154 VM

June 1999

update

In this issue

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 £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £16.00 ($23.00) 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.

A quick monitor for virtual machines

The following EXEC is a quick way to find out whether a virtual machine is eating CPU or doing I/O. If you have no other monitor available, this program gives you a quick idea of what is going on.

It is based on the QUERY NAMES command, followed by an IND USER for each logged-on machine (class A or E authority will be needed). After a delay time (I used 10 seconds, but it can be adjusted for better results, depending on the installation), the program issues a second round of IND USER queries and compares the results with the first. In this way, the difference between values will tell you whether a machine is actually consuming CPU or doing I/Os.

I show the results of only those machines whose difference values are not both zero, but you can change this behaviour by giving a value other than zero to the 'showidle' variable at the beginning of the program.

The output display will look like this:

```
Ready; T=Ø.Ø5/Ø.Ø9 1Ø:41:53
seecpu
WAIT 1Ø SECONDS

Maq.        CPU Total    Dif.     IO Total    Dif.
VSE1          423:99      8       755123      ØØ65
VTAM          643:1Ø      2       998213      ØØ31
VSE2          233:32      3       Ø51253      ØØ21
RSCS          112:Ø6      1       ØØ6323      ØØØØ
USERØ8        Ø12:22      2       ØØ1434      ØØ24
USER15        Ø56:93      5       ØØ7581      ØØ87

Ready; T=Ø.15/Ø.19 1Ø:42:Ø5
```

SEECPU SOURCE CODE

```
/*================================================================*/
/*     SEECPU - Displays CPU and I/O of virtual machines          */
/*================================================================*/

interval = 1Ø
showidle = Ø
desbuf
```

```
conwait
"execio * CP ( ST QUERY NAMES"
nmaq = Ø
do i = 1 to queued()
   pull reg
   if left(word(reg,1),3) = "VSM" then iterate
   reg = translate(reg,"",",")
   reg = translate(reg,"","-")
   do k = 1 to words(reg) by 2
      nmaq = nmaq + 1
      maquina.nmaq = word(reg,k)
   end
end

do k=1 to nmaq
   desbuf
   conwait
   "execio * CP ( st IND USER " maquina.k
   if rc <> Ø then iterate
    do 5
       pull
    end
    pull linha
    dropbuf
    parse var linha . "TTIME=" valor1.k "IO=" ios1.k
end
SAY "WAIT "interval" SECONDS"
say
"CP SLEEP "interval" SEC"

do k = 1 to nmaq
   desbuf
   conwait
   "execio * CP ( ST IND USER " maquina.k
   if rc<>Ø then iterate
    do 5
       pull
    end
    pull linha
    dropbuf
    parse var linha . "TTIME=" valor2.k "IO=" ios2.k
end

say "Maq.  Total CPU    Diff.   Total IO    Diff."
say

do k = 1 to nmaq
   val1.k = space(translate(valor1.k,"",":"),Ø)
   val2.k = space(translate(valor2.k,"",":"),Ø)
   if datatype(val1.k,"W") &,
```

```
      datatype(val2.k,"W") then val.k = val2.k-val1.k
   else val.k = Ø
   if datatype(ios1.k,"W") &,
      datatype(ios2.k,"w") then ios.k = ios2.k-ios1.k
   else ios.k = "ØØØØØ"
   if val.k = Ø & ios.k = Ø & showidle = Ø then iterate
   say left(maquina.k,9) left(valor1.k,8) left(val.k,1Ø),
       right(ios1.k,6,"Ø") " " right(ios.k,4,"Ø")
end
```

Luis Paulo Figueiredo Sousa Ribeiro
Systems Engineer
Edinfor (Portugal)

VM:Secure enhancement rules – part 3

This month we continue the article providing special macros that enhance VM:Secure rules to allow additional resource access control.

