         EJECT
*********************************************************************
* PARSE THE INPUT LOGON COMMAND.
*********************************************************************
* SET UP THE PARSE PARAMETER LIST
*
BUILDPPL EQU   *
         LA    R1Ø,PPLIST               * PARSE PARAMETER LIST ...
         USING PPL,R1Ø                  * ... ADDRESSABILITY
*
         L     R15,AUPT                 * @(UPT PARM ENTRY)
         USING PARMENT,R15              * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR               * R1 = @(UPT)
         ST    R1,PPLUPT                * STORE @(UCT) IN THE PPL
         DROP  R15                      * FINISHED WITH PARM ENTRY
```

```
*
         L     R15,AECT                  * @(ECT PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR                * R1 = @(ECT)
         ST    R1,PPLECT                 * STORE @(ECT) IN THE PPL
         DROP  R15                       * FINISHED WITH PARM ENTRY
*
         LA    R1,CPECB                  * STORE @(ECB) ...
         ST    R1,PPLECB                 * ... IN THE PPL ...
         XC    CPECB,CPECB               *     ... AND CLEAR THE ECB
*
         L     R1,APCL                   * STORE @(@(PCL)) ...
         ST    R1,PPLPCL                 * ... IN THE PPL
*
         LA    R1,APDL                   * STORE @(@(PDL)) ...
         ST    R1,PPLANS                 * ... IN THE PPL
         XC    APDL,APDL                 *     ... AND CLEAR @(PDL)
*
         LA    R1,CMDBUF                 * STORE @(COMMAND BUFFER) ...
         ST    R1,PPLCBUF                * ... IN THE PPL
*
         XC    PPLUWA,PPLUWA             * NO UWA
         XC    PPLVEWA,PPLVEWA           * NO VEWA
*-------------------------------------
* THE SUPPLIED INPUT COMMAND BUFFER LACKS THE STANDARD HEADER EXPECTED
* BY IKJPARS, SO WE BUILD A COPY THAT INCLUDES THIS HEADER.  NOTE THAT
* WE ASSUME THE INPUT BUFFER CONTAINS AT LEAST THE TEXT 'LOGON'.
*-------------------------------------
         L     R15,ACMDBUF               * @(INPUT BUFFER PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R2,PARMLEN                * R2 = L'(INPUT BUFFER)
         L     R3,PARMADR                * R4 = @(INPUT BUFFER)
         DROP  R15                       * FINISHED WITH PARM ENTRY
*
         LR    R4,R2                     * L'COMMAND ...
         BCTR  R4,Ø                      * ... FOR EXECUTE
         EX    R4,COPYCMD                * COPY COMMAND TO BUFFER
         LA    R4,5(R4)                  * STORE L'BUFFER ...
         STH   R4,CMDBUF                 * ... IN BUFFER HEADER
*
         SR    R4,R4                     * ZERO OFFSET COUNTER
         LA    R5,CMDBUF+4               * @(COMMAND)
         LR    R6,R5                     * @(END+1 OF ...
         AR    R6,R2                     *   ... COMMAND)
CMDSCAN1 EQU   *
         CLI   Ø(R5),C' '                * DELIMITER ?
         BE    CMDSCAN3                  * YES
         LA    R4,1(R4)                  * INCREMENT OFFSET COUNTER
         LA    R5,1(R5)                  * INCREMENT SCAN POINTER
         CR    R5,R6                     * DROPPED OFF END YET ?
         BL    CMDSCAN1                  * NOT YET SO KEEP SCANNING
         B     CMDSCAN4                  * YES - NO OPERANDS
```

```
*
CMDSCAN2 EQU   *
         CLI   Ø(R5),C' '               * DELIMITER ?
         BNE   CMDSCAN4                 * NO, IE OPERAND FOUND
CMDSCAN3 EQU   *
         LA    R4,1(R4)                 * INCREMENT OFFSET COUNTER
         LA    R5,1(R5)                 * INCREMENT SCAN POINTER
         CR    R5,R6                    * DROPPED OFF END YET ?
         BL    CMDSCAN2                 * NOT YET SO KEEP SCANNING
*
CMDSCAN4 EQU   *
         STH   R4,CMDBUF+2              * INSERT OFFSET INTO BUFFER
         B     CALLPARS                 * GO AND PARSE THE COMMAND
*
         DROP  R1Ø                      * FINISHED WITH PPL
COPYCMD  MVC   CMDBUF+4(Ø),Ø(R3)        * MOVE COMMAND INTO BUFFER
         EJECT
*-------------------------------------
* PARSE THE LOGON COMMAND
*-------------------------------------
CALLPARS EQU   *
         OI    ATTNFLAG,X'4Ø'           * SET 'PARS' FLAG FOR ATTNEXIT
*
         CALLTSSR EP=IKJPARS,MF=(E,PPLIST)
*
         TM    ATTNFLAG,X'8Ø'           * DID USER HIT ATTENTION ?
         BO    ALLDONE                  * YES, SO QUIT
*
         NI    ATTNFLAG,X'FF'-X'4Ø'     * CLEAR 'PARS' FLAG
         LTR   R15,R15                  * PARSE OK?
         BNZ   PARSERR                  * NO
*
* IF NO USERID PROMPT USER FOR ONE.  FOR SOME REASON THE IKJPARS
* PROMPT MECHANISM IS NOT WORKING HERE.
*
         L     R1Ø,APDL                 * PDL ADDRESSS ...
         USING LOGONPDL,R1Ø             * ... AND ADDRESSABILITY
*
         TM    UIDPDE+6,X'8Ø'           * USERID PRESENT ?
         BO    GOTUID                   * YES, SO CARRY ON
*
         MVC   WTOBUF(LLGN1Ø3W),LGN1Ø3W * ISSUE AUTHORISATION ...
         TPUT  WTOBUF,LLGN1Ø3W          * ... WARNING MESSAGE
*
         MVC   WTOBUF(LLGN1Ø3A),LGN1Ø3A * MOVE PROMPT MSG BELOW LINE
         OI    ATTNFLAG,X'2Ø'           * SET 'TGET' FLAG FOR ATTNEXIT
PLOOP    EQU   *
         TPUT  WTOBUF,LLGN1Ø3A,ASIS     * ASK USER FOR UID
*
         LTR   R15,R15                  * TPUT OK ?
         BZ    GETUID                   * YES, SO GO GET REPLY
*
```

```
         CH    R15,H8                    * ATTENTION ?
         BE    DISCONCT                  * YES, SO DISCONNECT
         CH    R15,H2Ø                   * TERMINAL DISCONNECTED ?
         BE    DISCONCT                  * YES, SO DISCONNECT
         B     ALLDONE                   * LET IBM HANDLE OTHER ERRORS
*
GETUID   EQU   *
         TGET  TGETBUF,7                 * GET THE REPLY
*
         LTR   R15,R15                   * TGET OK ?
         BZ    TESTATTN                  * YES, SO CONTINUE PROCESSING
*
         CH    R15,H8                    * ATTENTION ?
         BE    DISCONCT                  * YES, SO DISCONNECT
         CH    R15,H2Ø                   * TERMINAL DISCONNECTED ?
         BE    DISCONCT                  * YES, SO DISCONNECT
*
TESTATTN EQU   *
         TM    ATTNFLAG,X'8Ø'            * DID USER HIT ATTENTION ?
         BO    ALLDONE                   * YES, SO QUIT
*
         LTR   R1,R1                     * IF NULL, ...
         BNP   PLOOP                     * ... KEEP ASKING
*
         NI    ATTNFLAG,X'FF'-X'2Ø'      * CLEAR 'TGET' FLAG
         LA    R2,TGETBUF                * STORE @(USERID) ...
         ST    R2,UIDPDE                 * ... IN USERID PDE
         STH   R1,UIDPDE+4               * STORE L'USERID IN PDE
         OI    UIDPDE+6,X'8Ø'            * SET 'USERID PRESENT' FLAG
         BCTR  R1,Ø                      * ENSURE USERID ...
         EX    R1,EXUC                   * ... IS UPPER CASE
         B     GOTUID                    * JUMP OVER EXECUTED MVC
*
EXUC     OC    TGETBUF(Ø),UCMASK         * CONVERT USERID TO UPPER CASE
*
GOTUID   EQU   *
         CLI   DEBUGPDE+1,X'Ø1'          * WAS DEBUG ENTERED ?
         BNE   CHEKLINE                  * NO
*
* IF DEBUG SPECIFIED, WTO INPUT COMMAND TO CONSOLE
*
         MVC   WTOBUF(LLGN11ØD),LGN11ØD  * MOVE DEBUG WTO INTO WORKAREA
*
         L     R15,ACMDBUF               * @(INPUT BUFFER PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R9,PARMLEN                * R8 = L'(INPUT BUFFER)
         L     R8,PARMADR                * R9 = @(INPUT BUFFER)
         DROP  R15                       * FINISHED WITH PARM ENTRY
*
         CH    R9,H8Ø                    * L'(INPUT BUFFER) > 8Ø ?
         BNH   *+8                       * NO - TAKE THE LOT
         LH    R9,H8Ø                    * YES - TRUNCATE AT 8Ø CHARS
```

```
         BCTR  R9,Ø                      * DECREMENT FOR EXECUTE
         EX    R9,DBGM1                  * MOVE INPUT BUFFER INTO WTO
*
         WTO   MF=(E,WTOBUF)             * ISSUE DEBUG MESSAGE
*
         B     ASVTSCAN                  * JUMP OVER EXECUTED MVC
*
DBGM1    MVC   WTOBUF+12(Ø),Ø(R8)        * MOVE DATA INTO DEBUG WTO
         EJECT
*********************************************************************
* UNLESS THE USER SPECIFIED 'RECONNECT'   (IN WHICH CASE WE EXPECT TO
* FIND HIM ALREADY LOGGED ON AND DON'T CARE IF HE IS), SCAN FOR OTHER
* ACTIVE SESSIONS FOR THIS USER. IF ANY EXIST, TELL BOTH THEM AND THE
* LOGGING ON SESSION.   THIS  CAN HAPPEN IF A USER IS LOGGED ON ELSE-
* WHERE AND TRIES LOGGING ON AGAIN. THE RESTRICTION ON DUPLICATE LOG-
* ONS DOES NOT COME INTO FORCE UNTIL AFTER THIS INITIAL LOG-ON PROCESS
* HAS COMPLETED.
*********************************************************************
ASVTSCAN EQU   *
         CLI   RECONPDE+1,X'Ø1'          * WAS 'RECONNECT' REQUESTED ?
         BE    BUILDLCB                  * YES - SKIP THIS BIT
*
         L     R2,UIDPDE                 * R2 = @(USERID)
         LH    R3,UIDPDE+4               * R3 = L'USERID
         BCTR  R3,Ø                      * DECREMENT FOR EXECUTE
         MVC   USERNAME,BLANKS           * CLEAR USERNAME TO BLANKS
         EX    R3,MOVEUID                * MOVE UID INTO USERNAME

         L     R7,PSAAOLD-PSA(RØ)        * PSA -> ASCB ...
         L     R7,ASCBTSB-ASCB(R7)       * ... ASCB -> TSB
*
         LA    R1,L'LUNAME                  * R1 = 'TRUE' LENGTH
         IVSK  R2,R7                        * R2 =  TSB STORAGE KEY
         MVCK  LUNAME(R1),TSBTRMID-TSB(R7),R2 * COPY TERMINAL ID
*
         MVC   WTOBUF(LLGN1Ø4I),LGN1Ø4I  * MOVE LGN1Ø4I MSG BELOW LINE
         MVC   WTOBUF+8(6),USERNAME      * MOVE IN USERID
         MVC   WTOBUF+54(8),LUNAME       * MOVE IN TERMINAL NAME
*
         L     R9,CVTPTR                 * CVT ADDRESS ...
         USING CVT,R9                    * ... AND ADDRESSABILITY
*
         L     R9,CVTASVT                * ASVT ADDRESS ...
         USING ASVT,R9                   * ... AND ADDRESSABILITY
*
         L     R8,ASVTMAXU               * # ASVT ENTRIES
         LA    R9,ASVTENTY               * @(FIRST ASVT ENTRY)
*
         DROP  R9                        * FINISHED WITH CVT/ASVT
*
* SCAN ACTIVE ASCBS FOR OTHER TSO SESSIONS OF LOGGING-ON USER
*
```

```
         SR    R6,R6                     * ZERO @(ASID)
ASCBSCAN EQU   *
         TM    Ø(R9),ASVTAVAL            * IS THIS ENTRY IN USE ?
         BO    NEXTASCB                  * NO, SO SKIP IT
*
         L     R7,Ø(R9)                  * ASCB ADDRESS ...
         USING ASCB,R7                   * ... AND ADDRESSABILITY
*
         ICM   R1,B'1111',ASCBTSB        * IS THIS A TSO SESSION ?
         BZ    NEXTASCB                  * IF NOT, TRY NEXT ASCB
*
         L     R1,ASCBJBNS               * @(TSU SESSION NAME)
         CLC   USERNAME,Ø(R1)            * SAME AS CURRENT LOGON UID ?
         BNE   NEXTASCB                  * IF NOT, TRY NEXT ASCB
*
         LA    R6,ASCBASID                  * R6 = @(ASID)
         TPUT  WTOBUF,LLGN1Ø4I,ASIDLOC=(R6) * TELL OTHER USER
*
NEXTASCB EQU   *
         LA    R9,4(R9)                  * MOVE TO NEXT ASVT ENTRY
         BCT   R8,ASCBSCAN               * SCAN UNTIL ALL DONE
*
         DROP  R7                        * FINISHED WITH ASCB
*
* IF THIS USER ALREADY LOGGED ON, TELL CURRENT SESSION
*
         LTR   R6,R6                     * USER ALREADY LOGGED ON ?
         BZ    BUILDLCB                  * NO
*
         TPUT  WTOBUF,LLGN1Ø4I           * TELL THIS USER OF OTHERS
*
         STIMER WAIT,BINTVL=F2ØØ         * LET USER READ MESSAGE
*
         B     BUILDLCB                  * GO AND BUILD LCB
*
MOVEUID  MVC   USERNAME(Ø),Ø(R2)         * MOVE UID INTO USERNAME
         EJECT
*********************************************************************
* BUILD THE LOGON COMMUNICATION BLOCK.   NOTE THAT WE DO THIS EVEN IF
* THE USER SPECIFIED 'RECONNECT'   -  AT THIS STAGE WE CANNOT TELL IF
* RECONNECTION WILL SUCCEED. IF IT DOES NOT, NORMAL LOG-ON PROCESSING
* (INCLUDING IKJEFLD3) WILL OCCUR.
*********************************************************************
* CREATE LCB AND STORE ADDRESS IN EXIT COMMUNICATION WORD
*
BUILDLCB EQU   *
         GETMAIN R,LV=LCBLENX,SP=13Ø     * GET LCB
*
         MVI   Ø(R1),X'ØØ'               * CLEAR LCB ...
         MVC   1(LCBLENX-1,R1),Ø(R1)     * ... TO ZEROS
*
         L     R15,AEXCOMM               * @(EXIT COMM. WORD ENTRY)
```

