147

VM

update

November 1998

In this issue

VM Update

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Editorial panel
Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($22.50) each including postage.

Editor
Robert Burgess

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.

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

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Printed in England.

Operating the non-shared resource

GENERAL DESCRIPTION

A non-shared resource may be a single CMS file or a SQL/DS PRIVATE table, which cannot be accessed simultaneously. Usually the non-shared resource must be locked when it is being updated in real time by different virtual machines. In normal circumstances, a lock may be required and read-only access allowed to synchronize the use of the resource.

Two solutions to this problem are available. To provide high reliability, they are based on existing CMS tools. Both isolate the non-shared resource by having only one virtual machine access it – a resource control machine or a server. All other virtual machines are defined as requestors of the resource and communicate with the server. The solutions differ in the communication tool used and the location of the data queue:

1 The communication tool is an Inter-User Communication Vehicle (IUCV) and the data queue is on the 191 A disk of the requestor – this solution allows up to 101 requestor virtual machines to access the non-shared resource.

2 Communication is achieved by the CMS macros PUNCHC and RDCARD. In this case the data queue is on the CP spool and the requestor virtual machines do not lock. Therefore, they cannot determine the moment at which the server finishes processing their requests.

The modules are written in Assembler and the program code is developed in CMS with VM/SP Release 5.

IUCV COMMUNICATIONS AND DATA QUEUE ON 191 DISK

This solution is supported by the modules IUCVREQ and IUCVSRV. These are called from a REXX program and are used to synchronize requestor and server machine actions.

IUCVREQ and IUCVSRV are invoked as shown:

```
IUCVREQ <server id>  <data>
IUCVSRV <exec id>
```

where:

- <server id> is the name of the server virtual machine.
- <data> is a character string with a length of up to eight bytes, which must not contain blanks.
- <exec id> is the identifier of the REXX EXEC which processes the parameter <data>.

The REXX EXEC <exec id> is started by IUCVSRV in the server virtual machine when communication is established between the requestor and the server. IUCVSRV passes the <data> string, received from the requestor, as a parameter to the REXX EXEC <exec id>. The requestor is in a wait state until the end of the execution of <exec id>.

If a second requestor virtual machine (B) wishes to communicate with a server when there is already a communication path between the server and the first requestor virtual machine (A), the requestor B is put into a queue and dropped into a wait state. Up to 101 requestor virtual machines may be held in a queue.

The sequence of events, also shown in Figure 1, is as follows:

1 IUCVSRV is invoked with the parameter WORKEXEC.

2 The server virtual machine waits for a connection.

3 The requestor virtual machine creates a file REQUESTOR DATA A.

4 IUCVREQ is invoked with the parameters SERVER, which is the name of the server machine, and IDENTDATA, which is the string identifier request.

5 The server establishes a communication path to the requestor.

6 The server starts WORKEXEC EXEC internally with the received parameter IDENTDATA. WORKEXEC EXEC accepts the parameter IDENTDATA, links to the 191 disk of the requestor,

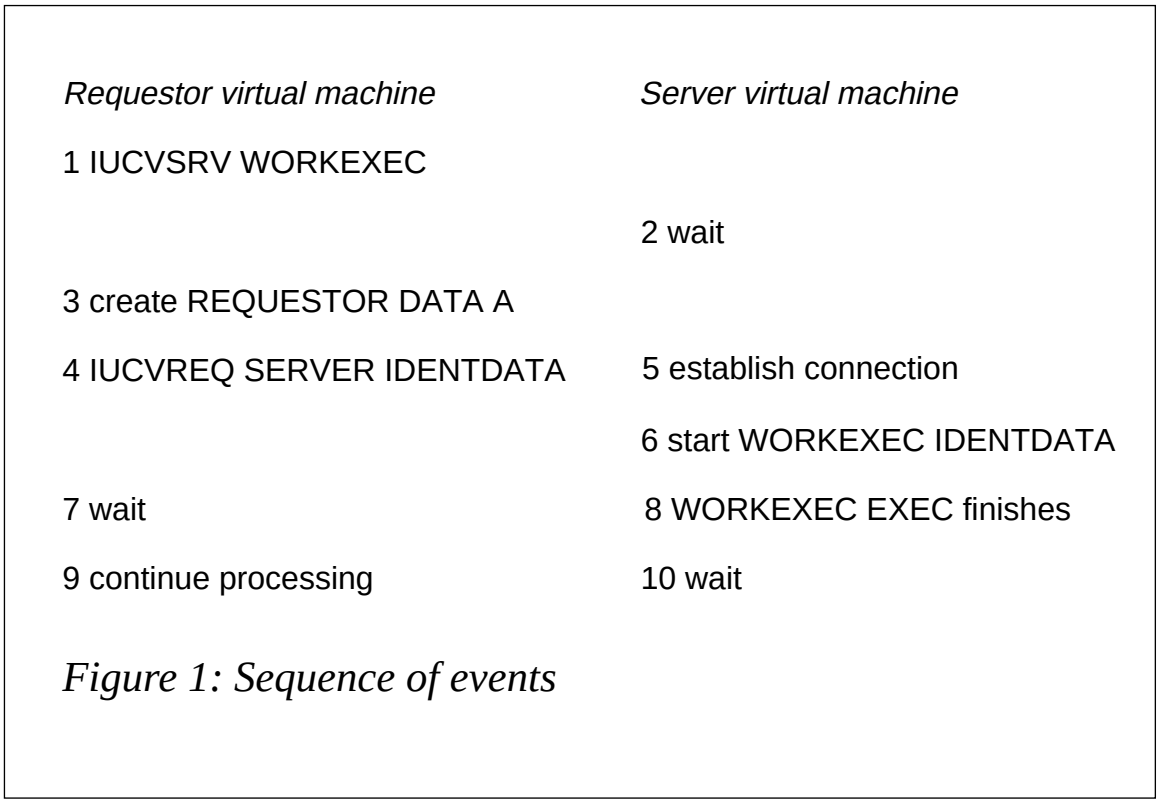

Figure 1: Sequence of events

and processes the file REQUESTOR DATA A.

7 The requestor is in a wait state during WORKEXEC EXEC execution.

8 WORKEXEC EXEC completes.

9 The requestor unlocks and continues processing.

10 The server waits for the next connection.

PUNCHC-RDCARD COMMUNICATION

This solution supports only one-way communication. The requestor virtual machines are not locked and they cannot determine when their data has been processed by the server virtual machine. Therefore, this solution should be used only when the succeeding data processing in the requestor virtual machine does not depend on data processing in the server virtual machine. This may be a terminal load of a file or PRIVATE SQL/DS table by an operator, or users account data collection in the SQL/DS table or sequential file.

CMSPNCH and CMSRCRD achieve PUNCHC-RDCARD communication. They may be called from programs written in Assembler or PL/I. CMSPNCH punches a card to the spool and CMSRCRD reads a card from spool.

The corresponding PLISTs are:

- CMSPNCH (<card>)
- CMSRCRD (<card>,<read len>).

where:

- <card> has a string length of 80 bytes. If the first byte of <card> is the ‘+’character then CMSPNCH closes the spool file.
- <read len> is half-word binary. Before the call, this contains a number of processed bytes and must have a value between 1 and 80. After the call, this contains the RDCARD macro return code.

An example of the use of CMSPNCH and CMSRCRD is shown in Figure 2.

INSTALL EXEC

```
/*********************************************************************/
/***                                                    ***         ***/
/*** INSTALL    generate IUCVREQ and IUCVSRV MODULE     ***         ***/
/***                                                    ***         ***/
/*********************************************************************/
/***   SIZE 00073  VER 1.0 MOD 000                                   ***/
/*********************************************************************/

  CLRSCRN
  MESSAGE = 'user request'
  SAY ' — Start IUCVREQ and IUCVSRV MODULE generation-Reply Y or N'
  PULL REPLY
  IF REPLY ¬= 'Y' THEN
  SIGNAL ERROR
  MESSAGE = 'error when 194 disk VM MAINT is accessed'
  SAY '— Enter password for read-only access of 194 disk VM MAINT'
  PULL REPLY .
  IF REPLY = '' THEN
  SIGNAL ERROR
  SAY '— Disk X will be released'
  SAY '— Disk 001 will be detached'
  SET CMSTYPE HT
```

```
                        requestor
                        virtual machine

   PL/I                                              Assembler
   CARD = 'FIRST CARD';                              CALL CMSPNCH,(FCARD)
   CALL CMSPNCH(CARD);                               CALL CMSPNCH,(LCARD)
   CARD = 'LAST CARD';                               CALL CMSPNCH,(CLOSE)
   CALL CMSPNCH(CARD);
   CARD = '+';                                       FCARD DC   CL80'FIRST CARD'
   CALL CMSPNCH(CARD);                               LCARD DC   CL80'LAST CARD'
   DCL CMSPNCH ENTRY OPTIONS (ASM INTER);            CLOSE DC   C'+'
   DCL CARD CHAR (80) AUTO;

                        server
                        virtual machine

   CALL CMSRCRD (CARD, LEN);                         CALL CMSRCRD,(CARD,LEN)
   IF I = 1 THEN                                     CLI  LEN+1,X'01'
   GOTO END_OF_FILE;                                 BE   EOF
   IF I = 2 THEN                                     CLI  LEN+1,X'02'
   GOTO END_OF_FILE;                                 BE   EOF
   IF I ¬= 0 THEN                                    CLI  LEN+1,X'00'
   IF I ¬= 5 THEN                                    BE   READOK
   GOTO READ_ERROR;                                  CLI  LEN+1,X'05'
   DCL CMSRCRD ENTRY OPTIONS (ASM INTER);            BE   READOK
   DCL CARD CHAR (80) AUTO;                          B    ERROR
   DCL LEN FIXED BIN (15) STATIC INIT (50);    CARD DS   CL80
   /* only first 50 bytes are processed */     LEN DC   H'50'
```

Figure 2: Example of CMSPNCH and CMSRCRD use

```
SIGNAL OFF ERROR
  REL X
  DIAG(8, DET 001, 128)
SIGNAL ON ERROR
  DIAG(8, LINK MAINT 194 001 RR REPLY, 128)
  AC 001 X
  RESTORE_MACLIBS = 'Y'
  QUERY MACLIB '(' STACK
  PULL LIBID EQUAL_SIGN OLD_MAC_LIBS
  MESSAGE = 'error when declare maclibs'
  GLOBAL MACLIB DMSSP CMSLIB OSMACRO DMKSP
  IUCV.1 = 'IUCVREQ'
  IUCV.2 = 'IUCVSRV'
```

```
  DO I = 1 TO 2
    MESSAGE = 'error when assemble' IUCV.I
    STATE IUCV.I MODULE A
    SAVE_RC = RC
    SET CMSTYPE RT
    IF SAVE_RC = Ø THEN
    DO
      SAY '- 'IUCV.I MODULE 'found on disk A'
      SAY '- Replace' IUCV.I MODULE A '- reply Y or N'
      PULL REPLY
      IF REPLY ¬= 'Y' THEN
      SIGNAL ERROR
    END
    SET CMSTYPE HT
    SIGNAL ON ERROR
    ASSEMBLE IUCV.I
    ERASE IUCV.I LISTING A
    MESSAGE = 'error when load' IUCV.I
    LOAD IUCV.I '(' NOMAP NOLIBE
    MESSAGE = 'error when genmod' IUCV.I
    GENMOD
    ERASE IUCV.I TEXT A
    SIGNAL OFF ERROR
    SET CMSTYPE RT
    SAY '- 'IUCV.I MODULE 'generated successfully'
  END
  IF OLD_MAC_LIBS ¬= 'NONE' THEN
  GLOBAL MACLIB OLD_MAC_LIBS
  SAY '- Generation completed'
  SIGNAL EXIT
ERROR:
  SET CMSTYPE RT
  SAY '-  IUCVREQ or IUCVSRV MODULE not generated due to -'
  SAY '     'MESSAGE
EXIT:
  IF RESTORE_MACLIBS = 'Y' THEN
    IF OLD_MAC_LIBS ¬= 'NONE' THEN
  GLOBAL MACLIB OLD_MAC_LIBS
```