OBJADD VMSECURE

```
/* Add an object file for a user */
/* NW */

'TRANSFER OUTPUT  SYSID  USERID'
Pull output  sysid  user
Call Trace output
'TEST PROCESS AUTHORIZ $OBJADD ANYUSR'
If rc ¬= Ø Then Exit -1
/*********************************************************/
/* Common routine to load the OBJECT settings.           */
/* Variables set:  objcuu         virt dev of object disk */
/*                 objmode        file mode of disk       */
/*                 objdefault     ACCEPT|REJECT default   */
/*********************************************************/
'TEST CMS PIPE (name OBJCLOAD)',
  '< OBJECT SETTINGS |',
  'VAR OBJSET'
If Symbol('OBJSET') ¬= 'BAD' Then Interpret objset
If Symbol('OBJDEFAULT') = 'BAD' Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*********************************************************/
/* Common routine to check the availablity of OBJECT RULES.*/
/*********************************************************/
```

```
'TEST CMS STATE OBJECTS LOCKED' objmode
If rc = Ø Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*************************************************************/
Arg userid uft ufm . '(' replopt .
If userid = '' Then Do
  'TEST FORMAT EMSG Ø38E'
  Exit 2
  End
If uft = '' Then uft = 'OBJECTS'
If ufm = '' Then ufm = 'A'
If replopt ¬= '' & ¬Abbrev('REPLACE',replopt,1) Then Do
  'TEST FORMAT EMSG Ø39E' replopt
  Exit 4
  End
replace = Abbrev('REPLACE',replopt,1)
userfile = userid uft ufm
userobj = userid 'OBJECTS' objmode
lockname = objmode 'OBJECTS' userid
workfile = userid 'CMSUT1' objmode
'TEST CMS STATE' userobj
If rc = Ø & ¬replace Then Do
  'TEST FORMAT EMSG 8Ø21E' userid
  Exit 1Ø
  End
'TEST PROCESS AUTHORIZ $OBJADD' userid
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 265E OBJADD' userid
  Exit 12
  End
'TEST LOCK COND PRIVATE DISK' lockname
If rc ¬= Ø Then Do
  'FORMAT EMSG 364E' userobj
  Exit 14
  End
'TEST USER EXECUTE STATE' userfile
If rc ¬= Ø Then Do
  'FORMAT EMSG Ø21E' Translate(userfile,'ØØ'x,' ')
  'LOCK CLEAR DISK' lockname
  Exit 28
  End
'TEST USER COPYFROM' userfile workfile
If rc ¬= Ø Then Do
  'TEST CMS ERASE' workfile
  'LOCK CLEAR DISK' lockname
  Exit 1ØØ3
  End
'TEST EXEC OBJLOAD' userid
loadrc = rc
If rc = Ø Then Do
```

```
  'TEST CMS ERASE' userobj
  'TEST CMS RENAME' workfile userobj
  End
Else Call NoChange
'LOCK CLEAR DISK' lockname
Exit loadrc
NOCHANGE:
'TEST CMS ERASE' workfile
'FORMAT EMSG 621E' loadrc 'OBJLOAD'
loadrc = 3Ø
Return
```