```
         USING PARMENT,R15              * PARM ENTRY ADDRESSABILITY
         MVI   PARMKEY+3,X'Ø1'          * UPDATE KEY
         LA    R2,4                     * UPDATE ...
         ST    R2,PARMLEN               * ... LENGTH
         ST    R1,PARMADR               * UPDATE ADDRESS
         DROP  R15                      * FINISHED WITH PARM ENTRY
*
         LR    R9,R1                    * LCB ADDRESS ...
         USING LCB,R9                   * ... AND ADDRESSABILITY
*
         MVC   LCBID,LCBNAME            * 'LCB '
         ICM   RØ,B'1ØØØ',SP13Ø         * INSERT SUBPOOL INTO LENGTH
         ST    RØ,LCBSP                 * STORE IN LCB
*
* SET DEBUG FLAG IN LCBFLAG1 IF USER SPECIFIED 'DEBUG'
*
         CLI   DEBUGPDE+1,X'Ø1'         * WAS DEBUG ENTERED ?
         BNE   BLCBØØØØ                 * NO
         OI    LCBFLAG1,LCBF1DBG        * YES, SET DEBUG FLAG
*-------------------------------------
* INSERT CPU TIME LIMIT INTO LCB. A MAXIMUM UPPER LIMIT OF 3Ø MINUTES
* IS  ENFORCED BY THE FOLLOWING CODE.   IF NO TIME WAS SPECIFIED,   A
* DEFAULT OF 1Ø MINUTES IS SET.
*-------------------------------------
BLCBØØØØ EQU   *
         MVC   LCBTIME(L'LCBTIME),BLANKS * INITIALIzE FIELD TO BLANKS
         TM    TIMESPDE+6,X'8Ø'         * TIME SUPPLIED?
         BNO   BLCBØØØ5                 * NO, USE THE DEFAULT VALUE
*
         L     R1,TIMESPDE              * @(TIME)
         LH    R2,TIMESPDE+4            * L'TIME
*
         CH    R2,H4                    * LEN = 4?
         BNE   BLCBØØØ1                 * NO
         CLC   MAXTIME(4),Ø(R1)         * SPECIFIED TIME TOO BIG?
         BL    BLCBØØØ4                 * YES, SO USE THE DEFAULT MAX
         MVC   LCBTIME(4),Ø(R1)         * MOVE TIME INTO LCBTIME
         B     BUILDCMD                 * LCB COMPLETE
*
BLCBØØØ1 EQU   *
         CH    R2,H3                    * LEN = 3?
         BNE   BLCBØØØ2                 * NO
         CLC   MAXTIME+1(3),Ø(R1)       * SPECIFIED TIME TOO BIG?
         BL    BLCBØØØ4                 * YES, SO USE THE DEFAULT MAX
         MVC   LCBTIME+1(3),Ø(R1)       * MOVE TIME INTO LCBTIME
         B     BUILDCMD                 * LCB COMPLETE
*
BLCBØØØ2 EQU   *
         CH    R2,H2                    * LEN = 2?
         BNE   BLCBØØØ3                 * NO, MUST BE 1 SO MUST BE OK
         CLC   MAXTIME+2(2),Ø(R1)       * SPECIFIED TIME TOO BIG?
         BL    BLCBØØØ4                 * YES, SO USE THE DEFAULT MAX
```

```
         MVC   LCBTIME+2(2),Ø(R1)        * MOVE TIME INTO LCBTIME
         B     BUILDCMD                  * LCB COMPLETE
*
BLCBØØØ3 EQU   *
         MVC   LCBTIME+3(1),Ø(R1)        * MOVE TIME INTO LCBTIME
         B     BUILDCMD                  * LCB COMPLETE
*
BLCBØØØ4 EQU   *
         MVC   LCBTIME,MAXTIME           * INSERT DEFAULT MAX TIME
         B     BUILDCMD                  * LCB COMPLETE
*
BLCBØØØ5 EQU   *
         MVC   LCBTIME,DEFTIME           * INSERT DEFLT TIME (1Ø MINS)
*
         DROP  R9                        * FINISHED WITH LCB
         EJECT
*********************************************************************
* REBUILD THE LOG-ON COMMAND WITHOUT THE LOCAL PARAMETERS AND WITH THE
* FOLLOWING CHANGES :
* 1) IF THE USER SPECIFIED A PASSWORD  ('LOGON USERID/PASSWORD')  THE
*    PASSWORD IS DELETED, AS IT WILL HAVE TO BE ENTERED ON THE LOG-ON
*    PANEL ANYWAY.
* 2) THE MAIL  AND  NOTICES CONTROL SWITCHES ARE SET ON,   UNLESS THE
*    USER SPECIFIED 'NOMAIL' AND/OR 'NONOTICES'.  HAVING DONE THIS IT
*    IS NOT NECESSARY TO ADD 'MAIL' AND 'NOTICES' TO THE NEW COMMAND.
* 3) UNLESS THE USER SPECIFIED 'RECONNECT', 'NORECONNECT' IS APPENDED
*    TO THE REBUILT COMMAND - IT IS NOT ENOUGH TO LEAVE THE RECONNECT
*    CONTROL SWITCH OFF TO CLEAR A PRIOR RECONNECT REQUEST. MEMORY OF
*    SUCH A REQUEST CAN ONLY BE CLEARED IN THIS WAY.
*********************************************************************
BUILDCMD EQU   *
         L     R15,ACMDBUF               * @(INPUT BUFFER PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R8,PARMLEN                * R8 = L'(INPUT BUFFER)
         L     R9,PARMADR                * R9 = @(INPUT BUFFER)
         DROP  R15                       * FINISHED WITH PARM ENTRY
         MVI   Ø(R9),C' '                * CLEAR COMMAND BUFFER ...
         SH    R8,H2                     * ... TO ...
         EX    R8,CBUFCLR                *     ... BLANKS
         MVC   Ø(L'LOGON,R9),LOGON       * COMMAND IS 'LOGON'
         LA    R9,L'LOGON+1(R9)          * INCREMENT BUFFER POINTER
         L     R1,UIDPDE                 * @(USERID)
         LH    R2,UIDPDE+4               * L'USERID
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE USERID INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,2(R9)                  * ... BUFFER POINTER
*
         TM    ACCTSPDE+6,X'8Ø'          * ACCTNUM SUPPLIED ?
         BNO   BCMDØØØ1                  * NOT THIS TIME
         MVC   Ø(L'ACCT,R9),ACCT         * MOVE IN 'ACCT('
         LA    R9,L'ACCT(R9)             * INCREMENT BUFFER POINTER
```

```
         L     R1,ACCTSPDE               * @(ACCTNUM)
         LH    R2,ACCTSPDE+4             * L'ACCTNUM)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE ACCTNUM INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ1 EQU   *
         TM    PROCSPDE+6,X'8Ø'          * PROC SUPPLIED ?
         BNO   BCMDØØØ2                  * NOT THIS TIME
         MVC   Ø(L'PROC,R9),PROC         * MOVE IN 'PROC('
         LA    R9,L'PROC(R9)             * INCREMENT BUFFER POINTER
         L     R1,PROCSPDE               * @(PROCNAME)
         LH    R2,PROCSPDE+4             * L'PROCNAME)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE PROCNAME INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ2 EQU   *
         TM    SIZESPDE+6,X'8Ø'          * REGION SUPPLIED ?
         BNO   BCMDØØØ3                  * NOT THIS TIME
         MVC   Ø(L'SIZE,R9),SIZE         * MOVE IN 'SIZE('
         LA    R9,L'SIZE(R9)             * INCREMENT BUFFER POINTER
         L     R1,SIZESPDE               * @(REGION SIZE)
         LH    R2,SIZESPDE+4             * L'(REGION SIZE)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE REGION SIZE INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ3 EQU   *
         TM    PGNSPDE+6,X'8Ø'           * PERFORMANCE GROUP SUPPLIED ?
         BNO   BCMDØØØ4                  * NOT THIS TIME
         MVC   Ø(L'PERFORM,R9),PERFORM   * MOVE IN 'PERFORM('
         LA    R9,L'PERFORM(R9)          * INCREMENT BUFFER POINTER
         L     R1,PGNSPDE                * @(PGN)
         LH    R2,PGNSPDE+4              * L'(PGN)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE REGION SIZE INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ4 EQU   *
```

```
         TM    GRPSPDE+6,X'8Ø'           * RACF GROUP SUPPLIED ?
         BNO   BCMDØØØ5                  * NOT THIS TIME
         MVC   Ø(L'GROUP,R9),GROUP       * MOVE IN 'GROUP('
         LA    R9,L'GROUP(R9)            * INCREMENT BUFFER POINTER
         L     R1,GRPSPDE                * @(RACF GROUP)
         LH    R2,GRPSPDE+4              * L'(RACF GROUP)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE REGION SIZE INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ5 EQU   *
         TM    SECLSPDE+6,X'8Ø'          * SECLABEL SUPPLIED ?
         BNO   BCMDØØØ6                  * NOT THIS TIME
         MVC   Ø(L'SECLABEL,R9),SECLABEL * MOVE IN 'SECLABEL('
         LA    R9,L'SECLABEL(R9)         * INCREMENT BUFFER POINTER
         L     R1,GRPSPDE                * @(SECLABEL)
         LH    R2,GRPSPDE+4              * L'(SECLABEL)
         BCTR  R2,Ø                      * DECREMENT FOR EXECUTE
         EX    R2,CBUFLOAD               * MOVE REGION SIZE INTO BUFFER
         AR    R9,R2                     * INCREMENT ...
         LA    R9,1(R9)                  * ... BUFFER POINTER
         MVI   Ø(R9),C')'                * INSERT ')'
         LA    R9,2(R9)                  * INCREMENT BUFFER POINTER
*
BCMDØØØ6 EQU   *
         CLI   OIDCPDE+1,X'Ø1'           * 'OIDCARD' ENTERED ?
         BNE   BCMDØØØ7                  * NOT THIS TIME
         MVC   Ø(L'OIDCARD,R9),OIDCARD   * MOVE IN 'OIDCARD'
         LA    R9,L'OIDCARD+1(R9)        * INCREMENT BUFFER POINTER
*
BCMDØØØ7 EQU   *
         CLI   MAILPDE+1,X'Ø1'           * 'MAIL' ENTERED/DEFAULTED ?
         BNE   BCMDØØØ8                  * NOT THIS TIME
         L     R15,ACTLSW                * @(CONTROL SWITCH PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR                * R1 = @(CONTROL SWITCHES)
         OI    2(R1),X'8Ø'               * SET MAIL CONTROL SWITCH
         DROP  R15                       * FINISHED WITH PARM ENTRY
*
BCMDØØØ8 EQU   *
         CLI   NOTICPDE+1,X'Ø1'          * 'NOTICES' ENTERED/DEFAULTED?
         BNE   BCMDØØØ9                  * NOT THIS TIME
         L     R15,ACTLSW                * @(CONTROL SWITCH PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR                * R1 = @(CONTROL SWITCHES)
         OI    2(R1),X'4Ø'               * SET NOTICES CONTROL SWITCH
         DROP  R15                       * FINISHED WITH PARM ENTRY
*
BCMDØØØ9 EQU   *
```

```
         CLI   RECONPDE+1,X'Ø1'          * 'RECONNECT' ENTERED ?
         BNE   BCMDØØ1Ø                  * NOT THIS TIME
         L     R15,ACTLSW                * @(CONTROL SWITCH PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R1,PARMADR                * R1 = @(CONTROL SWITCHES)
         OI    1(R1),X'Ø1'               * SET RECONNECT CONTROL SWITCH
         DROP  R15                       * FINISHED WITH PARM ENTRY
         B     BCMDØØ11                  * ALL DONE
*
BCMDØØ1Ø EQU   *
         MVC   Ø(L'NORECON,R9),NORECON   * MOVE IN 'NORECONNECT'
         LA    R9,L'NORECON+1(R9)        * INCREMENT BUFFER POINTER
*
* LOGON COMMAND REBUILT - UPDATE COMMAND LENGTH
*
BCMDØØ11 EQU   *
         L     R15,ACMDBUF               * @(INPUT BUFFER PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R8,PARMADR                * @(INPUT BUFFER)
         SR    R9,R8                     * L'(REBUILT COMMAND)
         BCTR  R9,Ø                      * REMOVE TRAILING BLANK
         ST    R9,PARMLEN                * UPDATE L'(INPUT BUFFER)
         DROP  R15                       * FINISHED WITH PARM ENTRY
         CLI   DEBUGPDE+1,X'Ø1'          * WAS DEBUG ENTERED ?
         BNE   ALLDONE                   * NO
*
* IF DEBUG REQUESTED, WTO REBUILT COMMAND TO CONSOLE
*
         MVC   WTOBUF(LLGN11ØD),LGN11ØD  * MOVE DEBUG WTO INTO WORKAREA
         MVI   WTOBUF+9,C'1'             * CHANGE MSGID TO LGN111D
         CH    R9,H8Ø                    * L'(INPUT BUFFER) > 8Ø ?
         BNH   *+8                       * NO - TAKE THE LOT
         LH    R9,H8Ø                    * YES - TRUNCATE AT 8Ø CHARS
         BCTR  R9,Ø                      * DECREMENT FOR EXECUTE
         EX    R9,DBGM1                  * MOVE INPUT BUFFER INTO WTO
         WTO   MF=(E,WTOBUF)             * ISSUE DEBUG MESSAGE
         B     ALLDONE                   * ALL DONE
CBUFCLR  MVC   1(Ø,R9),Ø(R9)             * CLEAR COMMAND BUFFER
CBUFLOAD MVC   Ø(Ø,R9),Ø(R1)             * MOVE PARM INTO BUFFER
*
         DROP  R1Ø                       * FINISHED WITH PDL
         EJECT
*********************************************************************
* ALL DONE.
*********************************************************************
ALLDONE  EQU   *
         TM    ATTNFLAG,X'8Ø'            * USER HIT ATTN KEY ?
         BNO   FREEPDL                   * NO; PROCEED WITH LOGON
*
* ATTENTION HIT/TERMINAL GONE - REQUEST TERMINATION (DISCONNECTION)
*
DISCONCT EQU   *
```