CMSPNCH

```
*************************************************************************
****                                                  ***          ****
**** CMSPNCH                punch card                ***          ****
****                                                  ***          ****
*************************************************************************
****   SIZE ØØØ38  VER 1.Ø MOD ØØØ                                  ****
*************************************************************************
*                                                                       *
CMSPNCH  CSECT
```

```
         SAVE  (14,3)
         BALR  2,Ø
         USING *,2
         ST    13,SA+4
         LA    13,SA
         L     3,Ø(1)
         CLI   Ø(3),C'+'
         BE    CLOSE
         PUNCHC (3)
         B     RET
CLOSE    EQU   *
         LA    1,PLIST
         LA    Ø,EXTPLIST
         ICM   1,8,=X'Ø2'
         SVC   2Ø2
         DC    AL4(1)
RET      EQU   *
         L     13,4(13)
         RETURN (14,3)
SA       DC    18F'Ø'
PLIST    EQU   *
         DC    CL8'CMS'
EXTPLIST EQU   *
         DC    A(PLIST)
         DC    A(CLOSEPU)
         DC    A(CLOSEPU+L'CLOSEPU)
         DC    A(Ø)
CLOSEPU  DC    C'SP PU CLOSE'
         END   CMSPNCH
```

CMSRCRD

```
*************************************************************************
****                                                    ***         ****
**** CMSRCRD                 read card                 ***         ****
****                                                    ***         ****
*************************************************************************
****   SIZE ØØØ24  VER 1.Ø MOD ØØØ                                  ****
*************************************************************************
*                                                                       *
CMSRCRD  CSECT
         SAVE  (14,5)
         BALR  2,Ø
         USING *,2
         ST    13,SA+4
         LA    13,SA
         L     3,Ø(1)
         L     4,4(1)
         LR    5,4
         LH    4,Ø(4)
```

```
         RDCARD (3),(4),RDAHEAD=YES
         STH   15,Ø(5)
         L     13,4(13)
         RETURN (14,5)
SA       DS    18F
         END   CMSRCRD
```

IUCVREQ

```
*************************************************************************
****                                                    ***         ****
**** IUCVREQ              IUCV requestor               ***         ****
****                                                    ***         ****
*************************************************************************
****   SIZE ØØØ53  VER 1.Ø MOD ØØØ                                  ****
*************************************************************************
*                                                                       *
IUCVREQ  CSECT
         USING *,12
         LR    11,14
         MVC   SERVER(8),8(1)
         MVC   PARMS(8),16(1)
         LA    Ø,SYSTNAME
         LA    1,GETLEN
         DIAG  Ø,1,X'ØØ'
         LA    2,IUCVPLST
         USING IPARML,2
         LA    3,ECB
         HNDIUCV SET,NAME=REQSTR,EXIT=EXTEXIT
         IUCV  CONNECT,PRMLIST=IUCVPLST,PRMDATA=YES,USERID=SERVER,     X
               USERDTA=SERVER,MF=L
         CMSIUCV CONNECT,NAME=REQSTR,PRMLIST=IUCVPLST,ERROR=RET
         WAITECB ECB=ECB,FORMAT=OS
         XC    ECB(4),ECB
         IUCV  SEND,PRMLIST=IUCVPLST,DATA=PRMMSG,PATHID=PATH,          X
               PRMMSG=PARMS,TYPE=2WAY
         WAITECB ECB=(3),FORMAT=OS
         XC    ECB(4),ECB
         IUCV  SEVER,PRMLIST=IUCVPLST,PATHID=PATH,MF=L
         CMSIUCV SEVER,NAME=REQSTR,PRMLIST=IUCVPLST
         HNDIUCV CLR,NAME=REQSTR
         SR    15,15
RET      EQU   *
         BR    11
         DS    ØD
IUCVPLST DC    4ØX'ØØ'
SERVER   DC    CL8'SERVER'
PARMS    DC    CL8'PARMS'
SYSTNAME DS    CL8
VERMPROC DS    CL8
```

```
REQSTR   DS    CL8
GETLEN   EQU   *-SYSTNAME
         DS    ØD
EXTEXIT  EQU   *
         USING *,15
         MVC   PATH(2),Ø(2)
         OI    ECB,X'4Ø'
         BR    14
ECB      DC    F'Ø'
PATH     DS    2X
         COPY  IPARML
         END   IUCVREQ
```

IUCVSRV

```
*************************************************************************
****                                                  ***          ****
**** IUCVSRV                IUCV server              ***          ****
****                                                  ***          ****
*************************************************************************
****   SIZE ØØ11Ø  VER 1.Ø MOD ØØØ                                  ****
*************************************************************************
*                                                                       *
IUCVSRV  CSECT
         USING *,12
         LR    11,14
         MVC   EXECID(8),8(1)
         LA    Ø,SYSTNAME
         LA    1,GETLEN
         DIAG  Ø,1,X'ØØ'
         LA    2,IUCVPLST
         USING IPARML,2
         LA    3,ECB
         HNDIUCV SET,NAME=SERVER,EXIT=EXTEXIT
WAITFOR  EQU   *
         LA    9,QUE+1ØØ
         LA    8,1
         LA    7,QUE
FROMQUE  EQU   *
         CLI   Ø(7),X'FF'
         BNE   THISPROC
         BXLE  7,8,FROMQUE
         B     WAITEXT
THISPROC EQU   *
         MVC   PATH+1(1),Ø(7)
         MVI   Ø(7),X'FF'
         B     PROCQUE
WAITEXT  EQU   *
         WAITECB ECB=(3),FORMAT=OS
         XC    ECB(4),ECB
```

```
PROCQUE  EQU   *
         IUCV  ACCEPT,PRMLIST=IUCVPLST,PATHID=PATH,PRMDATA=YES,MF=L
         CMSIUCV ACCEPT,PRMLIST=IUCVPLST,NAME=SERVER
         WAITECB ECB=(3),FORMAT=OS
         XC    ECB(4),ECB
         IUCV  RECEIVE,PRMLIST=IUCVPLST,PATHID=PATH
         MVC   PARAM(8),PARMS
         LA    Ø,EXTPLIST
         LA    1,PLIST
         ICM   1,8,=X'Ø2'
         SVC   2Ø2
         DC    AL4(1)
         IUCV  REPLY,PRMLIST=IUCVPLST,PATHID=PATH,DATA=PRMMSG,          X
               PRMMSG=SERVER,MSGID=MSGID,TRGCLS=MSGCLS
         WAITECB ECB=(3),FORMAT=OS
         XC    ECB(4),ECB
         IUCV  SEVER,PRMLIST=IUCVPLST,PATHID=PATH,MF=L
         CMSIUCV SEVER,NAME=SERVER,PRMLIST=IUCVPLST
         MVI   PATH+1,X'FF'
         B     WAITFOR
         DS    ØD
EXTPLIST EQU   *
         DC    A(PLIST)
         DC    A(EXECID)
         DC    A(PARAM+L'PARAM)
         DC    A(Ø)
PLIST    EQU   *
CMSID    DC    CL8'CMS'
EXECID   DC    CL8'EXECID'
PAD      DC    X'4Ø'
PARAM    DC    CL8'PARAM'
         DS    ØD
IUCVPLST DC    4ØX'ØØ'
SYSTNAME DS    CL8
VERMPROC DS    CL8
SERVER   DS    CL8
GETLEN   EQU   *-SYSTNAME
         DS    ØD
EXTEXIT  EQU   *
         SAVE (14,12)
         BALR 11,Ø
         USING *,11
         ST   13,EXITSA+4
         LA   13,EXITSA
         CLI   2(2),X'8Ø'
         BNE   POST
         CLI   PATH+1,X'FF'
         BE    POST
         LA    9,QUE+1ØØ
         LA    8,1
```

```
         LA    7,QUE
CHECK    EQU   *
         CLI   Ø(7),X'FF'
         BE    FILLIN
         BXLE  7,8,CHECK
         ABEND 1313
FILLIN   EQU   *
         MVC   Ø(1,7),1(2)
         B     EXITEXIT
POST     EQU   *
         MVC   PATH(X'14'),Ø(2)
         OI    ECB,X'4Ø'
EXITEXIT EQU   *
         L     13,EXITSA+4
         RETURN (14,12)
EXITSA   DS    18F
ECB      DC    F'Ø'
PATH     DC    AL1(X'ØØ',X'FF')
         DS    2X
MSGID    DS    CL4
MSGCLS   DS    CL4
PARMS    DS    CL8
QUE      DC    1Ø1X'FF'
         COPY  IPARML
         END   IUCVSRV
```

IUCVREQ AND IUCVSRV PREPARATION

INSTALL EXEC should be used to generate IUCVREQ MODULE and IUCVSRV MODULE. IUCV ALLOW must be added to the system directory of the server virtual machine.

Dobrin Goranov
Information Services Co (Bulgaria)

Administering multiple machines – part 3

This month we conclude the collection of REXX procedures written to help administer and control multiple machines in an installation.