OBJCHK VMSECURE

```
/* Check the access allowed for a particular user and OBJECT */
/* NW */

'TRANSFER OUTPUT  SYSID  USERID  AUDT'
Pull output  sysid  user  audt
Call Trace output
Call Time 'R'
'TEST PROCESS AUTHORIZ $OBJCHK' user
If rc ¬= Ø Then Exit -1
/*************************************************************/
/* Common routine to load the OBJECT settings.               */
/* Variables set:  objcuu         virt dev of object disk */
/*                 objmode        file mode of disk       */
/*                 objdefault     ACCEPT|REJECT default   */
/*************************************************************/
'TEST CMS PIPE (name OBJCLOAD)',
  '< OBJECT SETTINGS |',
  'VAR OBJSET'
If Symbol('OBJSET') ¬= 'BAD' Then Interpret objset
If Symbol('OBJDEFAULT') = 'BAD' Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*************************************************************/
/* Common routine to check the availablity of OBJECT RULES.*/
/*************************************************************/
'TEST CMS STATE OBJECTS LOCKED' objmode
If rc = Ø Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*************************************************************/
Arg objname object_tokens '(' quietopt .
If objname = '' Then Do
  'TEST FORMAT EMSG 8ØØ6E'
  Exit 6
  End
```

```
object_tokens = Space(object_tokens)
quiet = Abbrev('QUIET',quietopt,1)
'TEST CMS STATE' objname 'OBJDEF' objmode
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 82ØØE' objname
  Exit 28
  End
If object_tokens = '',
   | Pos('*',object_tokens) > Ø ,
   | Pos('%',object_tokens) > Ø Then Do
  'TEST FORMAT EMSG 82Ø1E' objname
  Exit 2
  End
'TEST CMS PIPE <' objname 'RULEDEF | VAR OBJDEF'
If Symbol('OBJDEF') ¬= 'BAD' Then Interpret objdef
Else Do
  'TEST FORMAT EMSG 82Ø2E' rc objname 'RULEDEF'
  Exit 3ØØ
  End
If tokens.objname ¬= Words(object_tokens) Then Do
  'TEST FORMAT EMSG 82Ø6E' objname tokens.objname
  Exit 4
  End
If default_action.objname ¬= '' Then
  objdefault = default_action.objname
select = objname||'FF'x||Left(object_tokens,1)
findwild = objname||'FF'x||'*'
lookfor = Translate(objname object_tokens,'FF'x,' ')
Parse Value 'n/a n/a n/a n/a' With syskey usrkey sysmatch usrmatch,
                    access_allowed universal_found
'TEST CMS STATE SYSTEM OBJECTS' objmode
If rc = Ø Then Do
  'TEST CMS PIPE (ENDCHAR ?)|',
      '< SYSTEM USEROBJ |',
      'DROP 1 |',
      'A: FIND' select'|',
      'STEM SEARCH. |',
      'FIND' lookfor'_|',
      'VAR FOUND',
      '? A: |',
      'FIND' findwild'|',
      'VAR WILD'
  If found ¬= 'FOUND' Then Do
    access_allowed = Word(found,Words(found))
    universal_found = 'EXACT'
    End
  Else Do
    If wild = 'WILD' Then wild = ''
    If search.Ø > Ø | wild ¬= '' Then Do
      Parse Value FEntry() With syskey sysaccess sysmatch
      If syskey ¬= 'NOMATCH' Then Do
        universal_found = syskey
```

```
        access_allowed = sysaccess
        End
      End
    End
  End
'TEST CMS STATE' user 'OBJECTS' objmode
If rc = Ø Then Do
  'TEST CMS PIPE (ENDCHAR ?)|',
      '<' user 'USEROBJ |',
      'DROP 1 |',
      'A: FIND' select'|',
      'STEM SEARCH. |',
      'FIND' lookfor'_|',
      'VAR FOUND',
      '? A: |',
      'FIND' findwild'|',
      'VAR WILD'
  If found ¬= 'FOUND' Then
    access_allowed = Word(found,Words(found))
  Else Do
    If universal_found ¬= 'EXACT' Then Do
      If wild = 'WILD' Then wild = ''
      If search.Ø > Ø | wild ¬= '' Then Do
        Parse Value FEntry() With usrkey usraccess usrmatch
        If usrkey ¬= 'NOMATCH' Then