```
         L     R15,ACTLSW                * @(CONTROL SWITCH PARM ENTRY)
         USING PARMENT,R15               * PARM ENTRY ADDRESSABILITY
         L     R4,PARMADR                * R1 = @(CONTROL SWITCHES)
         OI    Ø(R4),X'1Ø'               * SET 'DISCONNECT' SWITCH
         DROP  R15                       * FINISHED WITH PARM ENTRY
         DROP  R11                       * FINISHED WITH PARMLIST
*
* FREE THE PDL AND DELETE THE ATTENTION EXIT
*
FREEPDL  EQU   *
         IKJRLSA APDL                    * DELETE PDL
         MVC   STAXLIST(STAXLEN),STAXDUM * DELETE ...
         STAX  MF=(E,STAXLIST)           * ... ATTENTION EXIT
*
* FREE THE WORKAREA AND RETURN
*
RETURN   EQU   *
         LR    R1,R13                    * GET SAVE AREA ADDR
         L     R13,4(R13)                * RECALL SAVE AREA ADDRESS
         FREEMAIN RC,LV=WKALEN,A=(R1)    * FREE WORKSPACE
         L     R14,12(R13)               * RESTORE R14
         SR    R15,R15                   * RC = Ø
         LM    RØ,R12,2Ø(R13)            * RESTORE RØ-R12
         BR    R14                       * AND RETURN
         EJECT
*********************************************************************
* ATTENTION EXIT ROUTINE, ENTERED IF THE USER HITS THE ATTENTION KEY.
*********************************************************************
         DS    ØF
ATTNEXIT EQU   *
         L     R3,8(R1)                  * @(USER INFO)
         L     R12,Ø(R3)                 * RESTORE BASE REGISTER
         L     R2,4(R3)                  * @(ATTNFLAG)
         OI    Ø(R2),X'8Ø'               * SET ATTENTION FLAG
         TM    Ø(R2),X'4Ø'               * ATTN DURING IKJPARS ?
         BNO   ATTNØØØ1                  * NO
         MVC   WTOBUF(LLGN1Ø6A),LGN1Ø6A  * ASK USER TO ...
         TPUT  WTOBUF,LLGN1Ø6A           * ... HIT ENTER KEY
         B     ATTNRETN                  *     ... AND RETURN
*
ATTNØØØ1 EQU   *
         TM    Ø(R2),X'2Ø'               * ATTN DURING TGET ?
         BO    ATTNRETN                  * YES - NO PROMPT NEEDED
*
         MVC   WTOBUF(LLGN1Ø7A),LGN1Ø7A  * ASK USER TO ...
         TPUT  WTOBUF,LLGN1Ø7A           * ... HIT ENTER KEY
*
ATTNRETN EQU   *
         BR    R14                       * RETURN TO ATTN HANDLER
         EJECT
*********************************************************************
* ERROR MESSAGES - SENT TO SYSTEM LOG AND/OR TERMINAL USER
```

```
*********************************************************************
STAXERR  EQU   *
         SRA   R15,1                     * DIVIDE STAX RC BY 2
         LA    R1,RETCODES(R15)          * ADDRESS OF EBCDIC RET CODE
         MVC   WTOBUF(LLGN101W),LGN101W  * MOVE WTO INTO WORKAREA
         MVC   WTOBUF+27(2),0(R1)        * MOVE RETCODE INTO WTO
         WTO   MF=(E,WTOBUF)             * TELL SYSLOG OF ERROR
         LH    R0,LGN101W                * LENGTH OF ...
         SH    R0,H4                     * ... ACTUAL MESSAGE TEXT
         TPUT  WTOBUF+4,(R0)             * INFORM TSO USER ...
         B     BUILDPPL                  * ... AND RESUME PROCESSING
*
PARSERR  EQU   *
         SRA   R15,1                     * DIVIDE IKJPARS RC BY 2
         LA    R1,RETCODES(R15)          * ADDRESS OF EBCDIC RET CODE
         MVC   WTOBUF(LLGN102E),LGN102E  * MOVE WTO INTO WORKAREA
         MVC   WTOBUF+54(2),0(R1)        * MOVE RETCODE INTO WTO
         WTO   MF=(E,WTOBUF)             * TELL SYSLOG OF ERROR
         LH    R0,LGN102E                * LENGTH OF ...
         SH    R0,H4                     * ... ACTUAL MESSAGE TEXT
         TPUT  WTOBUF+4,(R0)             * INFORM TSO USER ...
         B     ALLDONE                   * ... AND RESUME PROCESSING
*
LGN101W  WTO   'LGN101W STAX failed rc=??; continuing',               +
               ROUTCDE=2,DESC=3,MF=L
LLGN101W EQU   *-LGN101W
*
LGN102E  WTO   'LGN102E Unable to parse LOGON command; IKJPARS rc=??', +
               ROUTCDE=2,DESC=3,MF=L
LLGN102E EQU   *-LGN102E
*
LGN103W  DC    C'LGN103W Unauthorized use of this system may lead to pr+
               osecution'
LLGN103W EQU   *-LGN103W
*
LGN103A  DC    C'LGN103A Enter your USERID if you are an authorized use+
               r :'
LLGN103A EQU   *-LGN103A
*
LGN104I  DC    C'LGN104I USERID is establishing a duplicate session on +
               TERMINAL'
LLGN104I EQU   *-LGN104I
*
LGN105I  DC    C'LGN105I ATTENTION acknowledged - LOGON will terminate'
LLGN105I EQU   *-LGN105I
*
LGN106A  DC    C'LGN106A Hit ENTER and satisfy any following prompt to +
               complete termination'
LLGN106A EQU   *-LGN106A
*
LGN107A  DC    C'LGN107A Hit ENTER to complete termination'
LLGN107A EQU   *-LGN107A
```

```
*
         DS    ØF
LGN11ØD  WTO   'LGN11ØD                                                +
                                                ',ROUTCDE=2,DESC=3,MF=L
LLGN11ØD EQU   *-LGN11ØD
         EJECT
*********************************************************************
* CONSTANTS AND DATA AREAS
*********************************************************************
APCL     DC    V(IKJEFLE2)
F2ØØ     DC    F'2ØØ'
H2       DC    H'2'
H3       DC    H'3'
H4       DC    H'4'
H8       DC    H'8'
H2Ø      DC    H'2Ø'
H8Ø      DC    H'8Ø'
LCBNAME  DC    CL4'LCB '
SP13Ø    DC    XL1'82'
DEFTIME  DC    CL4'ØØ1Ø'
MAXTIME  DC    CL4'ØØ3Ø'
BLANKS   DC    CL8' '
UCMASK   EQU   BLANKS,7
RETCODES DC    C'ØØØ4Ø8ØC1Ø14181C2Ø'
*
LOGON    DC    CL5'LOGON'
ACCT     DC    CL5'ACCT('
PROC     DC    CL5'PROC('
SIZE     DC    CL5'SIZE('
PERFORM  DC    CL8'PERFORM('
GROUP    DC    CL6'GROUP('
SECLABEL DC    CL9'SECLABEL('
OIDCARD  DC    CL7'OIDCARD'
NORECON  DC    CL11'NORECONNECT'
*
         DS    ØF
STAXDUM  STAX   MF=L
STAXLEN  EQU   *-STAXDUM
*
         LTORG
         EJECT
*-------------------------------------
* PARAMETER CONTROL LIST FOR PARSE OF INPUT LOGON COMMAND
*-------------------------------------
*        PRINT NOGEN
IKJEFLE2 IKJPARM  DSECT=LOGONPDL
UIDPDE   IKJPOSIT USERID
ACCTPDE  IKJKEYWD
         IKJNAME  'ACCT',SUBFLD=ACCTSUB
PROCPDE  IKJKEYWD
         IKJNAME  'PROC',SUBFLD=PROCSUB
SIZEPDE  IKJKEYWD
```

```
         IKJNAME  'SIZE',SUBFLD=SIZESUB
PGNPDE   IKJKEYWD
         IKJNAME  'PERFORM',SUBFLD=PGNSUB
GROUPPDE IKJKEYWD
         IKJNAME  'GROUP',SUBFLD=GROUPSUB
SECLPDE  IKJKEYWD
         IKJNAME  'SECLABEL',SUBFLD=SECLSUB
OIDCPDE  IKJKEYWD
         IKJNAME  'OIDCARD'
MAILPDE  IKJKEYWD DEFAULT='MAIL'
         IKJNAME  'MAIL'
         IKJNAME  'NOMAIL'
NOTICPDE IKJKEYWD DEFAULT='NOTICES'
         IKJNAME  'NOTICES'
         IKJNAME  'NONOTICES'
RECONPDE IKJKEYWD DEFAULT='NORECONNECT'
         IKJNAME  'RECONNECT'
         IKJNAME  'NORECONNECT'
*
* LOCAL ADDITIONS TO THE LOGON COMMAND
*
TIMEPDE  IKJKEYWD
         IKJNAME  'TIME',SUBFLD=TIMESUB
*
DEBUGPDE IKJKEYWD DEFAULT='NODEBUG'
         IKJNAME  'DEBUG'
         IKJNAME  'NODEBUG'
*
* SUBFIELD DESCRIPTIONS
*
ACCTSUB  IKJSUBF
ACCTSPDE IKJIDENT 'ACCOUNT NUMBER',FIRST=ALPHA,OTHER=ALPHANUM,         +
               MAXLNTH=8,                                             +
               PROMPT='ACCOUNT NUMBER',                               +
               HELP='Account number as deptnnnn, eg DEPT1234'
*
PROCSUB  IKJSUBF
PROCSPDE IKJIDENT 'PROCEDURE NAME',FIRST=ALPHA,OTHER=ALPHANUM,         +
               MAXLNTH=8,                                             +
               PROMPT='PROCEDURE NAME',                               +
               HELP='Procedure name as 1-8 characters, eg FORPROC'
*
SIZESUB  IKJSUBF
SIZESPDE IKJIDENT 'REGION SIZE',FIRST=NUMERIC,OTHER=NUMERIC,MAXLNTH=5, +
               PROMPT='REGION SIZE',                                  +
               HELP='Region size in Kbytes, eg SIZE(4Ø96) = 4Mb'
*
PGNSUB   IKJSUBF
PGNSPDE  IKJIDENT 'PERFORMANCE GROUP',FIRST=NUMERIC,OTHER=NUMERIC,     +
               MAXLNTH=3,                                             +
               PROMPT='PERFORMANCE GROUP',                            +
               HELP='Performance group number between 1 and 255'
```

```
*
GROUPSUB IKJSUBF
GRPSPDE  IKJIDENT 'RACF GROUP ID',FIRST=ALPHA,OTHER=ALPHANUM,          +
               MAXLNTH=8,                                              +
               PROMPT='RACF GROUP NAME',                               +
               HELP='RACF group name as 6-8 characters, eg GRPDEPT1'
*
SECLSUB  IKJSUBF
SECLSPDE IKJIDENT 'SECURITY LABEL',FIRST=ALPHA,OTHER=ALPHANUM,         +
               MAXLNTH=8,                                              +
               PROMPT='RACF SECURITY LABEL',                           +
               HELP='RACF SECLABEL as 6-8 characters'
*
TIMESUB  IKJSUBF
TIMESPDE IKJIDENT 'CPU TIME IN MINS',FIRST=NUMERIC,OTHER=NUMERIC,      +
               MAXLNTH=4,                                              +
               PROMPT='CPU TIME LIMIT',                                +
               HELP='CPU time limit in minutes, eg TIME(10)'
         IKJENDP
         EJECT
*********************************************************************
* WORKSPACE DSECT
*********************************************************************
WORKAREA DSECT
SAVEAREA DS    18F                     * SAVE AREA
ATTNPARM DS    2F                      * DATA FOR ATTNEXIT
PPLIST   DS    8F                      * PARSE PARAMETER LIST
APDL     DS    A                       * @(PARSE DESCRIPTOR LIST)
CPECB    DS    F                       * PARSE ECB
ATTNFLAG DS    X                       * ATTENTION FLAG BYTE
TGETBUF  DS    CL7                     * TGET BUFFER
USERNAME DS    CL8                     * USER-ID FROM LOG-ON COMMAND
LUNAME   DS    CL8                     * VTAM TERMINAL NAME
*
* LOCAL COMMAND BUFFER
*
CMDBUF   DS    CL256                   * COMMAND BUFFER
*
         DS    0F
STAXLIST DS    CL(STAXLEN)
STAXOBUF DS    CL(LLGN105I)
*
WTOBUF   DS    CL256
*
WKALEN   EQU   *-WORKAREA
         EJECT
*-------------------------------------
* PARMLIST MAPPING DSECTS
*-------------------------------------
PARMLIST DSECT
ACMDBUF  DS    A                       * @(COMMAND BUFFER PARM)
ANCMDBUF DS    A                       * @(NEW COMMAND BUFFER PARM)
```