ES11 EXEC

```
/* ================================================================== */
/* Name        :  ES11  EXEC                                          */
```

```
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  EventId | EventNumber                               */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Display Help for Events                             */
/*                                                                    */
/* ================================================================== */
arg id_or_number

o.0=0
ofid='event helptext a'
es.tags=''  /* to prevent variable substitution */

if datatype(id_or_number,'W')
then
   call help_for_number
else
   call help_for_id id_or_number

'pipe stem o. | >' ofid
'browse' ofid
return

/* ================================================================== */
/* help for number                                                    */
/* ================================================================== */
help_for_number:

logrec=es10(id_or_number)
if logrec = ''
then
   do
     say 'Event Number' id_or_number 'not found in logfile'
     exit
   end

call out '── Event Data ──────────────────'
call out ' '

recdef=es08()  /* get the logdef */
ps='76 EVENTSTRING 476 .'
interpret 'parse var logrec' ps  /* get the pure event string */
interpret es02(eventstring) /* extract variables from it */

do i=1 to words(es.tags)
```

```
   tag=word(es.tags,i)
   call out left(tag,20)':  ' es.tag
end

call out ' '
call help_for_id es.id
return

/* ================================================================== */
/* help for id                                                        */
/* ================================================================== */
help_for_id:
arg event_id

edef=eS17('eventdef.'event_id)
if edef = ''
then
   do
     call out 'No event definition found for event id' event_id
     return
   end
parse value edef  with . 'title(' title ')' .
parse value edef  with . 'helptext(' helptext ')' .

call out '── Description ──────────────────'
call out ' '
call out title
if helptext ¬= ''
then
   do
     call out ' '
     'pipe <' helptext '| stem recs.'
     do j=1 to recs.0
        do i=1 to words(es.tags)
           tag=word(es.tags,i)
           p=pos('&'tag,recs.j)
           if p > 0
           then
              do
                recs.j=left(recs.j,p-1) || ,
                strip(es.tag) || ,
                substr(recs.j,p+length(tag)+1)
              end
        end
     end
     'pipe stem recs. | stem o. append'
   end
return

/* ================================================================== */
```

```
/*                                                                    */
/* ================================================================== */
out:
parse arg orec
'pipe var orec | stem o. append'
return
```

ES12 EXEC

```
#|/usr/bin/rexx
/* ================================================================== */
/* Name        :  ES12 EXEC                                           */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  id, tag contents, tag contents, ...                 */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Shellscript-Frontend for ESØ6                       */
/*                                                                    */
/* This procedure serves as an interface between a shell script       */
/* and ESØ6. Shell scripts cannot call ESØ6 directly because of the   */
/* way the parameters are handled in REXX procedures                  */
/*                                                                    */
/* call example:                                                      */
/*                                                                    */
/* parm1='FSNAME anyFileSystem'                                       */
/* parm2='USED 99%'                                                   */
/* parm3='LIMIT 9Ø%'                                                  */
/* ES12    'AIXFSFULL',$parm1,$parm2,$parm3                           */
/*                                                                    */
/* ================================================================== */
parse arg parms
parse var parms args.1 ',' parms

i=1
argstring='args.1'
do forever
   parse var parms parm ',' parms
   if parm = '' then leave
   i=i+1
   args.i=parm
   argstring=argstring',args.'i
end
interpret 'call ESØ6' argstring
return
```

ES13 EXEC

```
/* ================================================================== */
/* Name        :  ES13  EXEC                                          */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  -                                                   */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  Browse Event Logfile                                */
/*                                                                    */
/* ================================================================== */

/* ================================================================== */
/* inits                                                              */
/* ================================================================== */
s_id=''
s_origin=''
s_number=''
logfile=es17('event.logfile')
logdef=esØ8()   /* log recdef */
parsestring='1 W_NUMBER 16 . 16 W_ID 36 . 36 W_ORIGIN 56 .' ,
  '56 W_TIMESTAMP 76 . 76 W_EVENTSTRING 476 .'
id_fromto='16-35'
origin_fromto='36-55'

cursor=''             /* f r ios327Ø */
message=''            /* f r ios327Ø */

/* ================================================================== */
/* Main process                                                       */
/* ================================================================== */

fm=esØ9()  /* access the disk */
do forever
   call panel 'es14'
   select
      when iosk =  'PFØ3' then leave
      when iosk =  'PFØ6'   /* reset logfile */
        then
           call es16
      when iosk =  'ENTER' then call process
      otherwise call process
   end
end
call diskrel fm
```

```
return Ø
/* ================================================================== */
/* make selection and display                                         */
/* ================================================================== */
process:
s_id=translate(strip(s_id))
s_origin=translate(strip(s_origin))
find=''
if strip(s_id) ¬= ''
then
   do
     find=find '| locate' id_fromto '/'s_id'/'
   end
if strip(s_origin) ¬= ''
then
   do
     find=find '| locate' origin_fromto '/'s_origin'/'
   end
'pipe <' logfile   find  '| stem li.'
do i=1 to li.Ø
/*    interpret 'parse value li.i with' parsestring                */
   sl.i=left(li.i,74)
end
sl.Ø=li.Ø
call display_list
return

/* ================================================================== */
/* display list                                                       */
/* ================================================================== */
display_list:

cp=1   /* current pointer in select list */
dm=15  /* display maximum on screen */
sel=''   /* selection entered by user */
selected_task=''
do forever
   /* load display stem dl. from list stem sl. */
   do ixØ1=1 to dm
      ix=ixØ1+cp-1
      if ix > sl.Ø
      then
         dl.ixØ1=' '
      else
         dl.ixØ1=sl.ix
   end
   call panel 'es15'
   select
      when iosk =  'PFØ3' then leave
      when iosk =  'PFØ2'
```

```
        then
           call display_details sel
      when iosk =  'PFØ7'
        then
           do
             cp=cp-dm
             if cp < 1 then cp=1
           end
      when iosk =  'PFØ8'
        then
           do
             cp=cp+dm
             if cp > sl.Ø then cp=cp-dm
           end
      when iosk =  'PFØ5'   /* top */
        then cp=1
      when iosk =  'PFØ4'   /* bot */
        then cp=sl.Ø-dm+1
      otherwise call process
   end
end
return
/* ================================================================== */
/* display details                                                    */
/* ================================================================== */
display_details:
call es11 s_number
return
/* ================================================================== */
/* I/O panel                                                          */
/* ================================================================== */
panel:
arg panelid
if panelmsg = 'PANELMSG' then panelmsg=''
if cursor   = 'CURSOR'   then cursor=''
if cursor ¬= ''
then    do; options='(UPDATE 'CURSOR ; alarm='.A'; end
else    do; options=''               ; alarm=''  ; end
'IOS327Ø' panelid options
if rc ¬= Ø then do; say 'IOS327Ø Error:' rc; exit; end
panelmsg='';cursor='';return
```

ES14 EXEC

```
;*================================================================== */
;* Name        :  es14     IOS327Ø                                    */
;*================================================================== */
;* Application :  Event Services                                      */
;*                                                                    */
;* Usage       :  IOS327Ø Panel                                       */
```

```
;*                                                                    */
;* Arguments   :  -                                                   */
;*                                                                    */
;* Result      :  -                                                   */
;*                                                                    */
;* Function    :  Panel for Event Log Viewer       (Selection)        */
;*                                                                    */
;*==================================================================== */
.n
.y
.F F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
.JX SET CTL . off
.jx set  normal intensity color=white
.jx set  high   intensity color=green
.JX SET CTL [ HIG=underline col=yel TYPE=(SKIP UNPROTECTED NULLS)
.JX SET CTL % col=red
.JX SET CTL # col=green
.JX SET CTL } col=green hig=rev
.e]
.&alarm
.c
}EventServices     Log Viewer
]
 %&MELDUNG
]

     ID           [20&s_id

     ORIGIN       [20&s_origin

     All events are shown when no selection criteria are entered
     ID and ORIGIN can be entered partially

.b
     ENTER:Display Log           PF3:Exit     PF6:Reset Log
```

ES15 EXEC

```
;*==================================================================== */
;* Name        :  ES15  IOS3270                                       */
;*==================================================================== */
;* Application :  Event Services                                      */
;*                                                                    */
;* Usage       :  IOS3270 Panel                                       */
;*                                                                    */
;* Arguments   :  -                                                   */
;*                                                                    */
;* Result      :  -                                                   */
;*                                                                    */
;* Function    :  Panel for Event Serv. Log Viewer (List)             */
```

```
;*                                                                          */
;*========================================================================= */
.n
.y
.F F1 F2 F3 F4 F5 F6 F7 F8 F9 F1Ø F11 F12
.JX SET CTL . off
.jx set  normal intensity color=white
.jx set  high   intensity color=green
.JX SET CTL [ HIG=underline col=yel TYPE=(SKIP UNPROTECTED NULLS)
.JX SET CTL # col=red
.JX SET CTL % col=yellow
.JX SET CTL # col=green
.JX SET CTL } col=green hig=rev
.e]
.&alarm
.c
}EventServices   LogViewer
]
 œ&message
]
%Number         ID                  Origin              Timestamp
&dl.1
&dl.2
&dl.3
&dl.4
&dl.5
&dl.6
&dl.7
&dl.8
&dl.9
&dl.1Ø
&dl.11
&dl.12
&dl.13
&dl.14
&dl.15
     Number [14&s_number
.b
     PF2:Show Details  PF3:Exit  PF4:Bot  PF5:Top  PF7:Up  PF8:Down
```

ES16 EXEC

```
/* ===================================================================== */
/* Name        :  ES16  EXEC                                             */
/* ===================================================================== */
/* Application :  Event Services                                         */
/*                                                                       */
/* Usage       :  Procedure                                              */
/*                                                                       */
/* Arguments   :  -                                                      */
```

```
/*                                                                   */
/* Result      :  -                                                  */
/*                                                                   */
/* Function    :  Send the event RESETLOG to reset the logfile       */
/*                                                                   */
/* ================================================================= */

call es01 'RESETLOG'
```

ES17 EXEC

```
/* ================================================================= */
/* Name        :  ES17     EXEC                                      */
/* ================================================================= */
/* Application :  Event Services                                     */
/*                                                                   */
/* Usage       :  Function                                           */
/*                                                                   */
/* Arguments   :  Keyword                                            */
/*                                                                   */
/* Result      :  value_string                                       */
/*                                                                   */
/* Function    :  Define/return global variables                     */
/*                                                                   */
/* ================================================================= */
arg keyword

select
  when keyword = 'EVENT.SERVER'        /* name of server machine */
    then return 'ESSERVER'
  when keyword = 'EVENT.SERVERIPADDR'  /* ip addr  of server machine */
    then return '131.102.22.89'
  when keyword = 'EVENT.LOGFILE'
    then return 'es logfile'
  when keyword = 'EVENT.SERVERPORT'    /* port where server listens */
    then return '1958'
  when keyword = 'EVENTDEF.TEST1'
    then return ,
         'title(test-event number 1)',
         'notify(sysh)',
         'action()',
         'helptext(esh001 text k)'
  when keyword = 'EVENTDEF.VMPAGINGFULL'
    then return ,
         'title(VM paging space nearly full)',
         'notify(sysh)',
         'action()',
         'helptext(esh002 text k)'
  when keyword = 'EVENTDEF.AIXFSFULL'
```

```
    then return ,
         'title(AIX file system nearly full)',
         'notify(sysh)',
         'action()',
         'helptext(esh003 text k)'
  otherwise return ''
end
```

ES18 EXEC

```
/* ================================================================== */
/* Name        :  ES18 EXEC                                           */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  -                                                   */
/*                                                                    */
/* Result      :  -                                                   */
/*                                                                    */
/* Function    :  VM System-Checker                                   */
/*                                                                    */
/* This procedure is called every n hours by VMUTIL.                  */
/* It checks the usage of paging space and sends the event            */
/* VMPAGINGFULL if more than 80% space is used                        */
/*                                                                    */
/* ================================================================== */

paging_threshold=003
'pipe cp q alloc page  | take last | var in'
parse value word(in,words(in)) with percent_used '%' .

if percent_used > paging_threshold
then
   call    es01 'vmpagingfull',,
                'USED' percent_used ,,
                'ALLOWED' paging_threshold
```

ES19 EXEC

```
#|/usr/bin/rexx
/* ================================================================== */
/* Name        :  ES19                                                */
/* ================================================================== */
/*                                                                    */
/* Created     :                                                      */
/*                                                                    */
```

```
/* Usage       :  REXX/6000 Procedure                                  */
/*                                                                     */
/* Arguments   :  -                                                    */
/*                                                                     */
/* Result      :  -                                                    */
/*                                                                     */
/* Function    :  Check filesystem usage in AIX                        */
/*                                                                     */
/* This procedure is started every n hours by crontab and              */
/* checks file system space usage.                                     */
/* For every file system more than 80% full, an event                  */
/* AIXFSFULL is sent to the event server in VM                         */
/*                                                                     */
/* ================================================================== */
hostname='hostname'
limit1=80
fslines.=''
fsn=0
call POPEN('df -kM')
do queued()
   parse pull inline
   if translate(word(inline,1)) = 'FILESYSTEM' then iterate
   fsn=fsn+1
   fslines.fsn=inline
end
do ix01=1 to fsn
   out=''
   inline=fslines.ix01
   fsname=word(inline,1)
   mntpnt=word(inline,2)
   usedsp=word(inline,5)
   if strip(usedsp,'T','%') > limit1
   then
      do
        out=out 'space used 'usedsp '(Limit:' limit1'%)'
        parm1='FSNAME' fsname
        parm2='USED' usedsp
        parm3='LIMIT' limit1'%'
        parm4='MNTPNT' mntpnt
        call ES06 'AIXFSFULL',parm1,parm2,parm3,parm4
      end
end

return
```


A full screen console interface – part 4

Editor's note: this month we continue the code for the full screen console interface for Disconnected Service Machines (DSM). This article is an extensive piece of work which will be published over several issues of VM Update*. It was felt that readers could benefit from the entire article and from the individual sections. Any comments or recommendations would be welcomed and should be addressed either to Xephon or directly to the author at fernando_duarte@vnet.ibm.com.*

THE CSC SERVICE PROGRAM (CSCSVP)

To generate a minimal service program you need the modules CSCSVP, CSCMSG, CSCMSL, CSCCFG, CSCRDF, and CSCCPW.