          If (universal_found usrkey = 'PATTERN PATTERN') |,
             (universal_found usrkey = 'WILDCARD WILDCARD' &,
              Length(usrmatch) >= Length(sysmatch)) Then
            access_allowed = usraccess
        End
      End
    End
  End
If access_allowed = '' Then access_allowed = objdefault
If access_allowed = 'ACCEPT' Then erc = Ø
Else Do
  If ¬quiet Then 'TEST FORMAT EMSG 9ØØ1E' object_tokens
  erc = 298
  End
output = Date('S') Time(),
         Left(user,8) Left(access_allowed,8),
         Left(objname,8) object_tokens
'TEST CMS EXECIO 1 DISKW OBJECTS AUDIT' audt '(VAR OUTPUT'
Exit erc
/*********************************************************/
/*  Be sure to copy this code to OBJFOR !!!              */
/*********************************************************/
FENTRY: Procedure Expose objname object_tokens search. wild
If wild ¬= '' Then pipestream = 'VAR WILD | STEM SEARCH. |'
Else pipestream = 'STEM SEARCH. |'
'TEST CMS PIPE(endchar ? name FENTRY)|',
```

```
   pipestream,
   'A: LOCATE 1-* /%/|',
   'B: FANIN |',
   'CHANGE 1-* /'||'FF'x||'/ /|',
   'SPECS W 2-* 1 |',
   'STEM SEARCH.',
'? A: |',
   'LOCATE 1-* /*/|',
   'SORT DESCENDING|',
   'B:'
If search.Ø = Ø Then Return 'NOMATCH'
tokenwords = Words(object_tokens)
matched_on = 'WILDCARD'
matchtok = ''
Do i = 1 to search.Ø
  match = 1
  Do t = 1 to tokenwords
    token = Word(search.i,t)
    searchtoken = Word(object_tokens,t)
    tokenlen = Length(searchtoken)
    wildcard = Pos('*',token)
    pattern = Pos('%',token)
    If pattern wildcard = 'Ø Ø' Then minchk = Length(token)
    Else If WordPos('Ø',pattern wildcard) > Ø Then
      minchk = Max(pattern,wildcard)-1
    Else minchk = Min(pattern,wildcard)-1
    If Left(token,minchk) ¬== Left(searchtoken,minchk) Then Do
      match = Ø
      Leave t
      End
    Select
      When pattern > Ø & Length(token) ¬= tokenlen &,
           wildcard = Ø Then Do
        match = Ø
        Leave t
        End
      When pattern > Ø Then Do
        matched_on = 'PATTERN'
        Do While pattern > Ø
          searchtoken = Overlay('%',searchtoken,pattern)
          pattern = Pos('%',token,pattern+1)
          End
        If wildcard = Ø & searchtoken ¬= token Then Do
          match = Ø
          Leave t
          End
        If wildcard > Ø & ¬Check_WildCard(token,searchtoken) Then Do
          match = Ø
          Leave t
          End
        matchtok = matchtok token
```

```
        End
      When wildcard > Ø Then Do
        matched_on = 'WILDCARD'
        If ¬Check_WildCard(token,searchtoken) Then Do
          match = Ø
          Leave t
          End
        matchtok = matchtok token
        End
      Otherwise If token ¬= searchtoken Then Do
          match = Ø
          Leave t
          End
        Else Do
        matchtok = matchtok token
        End
      End
    End
  If match Then Do
    Return matched_on Word(search.i,Words(search.i)) Strip(matchtok)
    End
  End
Return 'NOMATCH'
/************************/
CHECK_WILDCARD: Procedure
Arg token , searchtoken
wildcard = Pos('*',token)
If wildcard = Length(token) Then Do
  wildcard = wildcard - 1
  If Left(searchtoken,wildcard) == Left(token,wildcard) Then Return 1
  Return Ø
  End
Else Do While Pos('*',token) > Ø
  Parse Value token With firstpart '*' . '.' token
  len = Length(firstpart)
  Parse Value searchtoken With srchfirst +(len) . '.' searchtoken
  If firstpart = '' Then Return 1      /* For "xxx*.*" entries */
  If firstpart ¬== srchfirst Then Return Ø
  End
If token ¬= '' & token ¬== searchtoken Then Return Ø
Return 1
```