```
AUPT     DS    A                        * @(UPT PARM)
AECT     DS    A                        * @(ECT PARM)
APSCB    DS    A                        * @(PSCB PARM)
AEXCOMM  DS    A                        * @(EXIT-TO-EXIT COMM. PARM)
AEXITRC  DS    A                        * @(EXIT REASON CODE)
         DS    A                        * RESERVED
         DS    A                        * RESERVED
ACTLSW   DS    A                        * @(CONTROL SWITCH PARM)
AUID     DS    A                        * @(UID PARM)
APWD     DS    A                        * @(PASSWORD PARM)
AACTNUM  DS    A                        * @(ACCOUNT NUMBER PARM)
APROCNAM DS    A                        * @(PROCEDURE NAME PARM)
AREGION  DS    A                        * @(REGION PARM)
AJCL     DS    A                        * @(JCL BUFFER PARM)
ANPWD    DS    A                        * @(NEW PASSWORD PARM)
ASAB     DS    A                        * @(SYSTEM ATTRIBUTE BITS PRM)
AUAB     DS    A                        * @(USER ATTRIBUTE BITS PARM)
AUNIT    DS    A                        * @(GENERIC UNIT PARM)
ACECB    DS    A                        * @(CANCEL ECB PARM)
APGN     DS    A                        * @(PERFORMANCE GRP NMBR PARM)
ASDEST   DS    A                        * @(SYSOUT DEST PARM)
AGDEST   DS    A                        * @(GROUP DEST PARM)
AHOLDCLS DS    A                        * @(SUBMIT HOLD CLASS PARM)
ASUBCLS  DS    A                        * @(SUBMIT CLASS PARM)
AMSGCLS  DS    A                        * @(SUBMIT MSGCLASS PARM)
ASYSOCLS DS    A                        * @(SYSOUT CLASS PARM)
AFSTCMND DS    A                        * @(FIRST COMMAND PARM)
ARBA     DS    A                        * @(RBA PARM)
ASECLBL  DS    A                        * @(SECLABEL PARM)
ACONPRFL DS    A                        * @(CONSOLE PROFILE PARM)
APLID    DS    A                        * @(PRIMARY LANGUAGE ID PARM)
ASLID    DS    A                        * @(SECONDARY LANGUAGE ID
PARM)
*
PARMENT  DSECT
PARMKEY  DS    F                        * PARAMETER KEY
PARMLEN  DS    F                        * PARAMETER LENGTH
PARMADR  DS    A                        * PARAMETER ADDRESS
         EJECT
*-------------------------------------
* SYSTEM DSECTS
*-------------------------------------
         IHAPSA  LIST=NO
         CVT     DSECT=YES,LIST=NO
         IHAASVT
         IHAASCB
         IKJTSB
         IKJPPL
         PRINT GEN
         LCB
         PRINT NOGEN
         END
```

IKJEFLD3

```
         TITLE 'IKJEFLD: TSO LOGON POST-PROMPT EXIT'
*********************************************************************
*  SUBROUTINE IKJEFLD3                                              *
*  ———————————                                                      *
*  This is the TSO LOGON post-prompt exit. It is called by the LOGON  *
*  processor after the dialogue with the user has completed.    This  *
*  version provides support for the  local TIME() parameter we  have  *
*  added to the LOGON command, allowing the user to  specify the CPU  *
*  time limit for the session.                                      *
*                                                                   *
*  The TIME parameter is processed by the pre-prompt exit, IKJEFLD1,  *
*  and stored in the Log-on Communication Block (LCB), which it hangs
*
*  from the exit-to-exit communication word  in  the  exit parameter  *
*  list.  This exit retrieves the CPU time from the LCB and rebuilds  *
*  the log-on JCL to include a standard TIME parameter.  It also adds
*
*  the 'programmer name' field, (using the name from the ACEE, which  *
*  has been created by the time this exit is called),  and  converts  *
*  the account number from 'RACF' format to 'batch' format :        *
*                                                                   *
*  Input JCL :                                                      *
*                                                                   *
*     //userid  JOB 'deptnnnn',REGION=nnnnK                         *
*     //procname EXEC procname                                      *
*                                                                   *
*  Rebuilt JCL :                                                    *
*                                                                   *
*     //userid   JOB (nnnn,dept),'programmer name',                 *
*     //         Time=(mmmm,ss),REGION=nnnnK                        *
*     //procname EXEC procname                                      *
*                                                                   *
*  For  obvious reasons this exit is designed to be fail-safe  -  if  *
*  for any reason it is unable to rebuild the JCL,   it gives up and  *
*  lets the standard JCL through.   If no LCB is passed,   the  same  *
*  default value as used in IKJEFLD1 (ie 1Ø minutes) is used.        *
*                                                                   *
*  ENVIRONMENT                                                      *
*                                                                   *
*    State    : Supervisor                                         *
*    Key      : 8                                                  *
*    APF      : Unauthorised                                       *
*    AMODE    : 31                                                 *
*    RMODE    : 24                                                 *
*    Location : Linklist or PLPA                                   *
*                                                                   *
*  REGISTERS ON ENTRY :                                             *
*                                                                   *
*    R1  - @(STANDARD EXIT PARAMETER LIST)                          *
*          + Ø : @(COMMAND BUFFER PE)                               *
```

```
*          + 4 : @(NEW COMMAND BUFFER PE)                              *
*          + 8 : @(UPT PE)                                             *
*          +12 : @(ECT PE)                                             *
*          +16 : @(PSCB PE)                                            *
*          +2Ø : @(EXIT-TO-EXIT COMMUNICATION WORD PE)                 *
*          +24 : @(EXIT REASON CODE PE)                                *
*          +28 : RESERVED                                              *
*          +32 : RESERVED                                              *
*          +36 : @(CONTROL SWITCH PE)                                  *
*          +36 : @(JCL BUFFER PE)                                      *
*    R13 - @(SAVEAREA)                                                 *
*    R14 - RETURN ADDRESS                                              *
*    R15 - ENTRY ADDRESS                                               *
*                                                                      *
*  REGISTERS ON RETURN :                                               *
*                                                                      *
*    RØ-R14 - AS AT ENTRY                                              *
*    R15 - RETURN CODE                                                 *
*           Ø : CONTINUE NORMAL PROCESSING                             *
************************************************************************
         EJECT
IKJEFLD3 CSECT
IKJEFLD3 AMODE 31
IKJEFLD3 RMODE 24
RØ       EQU   Ø
R1       EQU   1                  * PARM LIST ADDRESS ON ENTRY
R2       EQU   2                  * @(EXIT COMM. PARM ENTRY)
R3       EQU   3                  * @(JCL BUFFER PARM ENTRY)
R4       EQU   4                  * WORK REGISTER
R5       EQU   5                  * WORK REGISTER
R6       EQU   6                  * WORK REGISTER
R7       EQU   7                  * L'REGION=
R8       EQU   8                  * @(REGION=)
R9       EQU   9                  * @(ACCTNUM)
R1Ø      EQU   1Ø                 * @(JCL BUFFER)
R11      EQU   11                 * @(LCB)
R12      EQU   12                 * BASE REGISTER
R13      EQU   13                 * SAVEAREA/WORKAREA ADDRESS
R14      EQU   14                 * RETURN ADDRESS
R15      EQU   15                 * ENTRY ADDRESS
         USING *,R15              * ADDRESSABILITY
         B     START              * BRANCH TO START OF CODE
         DC    AL1(LASTL-FIRSTL)  * LENGTH OF HEADER TEXT
FIRSTL   EQU   *
         DC    C'IKJEFLD3'
LASTL    EQU   *
         DC    C' '
         DC    CL8'&SYSDATE'
         DC    C' '
         DC    CL5'&SYSTIME'
         DROP  R15                * FINISHED WITH R15
```

```
         DS    ØF                  * ALIGN TO FULL WORD BOUNDARY
*********************************************************************
* ADDRESSABILITY AND LINKAGE - REENTRANT FORM
*********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)          * SAVE REGISTERS IN HSA
         LR    R12,R15                  * LOAD BASE REGISTER
         USING IKJEFLD3,R12             * AND DEFINE ADDRESSIBILITY
*
         LR    R2,R1                    * SAVE R1 ACROSS GETMAIN
*
         GETMAIN R,LV=WKALEN            * GETMAIN NEW SAVEAREA
         ST    R13,4(R1)                * STORE HSA ADDRESS
         ST    R1,8(R13)                * STORE LSA ADDRESS
         LR    R13,R1                   * R13 = OUR SAVEAREA ADDRESS
         USING WORKAREA,R13             * WORKAREA ADDRESSABILITY
*
         LR    R1,R2                    * RESTORE ORIGINAL R1
         ST    R1,APARMLST              * AND SAVE IT
         EJECT
*********************************************************************
* GET ON WITH IT
*********************************************************************
         USING PARMLIST,R1              * PARMLIST ADDRESSABILITY
         L     R2,AEXCOMM               * R2 = @(EXIT COMM. PARM)
         L     R3,AJCL                  * R3 = @(JCL PARM)
         DROP  R1                       * FINISHED WITH PARMLIST
         USING PARMENT,R2               * EXIT COMM PARM ENTRY
         SR    R11,R11                  * PRE-CLEAR LCB ADDRESS
         CLI   PARMKEY+3,X'Ø1'          * CHECK FOR CORRECT KEY
         BNE   NOLCB                    * SHOUT IF LCB MISSING
         ICM   R11,B'1111',PARMADR      * @(LCB)
         BNP   NOLCB                    * SHOUT IF LCB MISSING
         DROP  R2                       * FINISHED WITH PARM ENTRY
         USING LCB,R11                  * LCB ADDRESSABILITY
         CLC   LCBID,LCBNAME            * IS THIS THE REAL THING ?
         BNE   BADLCB                   * OH DEAR
*
SKIPLCB  EQU   *
         USING PARMENT,R3               * JCL BUFFER PARM ENTRY
         ICM   R1Ø,B'1111',PARMADR      * @(JCL BUFFER)
         BNP   NOJCL                    * THIS SHOULD NEVER HAPPEN
         DROP  R3                       * FINISHED WITH PARM ENTRY
*
         USING JCLBUF,R1Ø               * JCL BUFFER ADDRESSABILITY
*-------------------------------------
* SCAN SUPPLIED JOB CARD AND IDENTIFY THE BITS WE WANT
*-------------------------------------
* IF USER SPECIFIED 'DEBUG', DISPLAY THE OLD JCL ON THE CONSOLE
*
         TM    LCBFLAG1,LCBF1DBG        * DEBUG SPECIFIED AT LOG-ON ?
         BNO   SKIPDBG1                 * NOT THIS TIME
```

```
         MVC   WTOBUF(LLGN31ØD),LGN31ØD  * MOVE MF=L WTO INTO WORKAREA
         MVC   WTOBUF+12(8Ø),JCLINE1     * WTO FIRST LINE ...
         WTO   MF=(E,WTOBUF)             * ... OF ORIGINAL JCL
         MVC   WTOBUF+12(8Ø),JCLINE2     * WTO SECOND LINE ...
         WTO   MF=(E,WTOBUF)             * ... OF ORIGINAL JCL
*
* ANALYSE THE SUPPLIED JOB CARD, WHICH SHOULD LOOK LIKE :
*    //USERID  JOB 'DEPTnnnn',REGION=nnnnK
*
SKIPDBG1 EQU   *
         CLC   JCLINE1+1Ø(3),JOB         * CONFIRM WE HAVE A JOB CARD
         BNE   BADJOB                    * OH DEAR
*
* EXTRACT ACCOUNT NUMBER (IN RACF FORMAT, IE DEPTnnnn)
*
         LA    R9,JCLINE1+14             * START @ FOR ACCTNUM SCAN
         LA    R6,JCLINE2                * END   @ FOR ACCTNUM SCAN
LOOP1    EQU   *
         CLI   Ø(R9),C''''               * FIRST ACCTNUM DELIMITER ?
         BE    GOTACNUM                  * BINGO !
         LA    R9,1(R9)                  * SHUFFLE ALONG ONE
         CR    R9,R6                     * FALLEN OFF END ?
         BL    LOOP1                     * NOT YET, SO KEEP SCANNING
         B     NOACNUM                   * OH DEAR
*
GOTACNUM EQU   *
         LA    R9,1(R9)                  * R9 = @(ACCTNUM)
         CLI   8(R9),C''''               * ACCTNUM LENGTH CORRECT ?
         BNE   BADACNUM                  * OH DEAR
*
* EXTRACT REGION= PARAMETER
*
         LA    R8,1Ø(R9)                 * START @ FOR REGION= SCAN
LOOP2    EQU   *
         CLC   Ø(6,R8),REGION            * REGION FOUND ?
         BE    GOTRGN                    * BINGO !
         LA    R8,1(R8)                  * SHUFFLE ALONG ONE
         CR    R8,R6                     * FALLEN OFF THE END ?
         BL    LOOP2                     * NOT YET, SO KEEP SCANNING
         B     NORGN                     * OH DEAR
*
GOTRGN   EQU   *
         LA    R7,9(R8)                  * FIRST POSS END OF REGION=
LOOP3    EQU   *
         CLI   Ø(R7),C' '                * PAST END OF REGION= ?
         BE    GOTRGNE                   * BINGO !
         CLI   Ø(R7),C','                * PAST END OF REGION= ?
         BE    GOTRGNE                   * BINGO !
         LA    R7,1(R7)                  * SHUFFLE ALONG ONE
         CR    R7,R6                     * FALLEN OFF THE END ?
         BL    LOOP3                     * NO, SO KEEP SCANNING
         B     BADRGN                    * OH DEAR
*
```