CSCSVP ASSEMBLE

```
         TITLE 'CSCSVP - CSC Service Machine Program'
*————————————————————————————————————*
*                                                                     *
*  CSCSVP Register usage                                              *
*                                                                     *
*              RØ-R3 Work registers                                   *
*              R4-R5 Work registers (carefully)                       *
*                                                                     *
*              R5  MSGSECT   MSG table                                *
*              R6  CCSBUFF   Scanning IUCV message                    *
*              R7  CCHSECT   Cache record                             *
*              R8  UIDSECT   User block                               *
*              R9  IUCV Parameter List                                *
*                                                                     *
*              R1Ø Base - Data area                                   *
*              R11 Base - Independent routines                        *
*              R12 Base - Common code                                 *
*————————————————————————————————————*
CSCSVP   START X'Ø14ØØØ'
CSCSVP   RMODE ANY
         PRINT NOGEN
         USING CSCSVP,R12             Base for common code
         USING CSCSVPD,R1Ø            Base for Data area
         USING IPARML,R9              IUCV Parameter List
         USING UIDSECT,R8             UID (user) Block
         USING CCHSECT,R7             CCH (cache) Block
```

```
         STM   R14,R12,12(R13)
         L     R1Ø,ACSCDATA            Address CSC Data area
         ST    R13,CSCSSV13            Save address of our save area
         B     INIT
ACSCDATA DC    A(CSCSVPD)
&DATE    SETC  '&SYSDATC'(1,4)'/'.'&SYSDATC'(5,2)'/'.'&SYSDATC'(7,2)
         DC    C'&DATE &SYSTIME'
         DC    C' Copyright CSC Inc, 1997'
         DS    ØH
         SPACE 3
*
* Init
*
*
INIT     MSG   ØØØ1                    Display initial message
         GO    CSCCFG                  Check Configuration file
         TM    CSCFLGØ1,CFGERROR       Any configuration errors?
         BO    CLOSE                   Yes, that's all for now...
         OI    CSCFLGØ2,CSCWAIT        Get ready to accept work
         BCTR  R2,Ø
         DIAG  R2,R3,X'ØØ24'           Get console address
         ST    R2,ADDRCONS
         HNDIO SET,DEVNAME=CONS,DEVICE=(R2),EXIT=CSCIOX,               *
               INTBLOK=(IOXBK,L'IOXBK)
         LTR   R15,R15                 Trap console interrupts
         BZ    INIT1ØØ                 No errors, keep going
         MSG   ØØØ2,RC
         B     CLOSE
         SPACE
INIT1ØØ  OI    CSCFLGØ1,HNDIOS         Remember to restore console
         HNDIUCV SET,NAME=CSCNAME,EXIT=CSCSTX
         LTR   R15,R15
         BZ    INIT2ØØ                 Good, IUCV enable
         LA    RØ,4
         CR    R15,RØ                  Is CSC already active?
         BNE   INIT11Ø
         MSG   ØØØ3,RC                 Yes, display nice message
         B     CLOSE
         SPACE
INIT11Ø  MSG   ØØØ4,RC                 No, display generic message
         B     CLOSE
         SPACE
INIT2ØØ  OI    CSCFLGØ1,HNDIUCVS       Remember it
         LA    R9,CSCPARMC             Address IUCV Parameter List
         MVC   IPVMID,CSCCPMSG         Move Service name (*MSG)
         MVC   IPUSER(8),CSCNAME       Keep CMS happy
         CMSIUCV CONNECT,NAME=CSCNAME,PRMLIST=CSCPARMC,EXIT=CSCCNX
         LTR   R15,R15
         BZ    INIT3ØØ                 IUCV *MSG Exit defined
         MSG   ØØØ5,RC
         B     CLOSE
```

```
         SPACE
INIT300  OI    CSCFLG01,CMSIUCVC       Remember it
         TM    CSCFLG01,CSCAPPC        Remote nodes defined?
         BZ    INIT400
         LA    R9,CSCPARMA             Address APPC/VM IUCV Parm List
         GO    CSCRNC                  Yes, initialize remote links
INIT400  MSG   0006                    All done, display message
         SPACE
PROCESS  BAS   R14,WAIT                Let's wait for some work to do
         TM    CSCFLG02,WORKIO
         BZ    PROC100
         BAS   R14,IOPROC              Console I/O interrupt
PROC100  TM    CSCFLG02,WORKCP
         BZ    PROC200
         BAS   R14,CPPROC              *MSG CP System Service
PROC200  TM    CSCFLG02,WORKID
         BZ    PROC300
         BAS   R14,IDPROC              IUCV interrupt (new session)
PROC300  TM    CSCFLG02,WORKMG
         BZ    PROC400
         BAS   R14,MGPROC              User incoming message
PROC400  TM    CSCFLG02,WORKRM
         BZ    PROC500
         GO    CSCRNL                  APPC/VM messages
PROC500  TM    CSCFLG02,WORKEND
         BZ    PROCESS
         SPACE 3
*
* Close the shop, turn off the lights
*
*
CLOSE    EQU   *
         MSG   0007                    We are terminating
         FSCLOSE FSCB=DFFILER          Close Data File
         FSCLOSE FSCB=DFFILEW
         TM    CSCFLG01,CSCAPPC        Remote nodes defined?
         BZ    CLOSE100
         LA    R9,CSCPARMA             Yes, IUCV Parameter List (APPC)
         GO    CSCRNCTR                Terminate APPC/VM
CLOSE100 LA    R9,CSCPARMU             IUCV Parameter List (Users)
         GO    CSCRLS                  Release all allocated storage
         LA    R9,CSCPARMC             IUCV Parameter List (*MSG)
         GO    CSCCLS                  Close the shop
         MSG   0008                    We are terminated
         L     R13,CSCSSV13            Address our save area
         L     R15,CSCRC               Load our return code
         ST    R15,16(,R13)            Store it into save area
         LM    R14,R12,12(R13)         Restore everything
         BR    R14                     Go back...
         SPACE 3
*
```

```
* Wait for something to do
*
*
WAIT     EQU   *                       Wait...
         WAITECB ECB=CSCECB
         XC    CSCECB,CSCECB           Clear ECB
         OI    CSCFLGØ2,CSCWAIT        Get ready to do more work
         BR    R14                     Check what happened
         SPACE 3
*
* HNDIO Exit Routine   Console I/O interrupts
*
*
CSCIOX   EQU   *                       HNDIO Exit Routine
         STM   R14,R12,12(R13)
         LA    RØ,CSCIOX-CSCSVP
         LR    R12,R15
         SR    R12,RØ                  Restore our base address (code)
         L     R1Ø,ACSCDATA            Restore our base address (data)
*        USING INTBLOK,R2
*        LA    R2,IOXBK                Did not work, nice try
*        L     R1,INTCCWAD             The idea was to keep the Console
*        LTR   R1,R1                   trap ON all the time...
*        BNZ   IOX9ØØ
         OI    CSCFLGØ2,WORKIO         Remember we had an interrupt
         TM    CSCFLGØ2,CSCWAIT
         BZ    IOX9ØØ
         NI    CSCFLGØ2,X'FF'-CSCWAIT  Post ECB once if required
         OI    CSCECB,CSCPOST
IOX9ØØ   LM    R14,R12,12(R13)
         SR    R15,R15                 Tell CMS everything is fine
         BR    R14                     Return
         SPACE 3
*
* CMSIUCV Exit Routine (CP)   IUCV Interrupts with CP System Services
*
*
CSCCNX   EQU   *                       CMSIUCV Exit Routine (CP)
         STM   R14,R12,12(R13)
         LA    RØ,CSCCNX-CSCSVP
         LR    R12,R15
         SR    R12,RØ
         L     R1Ø,ACSCDATA            Establish addressability
         OI    CSCFLGØ2,WORKCP         Remember we had an interrupt
         LR    R9,R2                   Address IUCV Parameter List
         LA    RØ,1                *T* Create trace entry
         BAS   R14,TRACE           *T*
         TM    CSCFLGØ3,CPFIRST
         BO    CNX2ØØ
         OI    CSCFLGØ3,CPFIRST        First time routine invoked
         CLI   IPTYPE,IPTYPCC          Must be connection complete
```

```
         BE    CNX100
         OI    CSCFLG03,CPCCERR        No, set error option
         B     CNX900
         SPACE
CNX100   OI    CSCFLG03,CPCC           CP accepted the connection
         B     CNX900
         SPACE
CNX200   TM    CSCFLG03,CPCC           Was Connection Complete received
         BO    CNX300
         OI    CSCFLG03,CPMSGERR       No, set error option
         B     CNX900
         SPACE
CNX300   CLI   IPTYPE,IPTYPMNP         Is this an incoming message?
         BE    CNX400
         OI    CSCFLG03,CPTYPERR       No, set error option
         B     CNX900
         SPACE
CNX400   LA    R0,1                    We have an incoming message
         A     R0,CPMSGQ
         ST    R0,CPMSGQ               Count queued messages
CNX900   TM    CSCFLG02,CSCWAIT
         BZ    CNX910
         NI    CSCFLG02,X'FF'-CSCWAIT  Post ECB once if required
         OI    CSCECB,CSCPOST
CNX910   LM    R14,R12,12(R13)
         BR    R14
         SPACE 3
*
* HNDIUCV Exit Routine (Users)   IUCV Connections pending
*
*
CSCSTX   EQU   *                       HNDIUCV Exit Routine
         STM   R14,R12,12(R13)
         LA    R0,CSCSTX-CSCSVP
         LR    R12,R15
         SR    R12,R0
         L     R10,ACSCDATA            Establish addressability
         OI    CSCFLG02,WORKID         Remember we had an interrupt
         LA    R0,UIDSIZE              Allocate storage for UID block
         BAS   R14,OBTAIN              Allocate storage
         LR    R8,R1                   Address UID block created
         LR    R9,R2                   Address IUCV Parameter List
         LA    R0,2                *T* Create trace entry
         BAS   R14,TRACE           *T*
         XC    UIDSECT(UIDSIZEB),UIDSECT Clear everything
         L     R1,IPPATHID             Load PATHID + FLAGS1 + IPTYPE
         ST    R1,UIDPID               Save in UID block
         MVC   UIDVMID,IPVMID          Copy also Userid
         MVC   UIDORIG,CSCLOCAL        Copy APPC/VM node name or blanks
         L     R1,UIDPTR               Queue block
         ST    R1,UIDFWD               Chain with previous blocks
```

```
         ST    R8,UIDPTR               Store new head of chain
         TM    CSCFLGØ2,CSCWAIT
         BZ    STX1ØØ
         NI    CSCFLGØ2,X'FF'-CSCWAIT  Post ECB once if required
         OI    CSCECB,CSCPOST
STX1ØØ   LM    R14,R12,12(R13)
         BR    R14
         SPACE 3
*
* CMSIUCV Exit Routine   IUCV Requests from users
*
*
CSCMGX   EQU   *                       CMSIUCV ACCEPT Exit Routine
         STM   R14,R12,12(R13)
         LA    RØ,CSCMGX-CSCSVP
         LR    R12,R15
         SR    R12,RØ
         L     R1Ø,ACSCDATA            Establish addressability
         OI    CSCFLGØ2,WORKMG         Remember we had an interrupt
         LA    R8,SSSPTR               Address chain of active sessions
         LR    R9,R2                   Address IUCV Parameter List
         LA    RØ,3                *T* Create trace entry
         BAS   R14,TRACE           *T*
MGX1ØØ   L     R8,UIDFWD               Scan list of active sessions
         CLC   IPPATHID,UIDPID         Check IUVC Path Id
         BNE   MGX1ØØ
         L     R1,IPPATHID             We found it
         ST    R1,UIDPID               Store PATHID + FLAGS1 + IPRCODE
         OI    UIDOPT1,UIDREQ          Set request pending option
         CLI   IPTYPE,IPTYPSV          Is connection being severed?
         BNE   MGX2ØØ
         OI    UIDOPT1,UIDSEV          Yes, set option in UID block
MGX2ØØ   CLI   IPTYPE,IPTYPRNP         Send completed?
         BNE   MGX9ØØ
         NI    UIDOPT1,X'FF'-UIDSEND   Yes, reset option in UID block
MGX9ØØ   TM    CSCFLGØ2,CSCWAIT
         BZ    MGX91Ø
         NI    CSCFLGØ2,X'FF'-CSCWAIT  Post ECB once if required
         OI    CSCECB,CSCPOST
MGX91Ø   LM    R14,R12,12(R13)
         BR    R14
         SPACE 3
*
* Process Console I/O   Process Console I/O interrupts
*
*
IOPROC   EQU   *                       Process Console I/O
         ST    R14,IOPRSV14
         NI    CSCFLGØ2,X'FF'-WORKIO   Reset option
         HNDIO CLR,DEVNAME=CONS        Disable Console trap
         LINERD DATA=CSCBUFF,TYPE=DIRECT,CASE=MIXED Read data
```

```
         LA    R6,CSCBUFF              Address input buffer
         AR    RØ,R6                   End address of entered data
         ST    RØ,CSCBUFFE             Store end address
         WAITT                         Wait until finished
         L     R2,ADDRCONS             Restore Console Interrupts trap