OBJDEL VMSECURE

```
/* Unload (erase) USER OBJECT files */
/* NW */
'TRANSFER OUTPUT  SYSID  USERID'
Pull output  sysid  user
Call Trace output
'TEST PROCESS AUTHORIZ $OBJDEL ANYUSR'
```

```
If rc ¬= Ø Then Exit -1
/*************************************************************/
/* Common routine to load the OBJECT settings.               */
/* Variables set:  objcuu         virt dev of object disk */
/*                 objmode        file mode of disk       */
/*                 objdefault     ACCEPT|REJECT default   */
/*************************************************************/
'TEST CMS PIPE (name OBJCLOAD)',
  '< OBJECT SETTINGS |',
  'VAR OBJSET'
If Symbol('OBJSET') ¬= 'BAD' Then Interpret objset
If Symbol('OBJDEFAULT') = 'BAD' Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*************************************************************/
Arg who . '(' promptopt .
If who = '' Then Do
  'TEST FORMAT EMSG Ø38E'
  Exit 2
  End
prompt = ¬Abbrev('NOPROMPT',promptopt,3)
userobj = who 'OBJECTS' objmode
'TEST CMS STATE' userobj
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 8ØØ3E User OBJECT' who
  Exit 28
  End
'TEST PROCESS AUTHORIZ $OBJDEL' who
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 265E OBJDEL' who
  Exit 1Ø
  End
If prompt Then Do Forever
  'FORMAT EMSG 4ØØI' who
  'TEST FORMAT PROMPT 4Ø4R'
  If rc ¬= Ø Then Do
    'FORMAT EMSG Ø99I OBJDEL'
    Exit 1ØØ
    End
  Pull ans .
  If ans = 'YES' Then Leave
  Else If ans = 'NO' Then Exit Ø
  Else 'FORMAT EMSG 431E' ans
  End
'TEST CMS EXECDROP' Word(userobj,1) 'USEROBJ'
'TEST CMS ERASE' userobj
'TEST FORMAT EMSG 8ØØ2I User Objects removed' who
Exit Ø
```