```
GOTRGNE  EQU   *
         SR    R7,R8                     * R7 = L'REGION=
*--------------------------------------
* BUILD NEW JOB CARD IN WORKAREA
*--------------------------------------
* LINE 1 : //USERID   JOB (NNNN,DEPT),'PROGRAMMER NAME',
*
         MVI   LINE1,C' '                * CLEAR ...
         MVC   LINE1+1(159),LINE1        * ... LINE1/LINE2
*
         MVC   LINE1(9),JCLINE1          * '//USERID '
         MVC   LINE1+11(3),JOB           * 'JOB'
*
         MVI   LINE1+15,C'('             * ACCTNUM START DELIMITER
         MVC   LINE1+16(4),4(R9)         * 'NNNN' PART OF ACCTNUM
         MVI   LINE1+2Ø,C','             * ACCTNUM SEPARATOR
         MVC   LINE1+21(4),Ø(R9)         * 'DEPT' PART OF ACCTNUM
         MVC   LINE1+25(3),BCQ           * ACCTNUM END DELIMITER ETC
*
* GET USER ('PROGRAMMER') NAME VIA THE ACEE, WHICH SHOULD EXIST BY
* THE TIME THIS EXIT IS CALLED.
*
         USING PSA,RØ                    * DEFINE PSA ADDRESSABILITY
         L     R15,PSAAOLD               * R11 = ADDRESS OF ASCB
         DROP  RØ                        * FINISHED WITH PSA
         USING ASCB,R15                  * DEFINE ASCB ADDRESSABILITY
         L     R15,ASCBASXB              * GET ASXB ADDRESS
         DROP  R15                       * FINISHED WITH ASCB
         USING ASXB,R15                  * DEFINE ASXB ADDRESSABILITY
         ICM   R15,B'1111',ASXBSENV      * GET ASXB ADDRESS
         BZ    NOACEE                    * PREVENT ØC4
         DROP  R15                       * FINISHED WITH ASXB
         USING ACEE,R15                  * DEFINE ACEE ADDRESSABILITY
         ICM   R6,B'1111',ACEEUNAM       * GET ADDRESS OF USER NAME
         BZ    NONAME                    * IF ZERO USE DUMMY NAME
         DROP  R15                       * FINISHED WITH ACEE
         SLR   R4,R4                     * CLEAR R4
         IC    R4,Ø(R6)                  * GET LENGTH OF NAME FIELD
         BCTR  R4,Ø                      * LENGTH OF NAME ITSELF
         LTR   R4,R4                     * CHECK LENGTH NOT ZERO
         BNP   NONAME                    * IF IT IS USE DUMMY NAME
*
* COPY THE PROGRAMMER NAME INTO THE JOB CARD. IF IT CONTAINS QUOTES,
* DOUBLE THEM UP  (EG O'AARDVARK - > O''AARDVARK).   ALSO  ELIMINATE
* TRAILING BLANKS, AS THESE WILL CAUSE A JCL ERROR.
*
         LA    R5,LINE1+28               * @(START OF NAME FIELD)
LOOP4    EQU   *
         CLI   1(R6),C''''               * QUOTE ?
         BNE   MOVENCHR                  * NO, SO JUMP
         MVI   Ø(R5),C''''               * YES, SO ...
         LA    R5,1(R5)                  * ... DOUBLE IT UP
```

```
MOVENCHR EQU   *
         MVC   Ø(1,R5),1(R6)            * MOVE IN NEXT CHARACTER
         LA    R6,1(R6)                 * INCREMENT SOURCE ADDRESS
         LA    R5,1(R5)                 * INCREMENT DEST ADDRESS
         BCT   R4,LOOP4                 * AND LOOP UNTIL DONE
*
         BCTR  R5,Ø                     * @(END   OF NAME FIELD)
         LA    R6,LINE1+28              * @(START OF NAME FIELD)
LOOP5    EQU   *
         CLI   Ø(R5),C' '               * TRAILING BLANK ?
         BNE   NAMEOK                   * NO, R5=@(LAST CHAR OF NAME)
         BCTR  R5,Ø                     * YES, MOVE BACK ONE CHARACTER
         CR    R5,R6                    * AND LOOP BACK ...
         BNL   LOOP5                    * ... IF MORE TO COME
         B     NONAME                   * NAME IS ALL BLANK
*
NAMEOK   EQU   *
         LA    R5,1(R5)                 * @(NAME TERMINATOR)
         B     ENDL1                    * TERMINATE LINE1
*
NONAME   EQU   *
         MVC   LINE1+28(L'DNAME),DNAME  * DUMMY NAME
         LA    R5,LINE1+28+L'DNAME      * @(NAME TERMINATOR)
*
ENDL1    EQU   *
         MVI   Ø(R5),C''''              * NAME TERMINATOR
         MVI   1(R5),C','               * CONTINUATION COMMA
         MVC   LINE1+72(8),JCLINE1+72   * ADD 'TSUNNNNN' COMMENT
*
* LINE 2 : //         TIME=(NNNN,NN),REGION=NNNNK
*
         MVC   LINE2(2),JCLINE1         * //
         MVC   LINE2+11(14),TIME        * TIME=(ØØ1Ø,ØØ)
         MVI   LINE2+25,C','            * ,
*
* IF LCB SUPPLIED, VALIDATE TIME VALUE. IF NO LCB, OR TIME VALUE IS
* ALL ZEROS, LEAVE THE DEFAULT VALUE.
*
         LTR   R11,R11                  * HAVE WE GOT AN LCB ?
         BZ    TIMEØØØ3                 * NO, SO LEAVE DEFAULT VALUE
*
         LA    R5,LCBTIME               * ADDRESS OF TIME PARAMETER
         LA    R6,4                     * LENGTH OF TIME PARAMETER
TIMEØØØ1 EQU   *
         CLI   Ø(R5),C' '               * LEADING BLANK ?
         BNE   TIMEØØØ2                 * NO
         MVI   Ø(R5),C'Ø'               * REPLACE WITH A ZERO
         LA    R5,1(R5)                 * POINT AT NEXT CHARACTER
         BCT   R6,TIMEØØØ1              * AND LOOP BACK
         B     TIMEØØØ3                 * ALL ZERO - STAY WITH DEFAULT
*
TIMEØØØ2 EQU   *
```

```
         MVC   LINE2+17(4),LCBTIME       * MOVE TIME VALUE INTO JOBCARD
*
TIME0003 EQU   *
         BCTR  R7,0                      * EXECUTE LENGTH OF REGION=
         EX    R7,MOVERGN                * REGION=NNNNK
         B     *+10
MOVERGN  MVC   LINE2+26(0),0(R8)         * EXECUTED MVC
*
* FINALLY, REBUILD THE CONTENTS OF THE JCL BUFFER
*
         MVC   JCLINE3,JCLINE2           * MOVE // EXEC DOWN A LINE
         MVC   JCLINE1,LINE1             * FIRST LINE OF JOB CARD
         MVC   JCLINE2,LINE2             * SECOND LINE OF JOB CARD
*
         USING PARMENT,R3                * JCL BUFFER PARM ENTRY
         LA    R5,240                    * UPDATE ...
         ST    R5,PARMLEN                * ... JCL LENGTH
         DROP  R3                        * FINISHED WITH PARM ENTRY
*
* IF USER SPECIFIED 'DEBUG', DISPLAY THE NEW JCL ON THE CONSOLE
*
         TM    LCBFLAG1,LCBF1DBG         * DEBUG SPECIFIED AT LOGON ?
         BNO   RETURN                    * NOT THIS TIME
*
         MVI   WTOBUF+9,C'1'             * CHANGE MSGID TO LGN311D
*
         MVC   WTOBUF+12(80),JCLINE1     * WTO FIRST LINE ...
         WTO   MF=(E,WTOBUF)             * ... OF JCL
*
         MVC   WTOBUF+12(80),JCLINE2     * WTO SECOND LINE ...
         WTO   MF=(E,WTOBUF)             * ... OF JCL
*
         MVC   WTOBUF+12(80),JCLINE3     * WTO THIRD LINE ...
         WTO   MF=(E,WTOBUF)             * ... OF JCL
*
         DROP  R10                       * FINISHED WITH JCL
         EJECT
*********************************************************************
* ALL DONE, SO FREEMAIN LCB AND WORKAREA AND RETURN
*********************************************************************
RETURN   EQU   *
         SR    R10,R10                   * RC = 0
*
         LTR   R1,R11                    * IS THERE AN LCB TO FREE ?
         BZ    FREEWKA                   * NOT THIS TIME
*
         L     R0,LCBSP                  * LCB SUBPOOL,LENGTH
         FREEMAIN R,A=(R1),LV=(0)        * GET RID OF LCB
*
         DROP  R11                       * FINISHED WITH LCB
*
         USING PARMENT,R2                * PARMLST ENTRY ADDRESSABILITY
```

```
         XC    PARMKEY,PARMKEY           * RESET KEY
         MVC   PARMLEN,F4                * RESET LENGTH
         XC    PARMADR,PARMADR           * RESET ADDRESS
         DROP  R2                        * FINISHED WITH PARMLIST ETC
*
FREEWKA  EQU   *
         LR    R1,R13                    * R1  = OUR SAVEAREA ADDRESS
         L     R13,4(R13)                * R13 = HSA ADDRESS
         FREEMAIN R,A=(R1),LV=WKALEN     * FREE WORKAREA
*
         L     R14,12(R13)               * RESTORE R14
         LR    R15,R1Ø                   * R15 = RETURN CODE
         LM    RØ,R12,2Ø(R13)            * RESTORE RØ-R12
         BR    R14                       * AND RETURN
         EJECT
*********************************************************************
* ERROR MESSAGES - SENT TO SYSTEM LOG AND TERMINAL USER
*********************************************************************
         DS    ØF
NOLCB    EQU   *
         WTO   'LGN3Ø1W LCB not passed by IKJEFLD1; internal defaults b+
               eing used',ROUTCDE=2,DESC=3
*
         LH    RØ,NOLCB+4                * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  NOLCB+8,(RØ)              * INFORM TSO USER ...
         B     SKIPLCB                   * ... AND RESUME PROCESSING
*
         DS    ØF
BADLCB   EQU   *
         WTO   'LGN3Ø2W Supplied LCB is invalid - internal defaults bei+
               ng used',ROUTCDE=2,DESC=3
*
         LH    RØ,BADLCB+4               * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  BADLCB+8,(RØ)             * INFORM TSO USER ...
         B     SKIPLCB                   * ... AND RESUME PROCESSING
*
         DS    ØF
NOJCL    EQU   *
         WTO   'LGN3Ø3E Log-on JCL buffer is missing - processing+
               abandoned',ROUTCDE=2,DESC=3
*
         LH    RØ,NOJCL+4                * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  NOJCL+8,(RØ)              * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
BADJOB   EQU   *
         WTO   'LGN3Ø4E Unable to identify JOB card in supplied log-on+
               JCL',ROUTCDE=2,DESC=3
```

```
*
         LH    RØ,BADJOB+4               * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  BADJOB+8,(RØ)             * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
NOACNUM  EQU   *
         WTO   'LGN3Ø5E Unable to find account number on supplied JOB c+
               ard',ROUTCDE=2,DESC=3
*
         LH    RØ,NOACNUM+4              * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  NOACNUM+8,(RØ)            * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
BADACNUM EQU   *
         WTO   'LGN3Ø6E Account number on supplied JOB card is not reco+
               gnised',ROUTCDE=2,DESC=3
*
         LH    RØ,BADACNUM+4             * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  BADACNUM+8,(RØ)           * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
NORGN    EQU   *
         WTO   'LGN3Ø7E Unable to find REGION parameter on supplied JOB+
                card',ROUTCDE=2,DESC=3
*
         LH    RØ,NORGN+4                * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  NORGN+8,(RØ)              * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
BADRGN   EQU   *
         WTO   'LGN3Ø8E REGION parameter on supplied JOB card not recog+
               nized',ROUTCDE=2,DESC=3
*
         LH    RØ,BADRGN+4               * LENGTH OF ...
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  BADRGN+8,(RØ)             * INFORM TSO USER ...
         B     RETURN                    * ... AND ABANDON PROCESSING
*
         DS    ØF
NOACEE   EQU   *
         WTO   'LGN3Ø9E Unable to locate ACEE - default username will b+
               e used',ROUTCDE=2,DESC=3
*
         LH    RØ,NOACEE+4               * LENGTH OF ...
```

```
         S     RØ,F4                     * ... ACTUAL MESSAGE TEXT
         TPUT  NOACEE+8,(RØ)             * INFORM TSO USER ...
         B     NONAME                    * ... AND RESUME PROCESSING
         EJECT
*********************************************************************
* CONSTANTS, VARIABLES, AND DATA AREAS
*********************************************************************
F4       DC    F'4'
LCBNAME  DC    CL4'LCB '
JOB      DC    CL3'JOB'
REGION   DC    CL6'REGION'
TIME     DC    CL14'TIME=(ØØ1Ø,ØØ)'
BCQ      DC    CL3'),'''
DNAME    DC    C'A.A. AARDVARK'
*
         DS    ØF
LGN31ØD  WTO   'LGN31ØD ....+....1....+....2....+....3....+....4....+..+
               ..5....+....6....+....7....+....8',ROUTCDE=2,DESC=3,MF=L
LLGN31ØD EQU   *-LGN31ØD
*
* PARMLIST MAPPING DSECTS
*
PARMLIST DSECT
ACMDBUF  DS    A                       * @(COMMAND BUFFER PARM)
ANCMDBUF DS    A                       * @(NEW COMMAND BUFFER PARM)
AUPT     DS    A                       * @(UPT PARM)
AECT     DS    A                       * @(ECT PARM)
APSCB    DS    A                       * @(PSCB PARM)
AEXCOMM  DS    A                       * @(EXIT-TO-EXIT COMM. PARM)
AEXITRC  DS    A                       * @(EXIT REASON CODE)
         DS    A                       * RESERVED
         DS    A                       * RESERVED
ACTLSW   DS    A                       * @(CONTROL SWITCH PARM)
AJCL     DS    A                       * @(JCL BUFFER PARM)
*
PARMENT  DSECT
PARMKEY  DS    F                       * PARAMETER KEY
PARMLEN  DS    F                       * PARAMETER LENGTH
PARMADR  DS    A                       * PARAMETER ADDRESS
*
JCLBUF   DSECT
JCLINE1  DS    CL8Ø                    * JCL LINE 1
JCLINE2  DS    CL8Ø                    * JCL LINE 2
JCLINE3  DS    CL8Ø                    * JCL LINE 3
JCLINE4  DS    CL8Ø                    * JCL LINE 4
JCLINE5  DS    CL8Ø                    * JCL LINE 5
JCLINE6  DS    CL8Ø                    * JCL LINE 6
JCLINE7  DS    CL8Ø                    * JCL LINE 7
JCLINE8  DS    CL8Ø                    * JCL LINE 8
JCLINE9  DS    CL8Ø                    * JCL LINE 9
JCLINE1Ø DS    CL8Ø                    * JCL LINE 1Ø
*
```

```
* WORKAREA DSECT
*
WORKAREA DSECT
*
SAVEAREA DS    18F
*
APARMLST DS    F
*
LINE1    DS    CL8Ø
LINE2    DS    CL8Ø
*
         DS    ØF
WTOBUF   DS    CL(LLGN31ØD)
*
WKALEN   EQU   *-WORKAREA
*
* LOGON EXIT COMMUNICATION CONTROL BLOCK
*
         LCB
*
* SYSTEM DSECTS
*
         PUSH  PRINT
         PRINT NOGEN,NODATA
*
         IHAPSA LIST=NO
         IHAASCB LIST=NO
         IHAASXB LIST=NO
         IHAACEE
         IKJTSB
*
         POP   PRINT
         END
```

LCB MACRO