         HNDIO SET,DEVNAME=CONS,DEVICE=(R2),EXIT=CSCIOX,               *
               INTBLOK=(IOXBK,L'IOXBK)
         GO    CSCOPC                  Process command
         L     R14,IOPRSV14
         BR    R14
         SPACE 3
*
* Process work (CP)   *MSG messages
*
*
CPPROC   EQU   *                       Process work (CP)
         ST    R14,CPPRSV14
         NI    CSCFLGØ2,X'FF'-WORKCP   Rest option
         TM    CSCFLGØ3,CPCCERR        Any CP communication errors
         BZ    CPPR1ØØ
         LA    R15,11                  Yes, close the shop and bye bye
         MSG   ØØ11,RC
         B     CLOSE
         SPACE
CPPR1ØØ  TM    CSCFLGØ3,CPMSGERR       Any message errors?
         BZ    CPPR2ØØ
         LA    R15,11
         MSG   ØØ11,RC
         B     CLOSE
         SPACE
CPPR2ØØ  TM    CSCFLGØ3,CPTYPERR       Any unexpected interrupt?
         BZ    CPPR3ØØ
         LA    R15,12                  Yes, display message, keep going
         MSG   ØØ12,RC
         SPACE
CPPR3ØØ  L     R1,CPMSGQ               Any message queued
         LTR   R1,R1
         BZ    CPPR9ØØ                 No, all done
         BCTR  R1,Ø                    Yes, decrement counter
         ST    R1,CPMSGQ               Store new value
         LA    R9,CSCPARMC             Address IUCV Parameter List
         LA    RØ,4               *T*  Create trace entry
         BAS   R14,TRACE          *T*
         MVI   IPFLAGS1,IPFGPID        Select right message
         IUCV  RECEIVE,PRMLIST=CSCPARMC,BUFFER=CSCBUFF,BUFLEN=CSCBLEN
         BZ    CPPR4ØØ                 Check for errors
         SR    R15,R15
         IC    R15,IPRCODE             Load IPRCODE
         LA    RØ,5
         CR    R15,RØ
         BNE   CPPR31Ø
```

```
         MSG   0013,RC
         B     CPPR300                 Life goes on
         SPACE
CPPR310  MSG   0014,RC
         B     CPPR300
         SPACE
CPPR400  L     R1,IPBFADR1             End of message
         ST    R1,CSCBUFFE             Save end of message address
         GO    CSCCPW                  We got a message, put it on disk
         B     CPPR300                 Check for more messages
         SPACE
CPPR900  L     R14,CPPRSV14
         BR    R14
         SPACE 3
*
* Process IUCV Pending/Severed Connections
*
*
IDPROC   EQU   *
         USING USRSECT,R2
         ST    R14,IDPRSV14
         NI    CSCFLG02,X'FF'-WORKID   Reset option
         L     R8,UIDPTR               Address pending connections
         LA    R9,CSCPARMU             Address IUCV Parameter List
IDPR100  L     R1,UIDFWD               Remove first entry from...
         ST    R1,UIDPTR               ... list of pending connections
         CLI   UIDTYPE,IPTYPPC         Is it a Pending Connection?
         BNE   IDPR500
         L     R1,UIDPID               Yes, Get PATHID from UID block
         ST    R1,IPPATHID             Copy to IUCV Parameter List
         LA    R0,5                *T* Create trace entry
         BAS   R14,TRACE           *T*
         MVI   IPFLAGS1,X'00'          Clear all flags
         L     R1,USRPTR               Address User table
IDPR200  LTR   R2,R1                   Check for End-Of-Table
         BZ    IDPR600                 Found it, user not authorized
         L     R1,USRFWD               Address next user entry
         CLC   USRNAME,UIDVMID         Compare names
         BE    IDPR300                 Match, copy classes
         CLI   USRNAME,C'*'            Is it a generic id (*)
         BNE   IDPR200                 No, try next entry
         CLI   USRNAME+1,C' '          Make sure it is a single "*"
         BNE   IDPR200
IDPR300  L     R0,USRCLASS             Load classes from User table
         ST    R0,UIDCLASS             Store into UID block
         CMSIUCV ACCEPT,NAME=CSCNAME,PRMLIST=CSCPARMU,EXIT=CSCMGX
         LTR   R15,R15                 Check for errors
         BZ    IDPR400
         MSG   0015,RC                 Display error message
         LA    R0,UIDSIZE              De-allocate storage
         LR    R1,R8
```

```
         BAS   R14,RELEASE
         B     IDPR900
         SPACE
IDPR400  LA    R0,UIDSCRSZ             Get screen size
         BAS   R14,OBTAINP             Allocate storage (page aligned)
         ST    R1,UIDSCRN              Save address in UID block
         ST    R1,UIDSCRNA             Alternate screen not used yet
         LA    R0,UIDBUFSZ             User buffer size
         BAS   R14,OBTAINP             Allocate storage (page aligned)
         ST    R1,UIDBUFF
         L     R1,SSSPTR               Address active sessions list
         ST    R1,UIDFWD               Address new accepted session
         ST    R8,SSSPTR
         MSG   0016                    Display info message (Connected)
         B     IDPR900
         SPACE
IDPR500  CLI   UIDTYPE,IPTYPSV         Is this a severed connection?
         BNE   IDPR800
         L     R2,UIDPID               Get PATHID (first two bytes)
         LA    R0,UIDSIZE
         LR    R1,R8
         BAS   R14,RELEASE             Release UID block
         LR    R0,R2
         GO    CSCSEV                  Terminate IUCV session
         B     IDPR900
         SPACE
IDPR600  LA    R0,UIDSIZE              UID block size in double words
         LR    R1,R8                   Address of current block
         BAS   R14,RELEASE             Release storage
         MSG   0017
         CMSIUCV SEVER,NAME=CSCNAME,PRMLIST=CSCPARMU Terminate IUCV
         LTR   R15,R15
         BZ    IDPR900                 All done
         MSG   0018,RC                 Problem, unable to end session
         B     IDPR900                 Nothing else we can do
         SPACE
IDPR800  MSG   0019                    Not pending, not severed
******** ***** ********************** We will fix it, if it happens
 DC H'0'                               ******************************
*        L     R0,UIDPID               Get PATHID (first two bytes)
*        GO    CSCSEV                  Sever connection
*        B     IDPR900
         SPACE
IDPR900  L     R8,UIDPTR               Check all pending connections
         LTR   R8,R8
         BNZ   IDPR100
         L     R14,IDPRSV14
         BR    R14
         DROP  R2
         SPACE 3
*
```

```
* Process UICV User requests
*
*
MGPROC   EQU   *
         ST    R14,MGPRSV14
         NI    CSCFLGØ2,X'FF'-WORKMG   Reset option
         LA    R8,SSSPTR               Address active sessions list
         LA    R9,CSCPARMU             Address IUCV Parameter List
MGPR1ØØ  L     R8,UIDFWD               Scan all list
         LTR   R8,R8
         BZ    MGPR9ØØ
         TM    UIDOPT1,UIDREQ          Is there a request pending?
         BZ    MGPR1ØØ
         NI    UIDOPT1,X'FF'-UIDREQ    Yes, reset option
         TM    UIDOPT1,UIDSEV          Severed connection?
         BO    MGPR5ØØ
         CLI   UIDTYPE,IPTYPMNP        No, incoming message?
         BNE   MGPR6ØØ
         MVC   IPPATHID,UIDPID         Move PATHID
         MVI   IPFLAGS1,IPFGPID        Select right message
         LA    RØ,6                *T* Create trace entry
         BAS   R14,TRACE           *T*
         IUCV  RECEIVE,PRMLIST=CSCPARMU,BUFFER=CSCBUFF,BUFLEN=CSCBLEN
         BZ    MGPR2ØØ                 Check for errors
         SR    R15,R15
         IC    R15,IPRCODE
         LA    RØ,5
         CR    R15,RØ
         BNE   MGPR11Ø
         MSG   ØØ13,RC                 Display error message
         B     MGPR12Ø
         SPACE
MGPR11Ø  MSG   ØØ14,RC
*        B     MGPR12Ø
         SPACE
MGPR12Ø  MSG   ØØ11                    Display info message
         L     RØ,UIDPID               Get PATHID (first two bytes)
         GO    CSCSEV                  Sever connection
         B     MGPR1ØØ
         SPACE
MGPR2ØØ  L     R1,IPBFADR1             End of message
         ST    R1,CSCBUFFE             Save end of message address
         MVI   Ø(R1),C' '              Terminate data for MSG scanner
         GO    CSCUSC                  Process User request
         B     MGPR1ØØ
         SPACE
MGPR5ØØ  L     RØ,UIDPID               Get PATHID (first two bytes)
         GO    CSCSEV                  De-allocate and terminate
         B     MGPR1ØØ
         SPACE
MGPR6ØØ  CLI   UIDTYPE,IPTYPRNP        Send completed?
```

```
         BNE   MGPR800
         TM    UIDOPT1,UIDPEND         Was anything pending?
         BZ    MGPR100                 No, check next user
         TM    UIDOPT1,UIDCONN         Is this a connected user?
         BO    MGPR700                 Yes, do not rebuild local screen
         OI    UIDOPT4,UIDBSCR         Set Build Screen option
         GO    CSCBLD                  Build new screen
MGPR700  BAS   R14,SEND                Send it, finally
         B     MGPR100
         SPACE
MGPR800  MSG   0020                    Unexpected IUCV type
******** ***** ********************** We will fix it, if it happens
 DC H'0'                               ******************************
         B     MGPR100
         SPACE
MGPR900  L     R14,MGPRSV14
         BR    R14
         SPACE 3
*
* Add entry to UID buffer
*
*        Input R1 addresses reference UID record or zero (first record)
*              R7 addresses record to add (cache image)
*              R8 addresses UIB block
*
*
ADD      EQU   *
         SR    R0,R0
         TM    UIDOPT1,UIDFFREE        Is record on Free List?
         BO    ADD100                  Yes, remove it
         L     R3,UIDFREE1             First record in Free List
         L     R2,CCHFWD-CCHSECT(,R3)  Address second Free record
         ST    R0,CCHBWD-CCHSECT(,R2)  Clear backward pointer
         ST    R2,UIDFREE1             Store new pointer to Free List
         MVC   0(CCHSIZEB,R3),CCHSECT  Copy record
         B     ADD600
         SPACE
ADD100   NI    UIDOPT1,X'FF'-UIDFFREE  Reset option
         L     R3,CCHBWD               Address previous record
         L     R2,CCHFWD               Address following record
         LTR   R3,R3                   Is it the first Free record?
         BZ    ADD200
         ST    R2,CCHFWD-CCHSECT(,R3)  No, chain previous with next
         B     ADD300
         SPACE
ADD200   ST    R2,UIDFREE1             Yes, we have a new first Free
ADD300   LTR   R2,R2                   Is it the last Free record?
         BZ    ADD400
         ST    R3,CCHBWD-CCHSECT(,R2)  No...
         B     ADD500
         SPACE
```

```
ADD400   ST    R3,UIDFREE2             Yes...
ADD500   LR    R3,R7                   Address record to add
ADD600   LTR   R1,R1                   Add as first record?
         BNZ   ADD700
         L     R2,UIDBUFF1             First record in UID buffer
         ST    R3,UIDBUFF1             Store new first UID record
         ST    R0,CCHBWD-CCHSECT(,R3)  Clear backward pointer
         ST    R2,CCHFWD-CCHSECT(,R3)  Chain with old first
         B     ADD800
         SPACE
ADD700   ST    R1,CCHBWD-CCHSECT(,R3)  Change backward pointer
         L     R2,CCHFWD-CCHSECT(,R1)
         ST    R3,CCHFWD-CCHSECT(,R1)  Insert new record
         ST    R2,CCHFWD-CCHSECT(,R3)
ADD800   LTR   R2,R2                   Is it the last record?
         BNZ   ADD900
         ST    R3,UIDBUFF2             Yes, address new last record
         BR    R14
         SPACE
ADD900   ST    R3,CCHBWD-CCHSECT(,R2)
         BR    R14
         SPACE 3
*
* Delete entry from UID buffer
*
*        Input R7 addresses record to delete (cache image)
*              R8 addresses UIB block
*
*
DELETE   EQU   *
         SR    R0,R0
         L     R1,CCHBWD               Address previous record
         L     R2,CCHFWD               Address following record
         LTR   R1,R1                   Is it the first record?
         BNZ   DEL100
         ST    R2,UIDBUFF1             Yes, we have a new first
         B     DEL200
         SPACE
DEL100   ST    R2,CCHFWD-CCHSECT(,R1)  No, chain previous with next
DEL200   LTR   R2,R2                   Is it the last record?
         BNZ   DEL300
         ST    R1,UIDBUFF2             Yes, we have a new last
         B     DEL400
         SPACE
DEL300   ST    R1,CCHBWD-CCHSECT(,R2)  No, chain next with previous
DEL400   L     R3,UIDFREE2             Add deleted record to bottom...
         ST    R3,CCHBWD               ... of Free List
         ST    R0,CCHFWD