OBJDLOAD VMSECURE

```
/* Load USER OBJECT files */
/* NW */

'TRANSFER OUTPUT  SYSID  USERID'
Pull output  sysid  user
Call Trace output
'TEST PROCESS AUTHORIZ $OBJLOAD ANYUSR'
If rc ¬= Ø Then Exit -1
/*********************************************************/
/* Common routine to load the OBJECT settings.           */
/* Variables set:  objcuu         virt dev of object disk */
/*                 objmode        file mode of disk       */
/*                 objdefault     ACCEPT|REJECT default   */
/*********************************************************/
'TEST CMS PIPE (name OBJCLOAD)',
  '< OBJECT SETTINGS |',
  'VAR OBJSET'
If Symbol('OBJSET') ¬= 'BAD' Then Interpret objset
If Symbol('OBJDEFAULT') = 'BAD' Then Do
  'TEST FORMAT EMSG 7ØØØE'
  Exit 299
  End
/*********************************************************/
objdefloaded. = Ø
default. = ''
/*********************************************************/
Arg loadwho . '(' loadopt .
'TEST PROCESS AUTHORIZ $OBJLOAD' loadwho
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 265E OBJLOAD' loadwho
  Exit 11
  End
If loadwho = '*' Then Do
  If user ¬= sysid Then Exit -1  /* Only SVM allowed */
  loadwho = '*ALL*'
  'TEST CMS PIPE(name LOADOBJ)|',
     'COMMAND LISTFILE * OBJECTS' objmode '|',
     'STEM FILE.'
  ten_percent = file.Ø%1Ø
  tell_at = Format(ten_percent,,Ø)
  told = 1
 'TEST CMS EXECDROP * USEROBJ'
  Do i = 1 to file.Ø
    If i = tell_at Then Do
      prct = tell_at/ten_percent*1Ø
      If prct > 1ØØ Then prct = 1ØØ
      'TEST FORMAT EMSG 8ØØ1I' prct file.Ø
      told = told + 1
      tell_at = Format(ten_percent*told,,Ø)
```

```
      If (tell_at/ten_percent*1Ø = 1ØØ & i ¬= file.Ø) |,
         tell_at > file.Ø Then tell_at = file.Ø
      End
    Call Build_Object_Load file.i
    erc = rc
    If erc ¬= Ø Then Do
      'TEST FORMAT EMSG 8Ø5E' erc file.i
      Exit erc
      End
    If i//1Ø = Ø Then 'TEST YIELD'
    End
  End
Else Do
  userobj = loadwho 'CMSUT1' objmode
  'TEST CMS STATE' userobj
  If rc ¬= Ø Then Do
    'TEST FORMAT EMSG 8Ø3E User OBJECT' loadwho
    Exit 28
    End
  Call Build_Object_Load userobj
  erc = rc
  If erc ¬= Ø Then Do
    'TEST FORMAT EMSG 8Ø5E' erc userobj
    Exit 3Ø5
    End
  End
'TEST FORMAT EMSG 8Ø2I User Objects loaded' loadwho
Exit
/*************************************************************/
Build_Object_Load:
Arg fn ft fm .
'TEST CMS PIPE(ENDCHAR ? )|',
   '<' fn ft fm '|',
   'STRIP BOTH |',
   'SPECS RECNO 1 1-* NW |',
   'NLOCATE 12.1 /*/ |',
   'STEM REC.'
Do r = 1 to rec.Ø
  rec.r = Space(rec.r)
  Parse Value rec.r With recnum acc_rej objname object_tokens
  If WordPos(acc_rej,'ACCEPT REJECT') = Ø Then Do
    'TEST FORMAT EMSG Ø39E' acc_rej
    Call PROCESS_ERROR 24
    End
  If loadopt ¬= 'FAST' Then Do
    If ¬objdefloaded.objname Then Call Load_Object_Def
    Call Validate_Object
    End
  rec.r = acc_rej objname object_tokens
  End
```

```
fm = Left(fm,1)'3'
'TEST CMS PIPE(ENDCHAR ? )|',
   'LITERAL /**/ |',
   'APPEND STEM REC. |',
   'CHANGE 8-* / /'||'FF'x||'/|',
   'SPECS W 2 1 W 1 NW |',
   '>' fn 'LOAD' fm
If loadwho ¬= '*ALL*' Then
  'TEST CMS EXECDROP' fn 'USEROBJ'
'TEST CMS EXECLOAD' fn 'LOAD' fm fn 'USEROBJ'
erc = rc
If erc ¬= Ø Then Do
  'TEST FORMAT EMSG 8ØØ5E' erc fn 'LOAD' fm
  erc = 3Ø5
  End
Return erc
/*****************************************************************/
Load_Object_Def:
'TEST CMS STATE' objname 'OBJDEF' objmode
If rc ¬= Ø Then Do
  'TEST FORMAT EMSG 82ØØE' objname
  Call PROCESS_ERROR 24
  End
'TEST CMS PIPE <' objname 'RULEDEF | VAR OBJDEF'
If Symbol('OBJDEF') ¬= 'BAD' Then Interpret objdef
Else Do
  'TEST FORMAT EMSG 82Ø2E' rc objname 'RULEDEF'
  Call PROCESS_ERROR 299
  End
objdefloaded.objname = 1
Return Ø
/*****************************************************************/
Validate_Object:
If object_tokens = '' Then Do
  'TEST FORMAT EMSG 82Ø1E' objname
  Call PROCESS_ERROR 24
  End
numtokens = Words(object_tokens)
If numtokens < tokens.objname Then Do
  Do t = numtokens+1 to tokens.objname
    If default.t.objname ¬= '' Then
      object_tokens = object_tokens default.t.objname
    Else Do
      'TEST FORMAT EMSG 82Ø4E' t objname
      Call PROCESS_ERROR 24
      End
    End
  End
Else If numtokens > tokens.objname Then Do
  'TEST FORMAT EMSG 82Ø3E' objname tokens.objname
  Call PROCESS_ERROR 24
```

```
  End
Do t = 1 to tokens.objname
  check = Word(object_tokens,t)
  length = Length(check)
  If check ¬= '*' Then Do
    If length > tokenmax.t.objname Then Do
      'TEST FORMAT EMSG 8Ø19E word' t ,
            'more max' tokenmax.t.objname
      Call PROCESS_ERROR 24
      End
    If length < tokenmin.t.objname Then Do
      'TEST FORMAT EMSG 8Ø19E word' t ,
            'less min' tokenmin.t.objname
      Call PROCESS_ERROR 24
      End
    tokenlist = Translate(token.t.objname,' ','|')
    If token.t.objname ¬= '' &,
       WordPos(check,tokenlist) = Ø Then Do
      'TEST FORMAT EMSG 8Ø2ØE word' t
      'TEST FORMAT EMSG 8Ø22I' tokenlist
      Call PROCESS_ERROR 24
      End
    End
  End
Return Ø
/*********************************************************/
PROCESS_ERROR:
Arg erc .
'TEST FORMAT EMSG Ø56I',
   recnum Translate(fn ft fm,'ØØ'x,' ')
Exit erc
```