```
         MACRO
&DSNAME  LCB &DSECT=YES,&LIST=NO
         PUSH  PRINT
         DS    ØF
         AIF   ('&DSECT' EQ 'YES').DSECT
&DSNAME  EQU   *                       * LOGON COMMUNICATION BLOCK
         AGO   .DSBODY
.DSECT   ANOP
         AIF   ('&LIST' EQ 'YES').LISTON
         PRINT OFF
.LISTON  ANOP
&DSNAME  DSECT                         * LCB DSECT
.DSBODY  ANOP
LCBID    DS    CL4                     * EYECATCHER, 'LCB '
LCBSP    DS    XL1                     * LCB SUBPOOL )   FOR
```

```
LCBLEN   DS    AL3                      * LCB LENGTH  ) FREEMAIN
LCBFLAG1 DS    BL1                      * FLAG BYTE
LCBF1DBG EQU   X'80'                    *   DEBUG FLAG
         DS    XL3                      * RESERVED
LCBTIME  DS    CL4                      * CPU TIME LIMIT, 'NNNN'
LCBLENX  EQU   *-&DSNAME                * LCB LENGTH FOR GETMAIN
         POP   PRINT
         MEND
```

P R S Wright
Associate Consultant
Tessella Support Services plc (UK)

A front end to monitor program usage

THE PROBLEM

There have been many occasions where it was necessary to determine whether a software package or program was currently being used. SMF data can be used for batch job steps and TSO commands, but this can give an incomplete picture. For example, a batch job step program may invoke another program or software package as a subroutine. This would not show up in SMF data. Similarly, a TSO user might invoke a program through an ISPF service, in which case the program would not be tracked by TSO command recording.

THE SOLUTION

The only sure way to completely track program usage is by placing a hook into the program in question, or by replacing the program with a front end. The hook concept requires considerable program patching and is difficult to maintain, while a front end is relatively simple (given that the program to be monitored has no aliases). It is for this reason that I decided to write a small program, called FRONTEND, which performs some very basic monitoring.

The concept of a front end program in general is quite simple. After you determine the program you want to monitor, you rename the original program to an alternative name that does not exist in your

installation load libraries. You can then modify the front end program to transfer control to the renamed program, and then assemble and link-edit the front end program with the original program name. At this point, when anyone invokes what they think is the original program, the front end program would get control first and be able to record its invocation.

A specific example of using FRONTEND would be if you wanted to monitor usage of program IEFBR14. You would rename IEFBR14 to an alternative name such as IEFBR14$. You would then need to modify the FRONTEND program at label XCTLPGM to specify IEFBR14$ as the program to transfer control to, replace all occurrences of the string FRONTEND with IEFBR14 in the front end program, and then assemble and link-edit it as IEFBR14 (possibly into the same load library as the original IEFBR14 resides). At this point, when anyone invokes IEFBR14 (as a TSO command, batch job step, called subroutine, etc), the FRONTEND program would determine the environment it is running in and issue the appropriate TSO or MVS SEND commands to notify a list of users that the program being monitored for has been invoked. The environments that FRONTEND can distinguish are: normal background jobs, TSO foreground commands, and TSO background execution (executing the terminal monitor program or TMP, known as IKJEFT01/IKJEFT1A/IKJEFT1B, in batch). The determination of TSO foreground versus background execution is done in a fashion similar to the way that the value for the TSO built-in CLIST variable &SYSENV is determined.

The FRONTEND program needs to be link-edited into an APF authorized library if it is to monitor batch program invocation. This is required for FRONTEND to issue MVS SEND commands to the users performing the monitoring. FRONTEND does, however, check if it is not APF authorized and skips the issuance of the SEND command, so that S047 abends are avoided. The SEND commands issued in TSO foreground and background environments have no such issue, since they do not require any special authorization.

Of course, there is the issue of system maintenance to be concerned about. If you rename an IBM or OEM module, you expose yourself to the problem that maintenance, via SMP/E, AMASPZAP, or other utilities, cannot be applied successfully without first undoing all the set-up required for FRONTEND. If you plan on using FRONTEND

for a long time on a specific program that is SMP/E maintained, you could build FRONTEND as an SMP/E user modification. In that way, if maintenance is applied to the original program, an SMP/E regression report would be generated alerting you to the fact that the original program had been replaced by FRONTEND. You could then restore the user modification, apply the maintenance, and then reapply the user modification. In a similar fashion, maintenance attempted with AMASPZAP VER and REP control cards should also (hopefully) fail with the data not being verified successfully.

ADDITIONAL MODIFICATIONS

There are a few areas where you may wish to perform further research before using FRONTEND. For example, I have not tested using FRONTEND to invoke TSO commands that are normally invoked from the TSOEXEC command, or using it to invoke authorized batch programs. There is also the issue of using FRONTEND on programs that have alias names.

This could be accomplished by renaming the original program aliases along with the true name, but would require assembling and linking multiple copies of FRONTEND so each could transfer control (using the XCTL macro) to the appropriate renamed true or alias name.

FRONTEND

```
         PUNCH ' SETCODE AC(1) '    NEED AUTHORIZATION IN BATCH
FRONTEND CSECT
FRONTEND AMODE 31                   OUCB ABOVE THE LINE
FRONTEND RMODE 24
         YREGS                      REGISTER EQUATES
         SAVE  (14,12),,FRONTEND-&SYSDATE
         LR    R12,R15              LOAD BASE REGISTER
         USING FRONTEND,R12
         LR    R14,R13              COPY CALLER'S SAVE AREA POINTER
         LA    R13,SAVEAREA         POINT TO MY SAVE AREA
         ST    R13,8(,R14)          CHAIN SAVE ...
         ST    R14,4(,R13)          ... AREAS.
         LR    R5,R1                SAVE A(PARM)-POSSIBLY A CPPL
         L     R3,CVTPTR            LOAD A(CVT)
         USING CVT,R3
         L     R3,CVTTCBP           LOAD A(TCBWORDS)
         L     R3,12(,R3)           LOAD A(CURR ASCB)
         DROP  R3
         USING ASCB,R3
```

```
         L     R4,ASCBOUCB          LOAD A(OUCB)
         USING OUCB,R4
         CLC   OUCBSUBN,=CL4'TSO'   ARE WE RUNNING TSO FOREGROUND
         BE    TSOFORE              YES, GO HANDLE TSO FOREGROUND
         L     R1,ASCBASXB          LOAD A(ASXB)
         USING ASXB,R1
         ICM   R1,15,ASXBLWA        IS THERE AN LWA (TSO BACKGROUND)
         BZ    BATSTC               NO, GO HANDLE BAT/STC ENVIRONMENT
         MVC   TSOLOC(4),=C'back'   SAY BACKGROUND
         B     TSOMODE              CONTINUE TSO PROCESSING
TSOFORE  MVC   TSOLOC(4),=C'fore'   SAY FOREROUND
         MVC   TSOREST(NOTFYLEN),NOTFYSTR ADJUST COMMAND
         MVI   TSOREST+NOTFYLEN,C' ' BLANK OUT REMAINDER
         MVC   TSOREST+NOTFYLEN+1(TSOBLNK+1),TSOREST+NOTFYLEN AGAIN
         B     ARNDTMVC             SKIP TSO JOBNAME CHECK
TSOMODE  ICM   R1,15,ASCBJBNI       IS THERE A TSO JOBNAME POINTER
         BNZ   MVCTSON              YES, GO USE IT
         L     R1,ASCBJBNS          ELSE USE ALTERNATE JOBNAME POINTER
MVCTSON  MVC   TSOJBNM(8),Ø(R1)     MOVE TSO BATCH JOBNAME
         USING CPPL,R5
ARNDTMVC L     R6,CPPLCBUF          SAVE A(REAL COMMAND BUFFER)
         LA    R15,TSOCMD           LOAD A(TSO SEND COMMAND)
         ST    R15,CPPLCBUF         STORE IT AS FAKE COMMAND BUFFER
         LR    R1,R5                RELOAD A(CPPL) FOR SEND COMMAND
         LINK  EP=SEND              LINK TO SEND COMMAND
         ST    R6,CPPLCBUF          RESTORE A(REAL CMD BUFFER) TO CPPL
         B     RETURN               SKIP BATCH/STC PORTION
BATSTC   ICM   R1,15,ASCBJBNI       IS THERE A JOBNAME POINTER
         BNZ   MVCNAME              YES, GO USE IT
         L     R1,ASCBJBNS          ELSE USE ALTERNATE JOBNAME POINTER
MVCNAME  MVC   OPERCMD+39(8),Ø(R1)  MOVE STC JOBNAME
         TESTAUTH FCTN=1            CHECK IF WE ARE MARKED APF AUTH
         LTR   R15,R15              TEST RETURN CODE FROM TESTAUTH
         BNZ   RETURN               SKIP OPER SEND IF NOT AUTHORIZED
         MODESET KEY=ZERO,MODE=SUP  ELSE GET SET FOR SVC 34
         SR    RØ,RØ                CLEAR PARM REG
         LA    R1,OPERCMD           POINT TO COMMAND TO BE ISSUED
         SVC   34                   ISSUE MVS COMMAND
         MODESET KEY=NZERO,MODE=PROB GET UNAUTHORIZED
RETURN   L     R13,SAVEAREA+4       POINT TO CALLER'S SAVEAREA
         LM    R14,R12,12(R13)      RELOAD CALLER'S REGISTERS
         DROP  R12
         USING FRONTEND,R15
         XCTL  EPLOC=XCTLPGM        TRANSFER CONTROL TO DESIRED PROGRAM
SAVEAREA DS    18F                  SAVE AREA
XCTLPGM  DC    CL8'IEFBR14 '        RENAMED PGM TO XCTL TO
TSOCMD   DS    ØF                   TSO COMMAND BUFFER HEADER---------|
TSOFLAGS DC    AL2(TSOCMDL,3)       COMAND LENGTH, OFFSET TO OPERAND  |
         DC    C'SE ''FRONTEND being run in TSO '                     |
TSOLOC   DC    c'****ground '       FORE/BACK                         |
TSOREST  DC    C'as '               TARGET OF COMMAND ADJUSTMENT---|  |
TSOJBNM  DC    CL8'********',C' '   JOBNAME FIELD                  |  |
```

```
TSOBLNK  EQU   *-TSOREST           LENGTH EQUATE-----------------|  |
NOTFYSTR EQU   *                  STRING TO (MAYBE) SHIFT UP---|    |
         DC    C'by'' U('                                     |    |
         DC    C'TSOBBOR'          USERID TO NOTIFY           |    |
         DC    C' '                DELIMITER                  |    |
         DC    C'TSOBBOR'          USERID TO NOTIFY           |    |
         DC    C') LOGON'          END OF SEND COMMAND        |    |
NOTFYLEN EQU   *-NOTFYSTR          LENGTH EQUATE--------------|    |
TSOCMDL  EQU   *-TSOCMD            LENGTH EQUATE--------------------|
OPERCMD  DS    ØF                  OPERATOR COMMAND---------------|
         DC    AL2(OPERCMDL,Ø)     LENGTH/FLAGS                   |
         DC    C'SE ''FRONTEND being run in batch by ********'',USER=('
         DC    C'TSOBBOR'          USERID TO NOTIFY               |
         DC    C','                DELIMITER                      |
         DC    C'TSOBBOR'          USERID TO NOTIFY               |
         DC    C'),LOGON'          END OF SEND COMMAND            |
OPERCMDL EQU   *-OPERCMD           LENGTH EQUATE------------------|
         LTORG
         PRINT NOGEN
         CVT   DSECT=YES,LIST=NO
         IHAASCB
         IHAASXB
         IRAOUCB
         IKJCPPL
         END
```


Cancelling TSO sessions

INTRODUCTION

When shutting down an MVS system prior to an IPL or when deactivating a cluster controller for maintenance, etc, it is often necessary to cancel a potentially large number of TSO sessions if the users ignore the usual 'please log off' broadcast messages. To ease this task I have implemented a simple started task, TSOABEND, which can be used via a simple console dialogue to cancel all TSO sessions, or only those on selected VTAM nodes.

THE TSOABEND PROGRAM

The TSOABEND program runs as a started task and may be started in one of two modes, CANCEL or LIST (the default), via a start command parameter, ie:

```
S TSOABEND,OPT={CANCEL|LIST}
```

In CANCEL mode, the selected TSO sessions are actually cancelled, in LIST mode they are simply listed with an indication that they would have been cancelled. In either case, a WTOR of the form:

```
TSA001R Cancel active TSO sessions - Reply END, ALL, or line
```

is issued and a reply waited for. This reply should be one of the following:

- END – terminate with no further processing.
- ALL – cancel all active TSO sessions.
- Line – where 'line' is at least the first three characters of a VTAM line address, eg ADE0, ATDA, etc. This will cause all sessions on lines starting with these characters to be cancelled, for example, if ADE0 is specified, all sessions on lines ADE0xxxx will be cancelled.

In CANCEL mode, as each session is cancelled, a console message is issued:

```
TSA001I User usernme on line linename has been cancelled
```

whereas in LIST mode, each session that *would* have been cancelled is listed via the message:

```
TSA002I User usernme on line linename would have been cancelled
```

When running in CANCEL mode, when TSOABEND terminates, either implicitly after an ALL request, or explicitly after an END request, it issues a summary message:

```
TSA003I TSOABEND normal termination, nnn sessions cancelled
```

In outline, the processing performed by TSOABEND consists of the following steps:

- The parm field is checked – if it contains 'C{ANCEL}' the processing mode is set to CANCEL, if it is null or contains anything other than C{ANCEL} the processing mode defaults to LIST.
- A check is made that the program is running as a started task - if not it terminates with an error message. This is to prevent

unauthorized use by someone running the program from a TSO session or as a batch job. We assume that only authorized users have access to a console to issue the Start command.

The TSA001R WTOR is issued and a reply waited for:

- If the reply is END the program terminates.
- If the reply is ALL, the cancel process is started.

Any other reply is assumed to be a partial or complete VTAM node name. The only requirement is that it is at least three characters long (imposed by our particular VTAM node naming convention).

The ASVT is scanned for active TSO address spaces. When one is found, its Terminal Status Block (TSB) is located and the TSBTRMID (node name) extracted. Note that the TSB is in fetch protected key 5 storage. If cancelling all sessions, processing jumps to the next step, otherwise the supplied node name is checked against the session's node name. If it matches, processing jumps to the next step, if not the session is bypassed and the ASVT scan continued.

If running in CANCEL mode a