         ST    R7,CCHFWD-CCHSECT(,R3)
         ST    R7,UIDFREE2             Store new last Free record
         BR    R14
```

```
         SPACE 3
*
* Add TOF, EOF or blank lines to the user buffer
*
*
ADDTOFT  SR    R1,R1                  Add TOF as first record
         B     ADDTOF                 Execute TOF common code
         SPACE
ADDTOFB  L     R1,UIDBUFF2            Add TOF after last record
ADDTOF   L     R7,UIDFREE1            TOF common code
         MVC   CCHUSER,BLANKS
         MVC   CCHDATA(L'TOF),TOF     Now TOF message
         MVI   CCHRLEN,L'TOF          Move length
         B     ADDTEB                 Execute common code
         SPACE
ADDEOFT  SR    R1,R1                  Add EOF as first record
         B     ADDEOF
         SPACE
ADDEOFB  L     R1,UIDBUFF2            Add EOF after last record
ADDEOF   L     R7,UIDFREE1
         MVC   CCHUSER,BLANKS
         MVC   CCHDATA(L'EOF),EOF
         MVI   CCHRLEN,L'EOF
         B     ADDTEB
         SPACE
ADDBLKT  SR    R1,R1                  Add blank as first record
         B     ADDBLK
         SPACE
ADDBLKB  L     R1,UIDBUFF2            Add blank after last record
ADDBLK   L     R7,UIDFREE1
         XC    CCHUSER,CCHUSER        Use nulls for User (no prefix)
         XC    CCHDATA,CCHDATA
         XC    CCHRLEN,CCHRLEN
*        B     ADDTEB
         SPACE
ADDTEB   ST    R14,ADDTSV14
         XC    CCHDATE,CCHDATE        Clear date, time...
         XC    CCHTIME,CCHTIME
         XC    CCHRECNO,CCHRECNO
         MVI   CCHOPTS,X'00'          Clear options and prefix
         MVI   CCHPREF,X'00'
         OI    UIDOPT1,UIDFFREE       Remember we are stealing a line
         BAS   R14,ADD                Add line to user buffer
         BAS   R14,PREFIX             Get prefix and attributes
         L     R14,ADDTSV14
         BR    R14
         SPACE 3
*
*  Delete all messages not on Hold. Only valid if CMS scroll is On
*
*
```

```
CLEAR    EQU   *                       Clear screen
         ST    R14,CLEASV14
         L     R7,UIDBUFF2             Address bottom line
CLEA100  L     R4,CCHBWD               Address previuos line
         TM    CCHOPTS,CCHHOLD         Message on Hold?
         BO    CLEA900
         L     R1,CCHRECNO             No, check if blank
         LTR   R1,R1
         BNZ   CLEA200
         CLI   CCHDATA,X'00'
         BE    CLEA900                 It is a blank line, that's good
CLEA200  BAS   R14,DELETE              Delete non blank lines
         BAS   R14,ADDBLKB             Replace with blank at the bottom
CLEA900  LTR   R7,R4
         BNZ   CLEA100                 Check all screen lines
         L     R14,CLEASV14
         BR    R14
         SPACE 3
*
* Send UID screen to destination user
*
*        Input R8 addresses UIB block
*
*
SEND     EQU   *
         TM    UIDOPT1,UIDSEND         Is last Send still in progress?
         BZ    SEND100                 No, send new data
         OI    UIDOPT1,UIDPEND         Yes, wait, set pending option
         BR    R14
         SPACE
SEND100  ST    R14,SENDSV14
         TM    UIDOPT1,UIDCONN         Is user connected?
         BZ    SEND200
         L     R0,UIDSCRN              Yes, swap screen and alternate
         L     R1,UIDSCRNA
         ST    R0,UIDSCRNA
         ST    R1,UIDSCRN
SEND200  OI    UIDOPT1,UIDSEND         Remember SEND is in progress
         NI    UIDOPT1,X'FF'-UIDPEND   Reset any pending SEND
         L     R4,UIDSCRN              Address user screen
         L     R5,UIDSCRNL             Screen length
         LA    R9,CSCPARMU             Address User IUCV Parameter List
         MVC   IPPATHID,UIDPID         Move PATHID
         MVI   IPFLAGS1,X'00'          Clear all flags
         LA    R0,7                *T* Create trace entry
         BAS   R14,TRACE           *T*
         IUCV  SEND,PRMLIST=CSCPARMU,DATA=BUFFER,BUFFER=(R4),          *
               BUFLEN=(R5),TYPE=1WAY
         BZ    SEND900
         SR    R15,R15                 IUCV SEND error
         IC    R15,IPRCODE             Load IPRCODE
```

```
         MSG   0024,RC                Display error message
         L     R0,UIDPID              Get PATHID (first two bytes)
         GO    CSCSEV                 Sever connection
         B     SEND900                Return
         SPACE
SEND900  L     R14,SENDSV14
         BR    R14
         SPACE 3
*
* Get Prefix and Attribute fields
*
*        Input R7 addresses record to process (cache image)
*
*
PREFIX   EQU   *                      Build record prefix
         USING PFXSECT,R1
         LA    R1,PFXPTR              Prepare to scan table
PREF100  L     R1,PFXFWD              Address entry
         LTR   R1,R1                  Check for end of table
         BZ    PREF200                Done, not found
         CLC   CCHUSER,PFXUSER        Is this the entry?
         BNE   PREF100                No, check all table entries
         MVC   CCHPREF,PFXPREF        Move prefix into record
         MVC   CCHATTR,PFXATTR        Move also default attributes
         BR    R14                    Return
         SPACE
PREF200  MVI   CCHPREF,C' '           Not found
         MVI   CCHATTR,X'00'
         BR    R14
         SPACE 3
*
* Check message table
*
*        Input R7 addresses record to process (cache image)
*       Output R5 addresses MSG entry if match found, zero otherwise
*              A cc not zero is returned if no match found
*
*
MATCH    EQU   *                      Search message table
         USING MSGSECT,R5
         ST    R14,MATCSV14
         LA    R5,MSGPTR              Address message table
         LA    R15,4                  Used to generate a non zero cc
         IC    R2,CCHRLEN
         LA    R1,CSCBUFF             Use IUCV buffer as cache work
         STC   R2,CCHRLEN-CCHSECT(,R1) Store record length
         BCTR  R2,0                   Prepare to EXecute
         EX    R2,MATMVC              Move message text
         EX    R2,MATTR               Translate to uppercase
         LR    R1,R7                  Save address of real cache msg
MAT100   L     R5,MSGFWD              Address MSG table entry
```

```
         LTR   R5,R5                   Anything left?
         BZ    MAT900                  No, match not found
         CLC   MSGUSER,CSCASTER        Test MSG user for an asterisk
         BE    MAT200                  Good, that matches everything
         CLC   MSGUSER,CCHUSER         Now test originating user
         BNE   MAT100                  No good, search all MSG table
MAT200   TM    MSGOPTS,MSGCASE         NoCase specified?
         BZ    MAT300
         LA    R7,CSCBUFF              Yes, compare with uppercase msg
MAT300   BAS   R14,LOCATE              Check message against mask
         LR    R7,R1                   Restore real cache address
         BNZ   MAT100                  Not found
MAT800   SR    R15,R15                 We found a match
         MVC   CCHATTR,MSGATTR         Copy message attributes
         TM    MSGOPTS,MSGHOLD         Is message to be held?
         BZ    MAT810
         OI    CCHOPTS,CCHHOLD         Yes, set option in cache record
MAT810   TM    MSGOPTS,MSGNODSP        NoDisplay message?
         BZ    MAT900
         OI    CCHOPTS,CCHNODSP        Yes, set option in cache record
MAT900   LTR   R15,R15                 Generate cc
         L     R14,MATCSV14
         BR    R14
         SPACE
MATMVC   MVC   CCHDATA-CCHSECT(*-*,R1),CCHDATA
MATTR    TR    CCHDATA-CCHSECT(*-*,R1),CSCUPP
         SPACE 3
*
* Compare message text with mask
*
*        Input R7 addresses record to process (cache image)
*              R5 addresses MSG entry
*              A cc not zero is returned if data not found
*
*
LOCATE   EQU   *                       Compare message text with mask
         LA    R2,CCHDATA              Address message text
         LA    R3,MSGMASK              Address message mask
         SR    R0,R0                   Required by IC next
         IC    R0,CCHRLEN              Message length
         AR    R0,R2                   Address end of message
         ST    R0,MSGSUB               Initialize ARBCH save area
LOC100   CLC   0(1,R3),MSGARBCH        Test ARBCH first
         BNE   LOC200                  No good, keep trying
         LA    R3,1(,R3)               Skip ARBCH
         STM   R2,R3,MSGSUB            Save pointers
         C     R3,MSGMASKE             End of mask?
         BL    LOC100                  No, loop back
         BR    R14                     Yes, match found
         SPACE
LOC200   CR    R2,R0                   All message scanned?
```

```
         BNL   LOC5ØØ                  Yes, match not found
         CLC   Ø(1,R3),MSGANYCH        Is it ANYCH?
         BE    LOC3ØØ                  Yes, character match
         CLC   Ø(1,R3),Ø(R2)           Last chance, do characters match
         BE    LOC3ØØ
         LM    R2,R3,MSGSUB            Restore from last ARBCH
         LA    R2,1(,R2)               Advance message pointer
         ST    R2,MSGSUB               Save new value
         B     LOC2ØØ
         SPACE
LOC3ØØ   LA    R3,1(,R3)               Yes, advance both pointers
         LA    R2,1(,R2)
         CR    R2,RØ                   End of message?
         BL    LOC4ØØ
         C     R3,MSGMASKE             End of mask?
         BL    LOC1ØØ                  No, allow final ARBCH
         BR    R14                     Yes, match found
         SPACE
LOC4ØØ   C     R3,MSGMASKE             Just end of mask?
         BL    LOC1ØØ                  No, check next character
LOC5ØØ   LTR   R14,R14                 Match not found, generate cc
         BR    R14
         SPACE
         DROP  R5
         SPACE 3
*
* Check message
*
*        Input R7 addresses record to process (cache image)
*              R8 addresses UID Block
*       Output A cc zero is returned if message is selected
*
*
SELECT   EQU   *                       Select message
         TM    UIDOPT3,UIDFLTR+UIDNODSP Check NoDisplay messages?
         BZ    SEL1ØØ                  No keep going...
         NI    UIDOPT3,X'FF'-UIDNODSP  Reset option
         TM    CCHOPTS,CCHNODSP        NoDisplay message?
         BO    SEL9ØØ                  Yes, reject message
SEL1ØØ   TM    UIDOPT2,UIDEXC          Selective Exclude?
         BZ    SEL6ØØ                  No, try Include
         LA    R1,UIDSEL               Address field
         LA    R2,L'UIDSEL             Length
SEL2ØØ   CLC   CCHPREF,Ø(R1)           Check Prefix
         BE    SEL8ØØ                  We found it, exclude message
         LA    R1,1(,R1)               Advance pointer
         CLI   Ø(R1),C' '              End of selected prefixes
         BE    SEL9ØØ                  Yes, select message
         BCT   R2,SEL2ØØ
         CR    R14,R14                 Generate cc zero
         B     SEL9ØØ                  Not on table, select it
```

```
         SPACE
SEL6ØØ   TM    UIDOPT2,UIDINC          Selective Include?
         BZ    SEL9ØØ                  No, select message
         LA    R1,UIDSEL               Address field
         LA    R2,L'UIDSEL             Length
SEL7ØØ   CLC   CCHPREF,Ø(R1)           Check Prefix
         BE    SEL9ØØ                  We found it, select message
         LA    R1,1(,R1)               Advance pointer
         CLI   Ø(R1),C' '              End of selected prefixes
         BE    SEL8ØØ                  Yes, reject message
         BCT   R2,SEL7ØØ
*        B     SEL8ØØ                  Not found
         SPACE
SEL8ØØ   LTR   R14,R14                 Generate non zero cc
SEL9ØØ   BR    R14
         SPACE 3
*
* Call CMS to allocate storage
*
*    OBTAIN  Double word aligned
*    OBTAINP Page aligned
*
*
OBTAIN   EQU   *
         CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES
         B     OBT1ØØ
         SPACE
OBTAINP  EQU   *
         CMSSTOR OBTAIN,DWORDS=(Ø),MSG=YES,BNDRY=PAGE
OBT1ØØ   LR    R15,RØ                  Copy allocated double words
         A     R15,FSALLDW             Add number of double words alloc
         ST    R15,FSALLDW             Store new value
         L     R15,FSALL               Number of allocations
         A     R15,ONE                 Increment by one
         ST    R15,FSALL               Store new value
         BR    R14
         SPACE 3
*
* Call CMS to release storage
*
*
```

Editor's note: this article will be continued next month.

Fernando Duarte
Analyst (Canada)

Sterling Software's VM Division Web site

Continuing the series of VM Web site reviews, we visit Sterling Software's VM Division Web site, which can be accessed at http://www.vm.sterling.com/. If you have comments on the Web sites reviewed in this series, or suggestions for relevant sites to review, please feel free to contact the author at gabe@acm.org or Xephon at any of the addresses shown on page 2.

Sterling Software's VM Division (VMD) is one of the longest-tenured vendors in the VM market, having been formed in 1981 as VM Software and later becoming Systems Center, which was then acquired by Sterling Software. During these differing incarnations, the original orientation towards vigorous VM community citizenship has continued. The opening Web page progresses from rejuvenating mainframe applications (upper left) to VMD employment opportunities (lower right).

One of the site's major values is the general 'VM info' button at the left, which links to both Sterling-published information and off-site resources. Its introduction reads: "*This section provides you with articles, documents, and information on VM, the Internet and intranets, and systems management issues*". The first of three columns of links offers general information and introductory material. It includes links to general VM/ESA topics, Year 2000 information, and PC Server/390 descriptions, all lower on the page, and IBM's VM Web site (reviewed in *VM Update*, Issue 139, March 1998). Another link provides IBM's 'VM/ESA fact sheet', including topics such as:

- VM/ESA benefits
- Introduction: function by release
- Hardware required
- Software required
- System/390 exclusives
- VM/ESA feature comparison

- VM operating systems comparison
- Key to symbols used in tables
- End of service information.

The benefits listed – none of which will surprise VM practitioners, and all of which should be emphasized to management responsible for endorsing and funding VM – include:

- *"Enables consolidating multiple system images to one providing easier systems management and lower personnel costs.*
- *Is one of the most scalable operating systems in the industry, from single-user systems to systems supporting thousands of concurrent users.*
- *Delivers portability with a single programming interface for all VM applications.*
- *Includes rich application tools, such as REXX, CMS Pipelines, and Java, empowering people to meet their own computing needs.*
- *Supports Message Queuing Interface (MQI) client to enable applications on different systems and architecture to work together.*
- *Allows large applications to use VM data spaces and 31-bit addressing.*
- *Enables testing parallel sysplex system environment for MVS and OS/390 on single VM image as a guest of VM."*

With such current benefits, it's easy to lose track of VM evolution, and be confused about which functions are available under which VM versions – especially important for developing portable applications and planning migrations. Several sections of the fact sheet are interesting and useful, for example 'Introduction: function by release' identifies which VM/ESA release introduced major functions. This includes details of the most recent release, VM/ESA Version 2 Release 3. *Editor's note: VM/ESA Version 2 Release 3,was reviewed in* VM Update*, Issue 146, October 1998.*

The page links to added information on several of these functions.

Other sections tabulate different functional comparisons. The last section gives 'sunset' information, noting when VM versions lost or will lose formal support. A subtle but useful feature of the facts page is that section headings all link back to the table of contents.

Recognizing the importance of VM's partnership with other operating systems, Sterling's VM information page links to IBM's white paper on the value of VM for VSE enterprises, which includes the key paragraph:

"IBM introduced the Virtual Machine (VM) construct in the late 1960s to allow users to run numerous systems on one physical machine. VM can also play the role of either, or both, client and server for applications. Thus, a company could be running three (or more) VSE operating systems simultaneously. One could be their production system, a second could be used for changes to mission-critical applications, and a third for testing today's applications with its system clock set to the date of February 29, 2000! All three, running simultaneously, but separately, are isolated from others but still share resources. For example, a database running natively on VM can serve data to all three VSE images at the same time."

The next Sterling link is to their VM user groups page, which lists 11 VM-focused organizations around the world. If you're near one of these groups, they're worth investigating. If you know of groups not listed here, Sterling would like to hear about and link to them. Continuing straight down the VM information page, there's a meaty section on VM/ESA fundamentals, news, and resources. The first link here, 'VM/ESA 2.3.0 update', provides a long article written by long-time VMer and IBMer Chuck Morse. Written soon after general availability of the current VM version, it begins:

"Last month IBM made available VM/ESA 2.3.0, which provides many new and enhanced functions that strengthen VM/ESA's role in network computing, application development, and guest support. Several changes in this release are intended to reduce the overall cost of computing, not only by lowering software costs, but also by simplifying installing and maintaining VM systems and improving the efficiency of VM/ESA."

Chuck's paper is followed by a VMD PowerPoint presentation on the

Value Of VM, highlighting issues such as the cost of computing, advantages of VM, how customers use VM, and how to determine the value of VM to an organization. This presentation can provide balance and insight in making decisions relating to downsizing or changing computing environments. This is followed by links to related topics: 'Justifying renewed investment in VM', 'The value of VM for VSE enterprises', and 'Comparing application costs on different computing platforms'.