```
C U=usernme,A=asid
```

command is built and issued via an MGCR macro, and a TSA001I message issued. Note the use of the A=asid parameter - this is necessary because we allow privileged users to have multiple sessions under the same usernme. If running in LIST mode, a TSA002I message is issued.

The ASVT scan continues until all active address spaces have been checked. If cancelling by node name, the TSA001R WTOR is then issued again to get another node name, or an END (or indeed ALL) reply.

On receipt of an END reply, or at the end of the ASVT scan after an ALL reply, TSOABEND exits, with the TSA003I message, if running in CANCEL mode.

OPERATIONAL ENVIRONMENT

TSOABEND requires to run APF-authorized and hence should be link-edited with AC(1) into a secure APF-authorized private library.

It should *not* be placed in the Linklist. The JCL to run the program should be placed in SYS1.PROCLIB or another suitable procedure library.

TSOABEND was written for use on an MVS/ESA 4.2.2 system, but should work on both earlier and later versions.

JCL FOR THE TSOABEND STARTED TASK

```
//TSOABEND PROC OPT=LIST
//*
//*****************************************************************//
//*    RUN THE TSOABEND STARTED TASK TO CANCEL TSO SESSIONS         *//
//*                                                                 *//
//*    TO RUN IN LIST MODE   : S TSOABEND,OPT=LIST                  *//
//*                                                                 *//
//*    TO RUN IN CANCEL MODE : S TSOABEND,OPT=CANCEL                *//
//*****************************************************************//
//*
//TSOABEND EXEC PGM=TSOABEND,PARM='&OPT'
//STEPLIB  DD   DSN=SYS1.APFLIB,DISP=SHR
//*
//SYSUDUMP DD   SYSOUT=X
//*
```

SOURCE CODE FOR THE TSOABEND PROGRAM

```
         TITLE 'TSOABEND - Cancel TSO Sessions'
*********************************************************************
*  PROGRAM TSOABEND
*  ________
*  This program, run as an authorized started task,  will cancel all
*  active TSO sessions, or only those on a specified cluster.  It is
*  intended for operator use either during comms work,  or  when the
*  the system is being shut down.
*
*  If invoked from a TSO session  or a batch job,  a highlit message
*  identifying  the  offending  user or job will be displayed on the
*  console and the program will terminate.
*
*  TSOABEND has two modes of operation, selectable at start time:
*
*  1) CANCEL mode - the default if a simple S TSOABEND is issued.
*  2) LIST   mode,  selectable  by  specifying OPT=LIST on the START
*     command,  ie S TSOABEND,OPT=LIST.  This mode simply causes the
*     sessions that  would  have been cancelled in CANCEL mode to be
*     listed on the console.
*
* In either mode, a WTOR will be displayed :
```

```
*
*     TSAØØ1R Cancel active TSO sessions - Reply END, ALL, or line
*
*  to which one of the following replies should be made:
*
*  END    : Terminate with no further processing.
*  ALL    : Cancel all active TSO sessions.
*  line   : Where 'line' is at least the first three characters of a
*           VTAM line address,  eg ADEØ, ATDA, etc.  This will cause
*           all  sessions on lines starting with these characters to
*           be cancelled,  eg if ADEØ  is specified, all sessions on
*           lines ADEØxxxx will be cancelled.
*
*  If ALL is specified,  the program will terminate when done;  if a
*  line name was specified a new name  will  be  requested when that
*  node is cleared.   When  all  required lines have been specified,
*  a reply of 'END' will terminate the program.
*
*  Environmental requirements:
*
*    STATE    : Problem
*    KEY      : 8 and Ø
*    APF      : Yes
*    AMODE    : 31
*    RMODE    : 24
*    LOCATION : Private APF-authorized library
*********************************************************************
         EJECT
TSOABEND CSECT
TSOABEND AMODE 31
TSOABEND RMODE 24
*
RØ       EQU     Ø               * USED BY MACROS
R1       EQU     1               * USED BY MACROS
R2       EQU     2               *
R3       EQU     3               *
R4       EQU     4               * WORK REGISTER
R5       EQU     5               * WORK REGISTER
R6       EQU     6               * LENGTH OF (PARTIAL) LINE NAME
R7       EQU     7               * UID ADDRESS
R8       EQU     8               * TSB ADDRESS
R9       EQU     9               * ASCB ADDRESS
R1Ø      EQU     1Ø              * CVT ADDRESS
R11      EQU     11              * ASVT ADDRESS
R12      EQU     12              * BASE REGISTER
R13      EQU     13              * SAVEAREA
R14      EQU     14              * RETURN ADDRESS
R15      EQU     15              * ENTRY ADDRESS/RETURN CODE
B     START-*(R15)       * BRANCH TO CODE
         DC    AL1(NT2-NT1)      * LENGTH OF NAME TEXT
NT1      EQU   *
         DC    C'TSOABEND'       * MODULE NAME
NT2      EQU   *
```

```
         DC    C' '
         DC    CL8'&SYSDATE'      * DATE
         DC    C' '
         DC    CL5'&SYSTIME'      * TIME
         DS    ØF                 * ALIGN TO FULL WORD BOUNDARY
*********************************************************************
* ADDRESSABILITY AND LINKAGE
*********************************************************************
START    EQU   *
         STM   R14,R12,12(R13)    * SAVE REGISTERS IN CALLERS SAVEAREA
         BASR  R12,Ø              * LOAD BASE REGISTER R12
         USING *,R12              * DEFINE R12 AS BASE REGISTER
*
         LR    R11,R13            * R11 = ADDRESS OF CALLERS SAVEAREA
*        LA    R13,SAVEAREA       * R13 = ADDRESS OF OUR     SAVEAREA
         ST    R11,4(R13)         * STORE HSA ADDRESS
         ST    R13,8(R11)         * STORE LSA ADDRESS
*
         L     R1,Ø(R1)           * R1 = ADDRESS OF PARM FIELD
         LH    R2,Ø(R1)           * R2 = LENGTH OF PARM FIELD
         LTR   R2,R2              * TEST VALUE
         BZ    CHKAUTH            * NO PARM -> DEFAULT 'LIST' ASSUMED
*
         CLI   2(R1),C'C'         * C FOR CANCEL SPECIFIED ?
         BNE   CHKAUTH            * NO, SO ASSUME 'LIST'
         MVI   PARMOPT,C'C'       * YES, SO UPDATE PARM OPTION FLAG
*-------------------------------------
* ARE WE PROPERLY AUTHORISED?
*-------------------------------------
CHKAUTH  EQU   *
         L     R1Ø,16                  * DEFINE CVT ...
         USING CVT,R1Ø                 * ... ADDRESSABILITY
         L     R1Ø,CVTTCBP             * GET IEATCBP ADDRESS
         DROP  R1Ø                     * FINISHED WITH CVT
*
         L     R9,12(R1Ø)              * GET CURRENT ASCB ADDRESS
         USING ASCB,R9                 * DEFINE ASCB ADDRESSABILITY
*
         LA    R15,12                  * SET POSSIBLE RETURN CODE
         CLC   ASCBASCB(4),=C'ASCB'    * IS ASCB REALLY AN ASCB ?
         BNE   RETURN                  * NO SO QUIT FAST
*
         ICM   R7,B'1111',ASCBJBNI     * GET @(JOBNAME)
         BNZ   NOTAUTH1                * NON ZERO => BATCH JOB
*
         ICM   R8,B'1111',ASCBTSB      * GET @(TSB)
         BZ    REQOPT                  * ZERO => STARTED TASK
*
         L     R7,ASCBJBNS             * GET @(TSO UID)
         B     NOTAUTH2                * NOT ZERO => TSO SESSION
*
         DROP  R9                      * FINISHED WITH ASCB
         EJECT
*********************************************************************
```

```
* WE ARE AUTHORISED. REQUEST (NEXT) CANCEL OPTION FROM OPERATOR
*********************************************************************
REQOPT   EQU   *
         XC    WTOECB,WTOECB             * CLEAR WTOR ECB
         MVI   REPLY,C' '                * AND REPLY ...
         MVC   REPLY+1(L'REPLY-1),REPLY  * ... AREA
*
         WTOR  'TSA00001R Cancel active TSO sessions - Reply END, ALL, or+
                line',REPLY,8,WTOECB,ROUTCDE=(2)
*
         WAIT  ECB=WTOECB                * WAIT FOR REPLY
*
* CHECK REPLY. LINE NAMES MUST BE AT LEAST 3 CHARACTERS LONG. IF OK,
* R6 WILL CONTAIN LENGTH-1 OF THE LINE NAME READY FOR EXECUTED CLC
*
         CLC   REPLY(3),END              * END ?
         BE    RETURN00                  * YES, SO QUIT
*
         CLC   REPLY(3),ALL              * ALL ?
         BE    INITSCAN                  * YES
*
         LA    R5,REPLY+7                * ADDRESS OF LAST REPLY CHAR
         LA    R6,7                      * INITIAL CHARACTER COUNT - 1
RCHK0001 EQU   *
         CLI   0(R5),C' '                * BLANK ?
         BNE   RCHK0002                  * NOPE
         BCTR  R6,0                      * DECREMENT CHARACTER COUNT
         BCT   R5,RCHK0001               * AND LOOP BACK
*
RCHK0002 EQU   *
         LA    R4,2                      * MINIMUM LINE NAME LENGTH - 1
         CR    R6,R4                     * CHECK AGAINST ACTUAL
         BL    REQOPT                    * IF SHORT ASK OPS AGAIN
         EJECT
*-------------------------------------
* SCAN THE ASVT FOR TSO SESSIONS ON THE ENTERED TERMINAL ADDRESS(S)
*-------------------------------------
* INITIALISE ASVT SCAN
*
INITSCAN EQU   *
         L     R10,16                    * DEFINE CVT ...
         USING CVT,R10                   * ... ADDRESSABILITY
         L     R11,CVTASVT               * ASVT BASE SEGMENT ADDRESS
         DROP  R10                       * FINISHED WITH CVT
*
         USING ASVT,R11                  * ASVT ADDRESSABILITY
         L     R10,ASVTMAXU              * MAXIMUM NUMBER OF ASIDS
         LA    R11,ASVTENTY              * ADDRESS OF FIRST ASCB
         DROP  R11                       * FINISHED WITH ASVT
*
* SEARCH FOR THE NEXT ACTIVE ASCB.
*
ASIDLOOP EQU   *
```

```
         TM    Ø(R11),ASVTAVAL           * ASID ASSIGNED ?
         BZ    ASSIGNED                  * YES IF BIT Ø IS ZERO
*
NEXTASID EQU   *
         LA    R11,4(R11)                * MOVE TO NEXT ASID
         BCT   R1Ø,ASIDLOOP              * AND LOOP BACK ...
         B     ENDASID                   * ... UNTIL END OF LIST
* ACTIVE ASCB FOUND. REJECT ALL NON-TSO SESSIONS
*
ASSIGNED EQU   *
         L     R9,Ø(R11)                 * GET ASCB ADDRESS
         USING ASCB,R9                   * DEFINE ASCB ADDRESSABILITY
         ICM   RØ,B'1111',ASCBJBNI       * GET @(JOBNAME)
         BNZ   NEXTASID                  * NON ZERO => BATCH JOB
         ICM   R8,B'1111',ASCBTSB        * GET @(TSB)
         BZ    NEXTASID                  * ZERO => STARTED TASK
         ICM   R7,B'1111',ASCBJBNS       * GET @(USERID)
         BZ    NEXTASID                  * GIVE UP IF ZERO
         DROP  R9                        * FINISHED WITH ASCB
*
* A BONA-FIDE TSO SESSION. EXTRACT THE TERMINAL ID FROM THE TSB AND
* CHECK IF IT MATCHES THE ENTERED (PARTIAL) LINE NAME.
*
         MODESET KEY=ZERO                * GET INTO KEY ZERO
*
         USING TSB,R8                    * DEFINE TSB ADDRESSABILITY
         MVC   TERMID,TSBTRMID           * EXTRACT TERMINAL ID FROM TSB
         DROP  R8                        * FINISHED WITH TSB
*
         MODESET KEY=NZERO               * REVERT TO KEY 8
*
         CLC   REPLY(3),ALL              * ALL SESSIONS TO DIE ?
         BE    KILLSESS                  * YUP
*
         EX    R6,COMPLINE               * COMPARE LINE AND TERMID
         BE    KILLSESS                  * A MATCH, SO CANCEL SESSION
         B     NEXTASID                  * NO MATCH, SO SKIP IT
*
COMPLINE CLC   TERMID(Ø),REPLY           * EXECUTED CLC
         EJECT
*-------------------------------------
* THIS SESSION FULFILS ALL THE CRITERIA FOR CANCELLING ...
*-------------------------------------
KILLSESS EQU   *
         L     R5,NCANCEL                * INCREMENT ...
         LA    R5,1(R5)                  * ... SESSIONS CANCELLED ...
         ST    R5,NCANCEL                *     ... COUNTER
*
         CLI   PARMOPT,C'C'              * CANCEL PARM SPECIFIED ?
         BNE   MSGONLY                   * NO, SO JUST DISPLAY MESSAGE
*
* CANCEL PARM SPECIFIED SO BUILD AND ISSUE CANCEL COMMAND
*
```