The next link provides IBM's page on DB2 for VM and VSE (formerly called SQL/DS), which opens:

"DB2 Server for VSE & VM is a powerful, full-function RDBMS that supports both production and interactive environments, improving and increasing the productivity of your company's continuous, distributed operations.

In today's fast-track world, managing vast amounts of data is just part of a day's work. And managing that data effectively makes the difference between getting by and getting ahead. DB2 Server Version 5, a key member of the DB2 family, provides all the pieces of the puzzle you need to start building your distributed database solution."

This page offers information on the current DB2 VM version, previews the next version and offers its beta test enrollment (a creative way for IBM to recruit!), highlights feature articles such as *Using locking and the new uncommitted read level*, and links to DB2-related tools. Category links include 'About', 'News', 'Events', 'Library', 'Education', 'Business partners', 'Support', 'Feedback', and 'Data management home page' – useful material for database devotees.

Other VM/ESA Sterling links include 'The byte file system' (by Don Sengpiehl, Sterling Software), 'OpenEdition VM' (by Jeff Savit, Merrill Lynch), and 'A cylinder saved is a cylinder earned... or is it' (by Robert Kusche, VM Assist). The next category on Sterling's page deals with VM's Year 2000 technology and issues, presenting a mixture of Sterling, IBM, and third-party information. The first link gives Year 2000 information on VMD products, including the techniques used for storing and displaying dates. That's followed by a link to IBM's VM Year 2000 page, which begins:

"VM/ESA Version 2 Release 2 is the 'Year 2000' release. So, if you are

not on that release yet, or need to learn more, the information provided herein can assist with the Year 2000 transition. Attention MVS, OS/390, TPF, VSE, and VM customers, do you know...” with the last phrase linking to a page beginning “*VM/ESA can support your (MVS, OS/390, TPF, VSE) production systems and simultaneously permit test systems to run with their clocks set independently to the Year 2000 and beyond*”. That’s not likely to be news to VMers, but might be of interest to non-VM colleagues. The VM Year 2000 page also highlights and explains VM’s industry association ITAA*2000 certification. A link worth following is Bob Kusche’s article *Year 2000 true stories*, which provides both amusement and motivation to address the millennium issue.

The next section of the VM information page addresses PC Server System/390 topics, noting (though it’s irritatingly blinking) that it’s possible to run VM for $20 per user, per month, with three information links followed by one to IBM’s page on this interesting equipment, which begins:

“PC Server System/390 combines the best of System/390 technology with the latest PC Server performance and cost effectiveness to provide an outstanding value for anyone needing System/390 capabilities at surprisingly low cost.

The Enhanced System/390 Microprocessor Adapter (P/390) includes a complete System/390 processor and dedicated memory to actually execute the full System/390 instruction set. This new version of the P/390 adapter provides a standard 256MB of ECC memory for the System/390 system, a System/390 processor which is 40% faster than the prior P/390 adapter, and at least twice the I/O throughput capacity of the prior adapter.”

The bottom resource linked is an article on implementing the CMS Shared File System, which begins “*Implementing the Shared File System is more than a way to free up a tremendous amount of DASD space. It’s also a way to increase satisfaction of your users”*. That’s hard to argue with, since many sites haven’t yet fully engaged or exploited SFS.

The VM information page tour continues with the centre and right-hand columns at the top: ‘Web-related topics’ and ‘Systems

management'. The first Web topic, 'Informative articles' is, in fact, a rich source of general Web and mainframe Web information, listing well over a dozen articles on the virtues of big-iron Web hosting, written by VM authorities such as Jeff Savit and Melinda Varian (whose Web page was reviewed in *VM Update*, Issue 141, May 1998). Topics such as 'Big-iron gets a case of Web fever' and 'The Internet, System/390, and serious e-Business' can reassure organizations wondering about positioning mainframes on the Internet.

Below the VM and mainframe information are three categories, 'Internet/Intranet info', 'HTML and CGI script info', and 'Search engines'. The first category includes entries from 'Beginner's Web glossary' and 'Intranets: readings and resources' to a list (maintained by Larry Dusold at the US FDA) of sites running VM-based Web servers, including Lafayette Life Insurance, many universities around the world, and a Canadian wildlife organization. The second category includes HTML resources and a topic of interest to VMers: writing Web CGI scripts in REXX. This link, to a paper written by Stanford Linear Accelerator Center's R L A Cottrell, describes several functions in a publicly available library of REXX functions that simplify writing CGI scripts. Topics include:

- Getting input to the script:
  - QUERY_STRING environment variable
  - *PATH_INFO environment variable*
  - Command line
  - Standard input
- Decoding forms input
- Sending document back to client
- Diagnostics and reporting errors
- Two simple REXX CGI scripts
- Security concerns:
  - Beware the INTERPRET, POPEN, or ADDRESS UNIX
  - Escaping dangerous characters

  - Restrict access to files
  - Restricting distribution of information
  - Test the script
- Further information
- Code referenced in presentation.

Back on Sterling's main page, several links are worth exploring. A management brief, entitled *Revitalize your mainframe applications,* explains why VM systems remain a strategic and flexible computing platform and investment, beginning:

"Your users have a love-hate relationship with your mainframe applications. They love the functionality, reliability, and proven performance that legacy systems deliver, but they hate the antiquated 'green screen' interfaces built for 3270 terminals.

Today's users rely on desktop and laptop systems, and they demand Graphical User Interfaces (GUIs). Users need access to information not just from the office, but from home and the road as well. Many users like the World Wide Web because it provides nearly universal access to a wealth of information and delivers it through an intuitive GUI.

So, how can you give your users access to mainframe applications from anywhere, and GUIs, without the high costs and risks of replacing your proven mainframe applications? The answer is to provide Web browser access to your mainframe applications (Web-enable), and more importantly exploit the browser capabilities to improve the applications' user interface (Web-enhance). Adding a graphical, Web browser front-end on your mainframe applications can extend the life of your applications, make your users happy, and keep your costs down.

But how do you provide Web browser GUIs for mainframe applications and help your company leverage its investment in mainframe applications and infrastructure? This management brief discusses the choices you have to bring your mainframe data and applications to the Web."

Success stories describe VM use at Columbia Energy, Lafayette Life Insurance, Trans World Airline, and WVNET – illustrating solid business reasons for VM use. A link to Anura Guruge's Newsletter offers perspectives from someone who:

"...is an independent strategic technical consultant, author, and raconteur who specializes in all aspects of contemporary IBM-related networking. He founded the 'SNA-Capable i-nets' forum (www.sna-inets.com), and is writing a book for Addison Wesley Longman on Integrating i-nets and data centers. *He authored the best selling* SNA: theory and practice *(1984), as well as* Reengineering IBM networks. *He has published over 250 articles."*

These newsletters, in Adobe's PDF format, are not easy to download, but are interesting and well-illustrated, exploiting the portable document technology. The July newsletter begins:

"Anu wraps up our in-depth look at 3270-to-HTML with a case study and a 'horses for courses' table; gives you the scoop on IBM's 8270-600, 8277, and yes the 2216; looks at 8277, Bluestone; latest happenings on tn3270(E) front, and applet 3270/5250 emulator comparisons...

$9.5B Lafayette Life slashes mainframe access costs for its 1,000 agents using Internet access with Sterling's VM:Webgateway 3270-to-HTML conversion.

The bewildering IBM 8270-600 has to be IBM's sad swansong to the future of Token-Ring.

The IBM/Xylan 8277 bolsters IBM's growing stable of enticing Fast Ethernet offerings.

OpenConnect partners with Bluestone Software to complement 'terminal' access with programmatic access.

The latest tn3270(E) proposal addresses 'server-client' contention scenarios."

A link in the main page's right column provides International Technology Group's *Enterprise solutions for Web server* management brief, with yet more information on technology linking enterprise servers and the Web:

"As of year end 1997, more than 1,000 organizations worldwide had implemented Web servers on mainframes. On current trends, by year end 1998 the number will exceed 2,300 and by the end of the decade more than 10,000 will be operational.

This is a fundamentally logical trend. In most organizations, the majority of business-critical data is located on mainframe systems. More than 75% of internal data accessed by corporate PC users, and more than 60% of all data available via the Web, originates in mainframe databases.

The large-scale, business-critical transaction-processing systems which will play a central role in the future evolution of electronic commerce are also predominantly mainframe-based. During 1997 more than 83% of all commercial transactions worldwide were processed by mainframe systems. In some industries, such as financial services, insurance, and transportation, the figure exceeded 95%."

A final link to highlight among the main page left-hand buttons is 'Events'; in addition to Sterling notices, this links to IBM's VM events Web page, describing IBM and other VM-related events planned through until 2001 around the world. At the time of writing, the first three events listed are in Hong Kong, Seoul, and Omaha. Links are provided to major groups such as SHARE, WAVV, TPF Users Group, and IPSO (French Club VM).

Sterling VMD does a good job of balancing information on its own VM products with material of interest to all VMers, whether VMD customers or not. The site, "*powered by System/390 and Sterling's VM:Webgateway*", also has added value customer-only links, and is worth browsing for both technical tips and VM/ESA general-information.

Gabe Goldberg
Computers and Publishing (USA)

VM news

VM:Webgateway is now available from IBM. It provides Web browser interfaces for all VM, VSE, OS/390, and TPF applications.

VM:Webgateway serves all types of Web data, protects data by combining mainframe-class authentication and access control with Secure Sockets Layer (SSL) technology, simplifies administration tasks with a Web browser interface, and scales to meet the needs of users.

It comes with the VM:Webgateway OfficeVision interface, which provides a graphical browser interface for OfficeVision's e-mail and calendar functions.

These products come from Sterling Software's VM Software Division and are available through IBM's Software Vendor Marketing Programme.

For further information contact your local IBM representative.

* * *

VM users can benefit from euroTUBES, Macro 4's euro display conversion utility. The currency conversion utility is designed to assist companies during the euro transition period.

EuroTUBES interfaces with any on-line application that runs under TUBES and enables the display and input of financial information in the euro without the need to modify existing applications. In addition to VM/ESA, euroTUBES is available for OS/390, VSE/ESA, and MVS/ESA.

For further information contact:
Macro 4, The Orangery, Turners Hill Road, Worth, Crawley, W Sussex, RH10 4SS, UK.
Tel: (01293) 886060.
Macro 4, 35 Waterview Blvd, PO Box 292, Parsippany, NJ 07054-0292, USA.
Tel: (201) 402 8000.
URL:http://www.macro4.com.

* * *

IBM has announced its new System/390 Integrated Server,which supports current levels of VM/ESA, OS/390, MVS/ESA, and VSE/ESA, and provides processor performance comparable to the 9221 Model 170 for most workloads.

The new unit comprises 256MB memory, internal SSA RAID-5 DASD, plus ESCON and parallel adapters, and an integrated battery back-up unit. The base unit will have 36GB DASD, expandable to 255GB within the box. There will be scope to use PCI I/O slots and ISA slots for ESCON and System/390 parallel channels, and standard connections to LANs, WANs, and other peripheral devices.

The machine is positioned as a packaged application server for System/390 software, and for development using old and new tools.

For further information contact your local IBM representative.