```
         MVC   COMMUID,Ø(R7)            * MOVE UID INTO C U= COMMAND
*
         USING ASCB,R9                  * DEFINE ASCB ADDRESSABILITY
         LH    R5,ASCBASID              * GET ASID
         DROP  R9                       * FINISHED WITH ASCB
         SLDL  R4,4                     * SHIFT IN A DUMMY SIGN NIBBLE
         STM   R4,R5,DOUBLE             * STORE IT AS PSEUDO-PACKED
         UNPK  COMMASID,DOUBLE+5(3)     * UNPACK ASID
         NC    COMMASID,ZONEMASK        * CONVERT ZONES TO ZEROS
         TR    COMMASID,HEXTAB          * CONVERT TO EBCDIC
*
         MODESET KEY=ZERO               * GET INTO KEY ZERO AGAIN
*
         SR    RØ,RØ                    * RØ = Ø
         LA    R1,COMMAND               * R1 = ADDRESS OF COMMAND
         SVC   34                       * ISSUE MGCR SVC
*
         MODESET KEY=NZERO              * REVERT TO KEY 8
*
         MVC   KILLWTO+21(7),COMMUID    * MOVE UID INTO WTO
         MVC   KILLWTO+36(8),TERMID     * MOVE TERMINAL ID INTO WTO
         CNOP  Ø,4
KILLWTO  WTO   'TSAØØ1I User ??????? on line ???????? has been cancelle+
               d',ROUTCDE=(2),DESC=(3)
*
         B     NEXTASID                 * LOOP BACK FOR NEXT ASID
*
* CANCEL PARM NOT SPECIFIED, SO JUST DISPLAY WHAT WOULD HAVE HAPPENED.
*
MSGONLY  EQU   *
         MVC   DISPWTO+21(7),Ø(R7)      * MOVE UID INTO WTO
         MVC   DISPWTO+36(8),TERMID     * MOVE TERMINAL ID INTO WTO
         CNOP  Ø,4
DISPWTO  WTO   'TSAØØ2I User ??????? on line ???????? would have been c+
               ancelled',ROUTCDE=(2),DESC=(6)
*
         B     NEXTASID                 * LOOP BACK FOR NEXT ASID
         EJECT
*-------------------------------------
* END OF ASID LOOP. IF A LINE WAS SPECIFIED ASK OPS FOR ANOTHER
*-------------------------------------
ENDASID  EQU   *
         CLC   REPLY(3),ALL             * ALL SESSIONS DEAD ?
         BNE   REQOPT                   * IF NOT ASK FOR NEXT LINE
         EJECT
*********************************************************************
* ALL DONE, SO ISSUE TERMINATION MESSAGE AND RETURN
*********************************************************************
RETURNØ  EQU   *
         CLI   PARMOPT,C'C'             * CANCEL PARM SPECIFIED ?
         BNE   RETURN                   * NO, SO EXIT WITHOUT MSG
*
         L     R5,NCANCEL               * EDIT ...
         CVD   R5,DOUBLE                * ... # CANCELS ...
```

```
         ED    ENDWTO+44(4),DOUBLE+6     *     ... INTO MESSAGE
         CNOP  Ø,4
ENDWTO   WTO   'TSAØØ3I TSOABEND normal termination; ??? sessions cance+
               lled',ROUTCDE=(2),DESC=(3)
         SR     R15,R15                  * SET RETURN CODE TO ZERO
*
RETURN   EQU   *
         L     R13,4(R13)               * RESTORE HSA ADDRESS
         L     R14,12(R13)              * RESTORE R14
         LM    RØ,R12,2Ø(R13)           * RESTORE RØ-R12
         BR    R14                      * AND RETURN
         EJECT
*********************************************************************
* ERROR CONDITIONS
*********************************************************************
NOTAUTH1 EQU   *
         MVC   NA1WTO+6Ø(8),Ø(R7)
         CNOP  Ø,4
NA1WTO   WTO   'TSAØØ1E Unauthorized attempt to run TSOABEND by job ???+
               ?????',ROUTCDE=(2),DESC=(2)
         LA    R15,4                    * SET RETURN CODE
         B     RETURN                   * AND RETURN
NOTAUTH2 EQU   *
         MVC   NA2WTO+61(7),Ø(R7)
         CNOP  Ø,4
NA2WTO   WTO   'TSAØØ2E Unauthorised attempt to run TSOABEND by user ??+
               ?????',ROUTCDE=(2),DESC=(2)
         LA    R15,8                    * SET RETURN CODE
         B     RETURN                   * AND RETURN
         EJECT
*********************************************************************
* CONSTANTS AND DATA AREAS
*********************************************************************
         DS    ØD
         DC    CL8'SAVEAREA'
SAVEAREA DC    18F'Ø'                   * SAVE AREA
WTOECB   DS    F
NCANCEL  DC    F'Ø'
REPLY    DS    CL8
TERMID   DS    CL8
DOUBLE   DS    D
         DS    ØF
ZONEMASK DC    XL8'ØFØFØFØFØFØFØFØF'
HEXTAB   DC    CL16'Ø123456789ABCDEF'
*
PARMOPT  DC    CL1'L'
END      DC    CL3'END'
ALL      DC    CL3'ALL'
```

P R S Wright
Associate Consultant
Tessella Support Services plc (UK)

Year 2000 aid: generation of edit macros – part 2

This month we complete our look at the Assembler module YEAR2KER which produces ISPF edit macros by providing the STDEF macros as defined in the descriptions of programs YEAR2K and YEAR2KR.

```
         DOQ   &A
&T       SETC  '&UNQ'
.*
         DOQ   &R
&O       SETC  '&UNQ'
.*
&I       SETA  72
.*
         AIF   ('&B' NE 'P' AND '&C' NE 'P' AND '&D' NE 'P').NOTP
         PUNCH ' ISREDIT CHANGE 1 &I ALL PREFIX &T &O'
.*
.NOTP    AIF   ('&B' NE 'S' AND '&C' NE 'S' AND '&D' NE 'S').NOTS
         PUNCH ' ISREDIT CHANGE 1 &I ALL SUFFIX &T &O'
.*
.NOTS    AIF   ('&B' NE 'W' AND '&C' NE 'W' AND '&D' NE 'W').NOTW
         PUNCH ' ISREDIT CHANGE 1 &I ALL WORD &T &O'
.*
.NOTW    AIF   (T'&B NE 'O').END
         PUNCH ' ISREDIT CHANGE 1 &I ALL &T &O'
.*
.END     MEND
*
         PUNCH 'ISREDIT MACRO (HELP) NOPROCESS'
         PUNCH 'ISPEXEC CONTROL ERRORS RETURN'
         PUNCH ' IF &&HELP = ? THEN DO '
         PUNCH '      ISPEXEC DISPLAY PANEL(YEAR2KRP)'
         PUNCH '      EXIT'
         PUNCH '    END'
         PUNCH '  ISREDIT CHANGE ALL DATE-WRITTEN TO @#$%-WRITTEN'
         PUNCH '  ISREDIT CHANGE ALL DATE-COMPILED TO @#$%-COMPILED'
         PUNCH '  ISREDIT EXCLUDE ALL'
*********************************************************************
***   THE 'STDEF' MACRO INSTRUCTIONS, BELOW, WERE CUT FROM SOURCE   ***
***   'YEAR2KR' AND PASTED THERE.                                   ***
*********************************************************************
         STDEF SPACE,'C'' ''',W
         STDEF ZERO,LOW-VALUE
         STDEF XYZ-DATE,XYZ-NEW-DATE,OPTION=FORCE
         STDEF XYZ-YY,XYZ-CCYY
         STDEF 'QUOTE''TEST',NEW''QUOTE''TEST
*********************************************************************
***   THE 'STDEF' MACRO INSTRUCTIONS, ABOVE, WERE CUT FROM SOURCE   ***
***   'YEAR2KR' AND PASTED THERE.                                   ***
*********************************************************************
```

```
        PUNCH '  ISREDIT CHANGE ALL @#$%-WRITTEN TO DATE-WRITTEN'
        PUNCH '  ISREDIT CHANGE ALL @#$%-COMPILED TO DATE-COMPILED'
        PUNCH '  ISREDIT EXCLUDE ALL DATE-WRITTEN.'
        PUNCH '  ISREDIT EXCLUDE ALL DATE-COMPILED.'
        PUNCH '  EXIT CODE(Ø)'
        END
```

JCL FOR ASSEMBLING MODULES

The following JCL assembles module YEAR2KE which produces the ISPF EDIT macro YEAR2K. The JCL assumes that the source for the module is in the partitioned dataset YEAR2K.SOURCE.LIBRARY. To obtain EDIT macro YEAR2KR, simply edit it and issue the following EDIT command:

```
     c all year2k year2kr
```

```
//SYSTØØ2L JOB ,'KEITH NICAISE',NOTIFY=SYSTØØ2,REGION=1Ø24K,
//             CLASS=A,MSGLEVEL=(1,1),MSGCLASS=X
//*-------------------------------------------------------------------*//
//*    ASSEMBLE 'YEAR2KE' TO PRODUCE EDIT MACROS TO SCAN FOR
//*    POTENTIAL YEAR 2ØØØ PROBLEM STRINGS OF SOURCE PROGRAMS.
//*-------------------------------------------------------------------*//
//S1   EXEC ASMHC,
//          PARM.C=(NOOBJECT,'XREF(SHORT)',DECK,TERM,ALIGN,
//         'LINECOUNT(55)')
//C.SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//C.SYSPUNCH DD DSN=YEAR2K.PUNCH,DISP=OLD
//C.SYSPRINT DD SYSOUT=*
//C.SYSTERM  DD SYSOUT=*
//C.SYSIN   DD DSN=YEAR2K.SOURCE.LIBRARY(YEAR2KE),DISP=SHR
```

GENERATED YEAR2K ISPF EDIT MACRO

```
ISREDIT MACRO (HELP) NOPROCESS
ISPEXEC CONTROL ERRORS RETURN
 IF &HELP = ? THEN DO
      ISPEXEC DISPLAY PANEL(YEAR2KP)
      EXIT
    END
  ISREDIT EXCLUDE ALL
 ISREDIT FIND 1 68 ALL PREFIX "AGE"
 ISREDIT FIND 1 68 ALL WORD "AGE"
 ISREDIT FIND 1 66 ALL PREFIX "BIRTH"
 ISREDIT FIND 1 66 ALL WORD "BIRTH"
 ISREDIT FIND 1 63 ALL "CALENDAR"
 ISREDIT FIND 1 64 ALL "CENTURY"
 ISREDIT FIND 1 65 ALL "CSADAT"
 ISREDIT FIND 1 65 ALL "CSAEID"
```

```
 ISREDIT FIND 1 65 ALL "CSAJYD"
 ISREDIT FIND 1 67 ALL PREFIX "DATE"
 ISREDIT FIND 1 67 ALL WORD "DATE"
 ISREDIT FIND 1 68 ALL "DMY"
 ISREDIT FIND 1 64 ALL "GREGJUL"
 ISREDIT FIND 1 62 ALL "GREGORIAN"
 ISREDIT FIND 1 64 ALL "JULGREG"
 ISREDIT FIND 1 65 ALL "JULIAN"
 ISREDIT FIND 1 68 ALL "MDY"
 ISREDIT FIND 1 65 ALL "MMDDYY"
 ISREDIT FIND 1 63 ALL "SCHEDULE"
 ISREDIT FIND 1 66 ALL WORD "TODAY"
 ISREDIT FIND 1 67 ALL "YEAR"
 ISREDIT FIND 1 68 ALL "YDD"
 ISREDIT FIND 1 69 ALL PREFIX "YM"
 ISREDIT FIND 1 69 ALL SUFFIX "YM"
 ISREDIT FIND 1 69 ALL WORD "YM"
 ISREDIT FIND 1 68 ALL "YMD"
 ISREDIT FIND 1 69 ALL "YY"
  ISREDIT EXCLUDE ALL DATE-WRITTEN.
  ISREDIT EXCLUDE ALL DATE-COMPILED.
  EXIT CODE(Ø)
```

GENERATED YEAR2KR ISPF EDIT MACRO

```
ISREDIT MACRO (HELP) NOPROCESS
ISPEXEC CONTROL ERRORS RETURN
 IF &HELP = ? THEN DO
      ISPEXEC DISPLAY PANEL(YEAR2KRP)
      EXIT
    END
  ISREDIT CHANGE ALL DATE-WRITTEN TO @#$%-WRITTEN
  ISREDIT CHANGE ALL DATE-COMPILED TO @#$%-COMPILED
  ISREDIT EXCLUDE ALL
 ISREDIT CHANGE 1 72 ALL WORD "SPACE" "C' '"
 ISREDIT CHANGE 1 72 ALL "ZERO" "LOW-VALUE"
 ISREDIT CHANGE 1 72 ALL "XYZ-DATE" "XYZ-NEW-DATE"
 ISREDIT CHANGE 1 72 ALL "XYZ-YY" "XYZ-CCYY"
 ISREDIT CHANGE 1 72 ALL "QUOTE'TEST" "NEW'QUOTE'TEST"
  ISREDIT CHANGE ALL @#$%-WRITTEN TO DATE-WRITTEN
  ISREDIT CHANGE ALL @#$%-COMPILED TO DATE-COMPILED
  ISREDIT EXCLUDE ALL DATE-WRITTEN.
  ISREDIT EXCLUDE ALL DATE-COMPILED.
  EXIT CODE(Ø)
```

Keith H Nicaise
Technical Services Manager
Touro Infirmary (USA)

MVS news

Sterling Software has announced the delivery of new capabilities for SAMS:Vantage for MVS. SAMS:Vantage delivers automation, interactive reporting, analysis and predictive modelling capabilities, and provides device-specific information across multiple RAID storage subsystems.

SAMS:Vantage's new scripting language and wizards make the automation capabilities easier to use. The system can now automatically find and fix the exact storage problem, down to the data set level, in one step. It can also take an incremental approach, making an unlimited number of passes, each using increasingly complex criteria, to find and fix problems. The SAMS:Vantage HSMPlus component now supplements IBM's DFSMShsm with more efficient monitoring of DFSMShsm functions. Pricing begins at $13,310 for Group 30.

Sterling Software Inc, Storage Management Division, 11050 White Rock Road, Ste 100, Rancho Cordova, CA 95670-6095, USA,
Tel: (916) 635 5535
Fax: (916) 635 5604 or
Sterling Software (UK) Ltd, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB.
Tel: (0181) 867 8000
Fax: (0181) 867 8001.

* * *

Prince Software has announced Release 3.0 of Translate 2000, an automated COBOL renovation product which now incorporates WinExpress.

TRANSLATE 2000 combines the functions of program scanning, logic analysis and automated repair and is available for MVS-based systems

For further information contact:
Prince Software, 3 Pearl Ct, Allendale, NJ 07401, USA.
Tel: (201) 934 0022
Fax: (201) 934 0220.

* * *

AMI Software has announced its Automated Compliance Testing (ACT) suite of testing software tools designed to complete remediation and testing of MVS applications for the year 2000. The ACT suite consists of a series of modules, each of which addresses a specific area of year 2000 compliance. The ACT software tools facilitate the identification of date fields within data and provide a facility to migrate them back and forth synchronously to make sure they are year 2000 compliant.

For further information contact:
AMI Software Limited, Fernlea House, Newby. Penrith, Cumbria, CA10 3EX, UK.
Tel: (01931) 714053
Fax: (01931) 714054.

* * *