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

James S Vincent
Software Specialist
Nationwide Insurance (USA)

Why not share your expertise and earn money at the same time? *VM Update* is looking for REXX EXECs, macros, program code, etc, that experienced VMers have written to make their life, or the lives of their users, easier. Articles can be of any length and can be sent or e-mailed to Robert Burgess at any of the addresses shown on page 2. Why not call now for a free copy of our *Notes for contributors*?

Mouse-clickable file development aids

Continuing the Mouse on the mainframe *series of articles on the manipulation of System/390 applications with a PC or workstation mouse, the author examines the creation of mouse-clickable file development aids.*

INTRODUCTION

Program and file development tools contribute to programmer productivity, in part, by simplifying routine file management functions. Programmers learn to use a tool, enjoy its benefits, and live with its drawbacks.

CMS programmers issue a variety of native file management commands or collect those commands into personal EXECs to avoid a lot of repetitive typing. Programmers may share these little utilities or write their own, more personalized versions. And of course, over time all these EXECs may require maintenance and suffer from the lack of centralized support and version control.

This article describes a strategy for building mouse-clickable CMS file development aids that are highly customizable, easily shared, and simple to maintain.

The strategy presented here for creating file development aids combines a number of XEDIT-based techniques that have been discussed in previous articles in this series. The rationale and REXX code has been given for the following functions:

- Using XEDIT reserved lines to display PF key help text.
- Using the 'HOTKEYS XEDIT' macro to enable reserved line help text to respond to workstation 'mouse clicks'.
- Using the 'KEYWIN XEDIT' macro to provide customizable pop-up menus of commands and subcommands in CMS windows.
- Assigning alternative functions (including invocations of KEYWIN XEDIT) to PF keys with alternative XEDIT profiles.

- Using the ‘PETPROF XEDIT’ macro to automatically select alternative XEDIT profiles based on filetype.

Therefore, developers should have access to the following macros and files:

- HOTKEYS XEDIT
- KEYWIN XEDIT
- PETPROF XEDIT
- DEFAULT PETPROF.

BASIC FILE DEVELOPMENT

Typically (and the native CMS command structure promotes this view), file development consists of several steps:

1 Open a file with XEDIT.

2 Add or modify text.

3 Save changes.

4 Quit XEDIT.

5 Process the file (compile, execute, print, sendfile, etc).

6 Repeat steps 1-5 to correct errors or to make further changes.

In the discussion that follows, it is assumed that all the commands necessary to process a file can be collected into pop-up menus which can be then be displayed ‘within XEDIT’. Therefore, programmers and documenters must slightly change how they work with files. Files are now developed in the following manner:

1 Open a file with XEDIT.

2 Add or modify text.

3 Save changes.

4 Select file processing commands from pop-up menus.

5 Repeat steps 2-4 to correct errors or to make further changes.

6 Quit XEDIT.

While the file development aids described below can be used with any 3270 terminal or terminal emulation software, productivity improves dramatically if (appropriately configured) mouse clicks are used to select PF keys and commands/subcommands from the pop-up menus. The rationale and coding techniques for creating 'Pointer Enabled Tools' (PETs) have been described in detail in earlier articles (*VM Update*, Issue 146, October 1998, and *VM Update*, Issue 150, February 1999).

In the sections that follow, development aids are presented for REXX programs and Script (DCF) documents. Programmers should be able to extend these techniques for any programming language (eg C, COBOL, FORTRAN) or document markup (eg HTML, TeX) as well as any other filetype.

AID FOR DEVELOPING REXX PROGRAMS

In addition to the KEYWIN and HOTKEYS macros, four files are required to implement a simple REXX program development aid:

- REXXHELP KEYWIN
- REXXAID KEYWIN
- XCMDS KEYWIN
- REXXAID XEDIT.

'REXXHELP KEYWIN' defines a menu of Help calls which are most relevant to creating EXECs and macros. This menu is assigned to XEDIT PF key 1. Programmers can alter REXXHELP KEYWIN quite easily, if desired. The REXXHELP KEYWIN menu contains the following lines:

```
                          'REXX Help
cms  help    task         '* HELP
help border  menu         '* BORDER
help cms     menu         '* CMS
help cp      menu         '* CP
help pipe    menu         '* PIPE
help rexx    menu         '* REXX
help vscreen menu         '* VSCREEN
help window  menu         '* WINDOW
help XEDIT   menu         '* XEDIT
```

'REXXAID KEYWIN' defines a menu of XEDIT and CMS commands that are relevant to managing EXECs and macros. This menu is assigned to XEDIT PF key 6. Programmers can add other XEDIT subcommands, macro calls, or CMS commands to suit their purposes.

The REXXAID KEYWIN menu contains the following lines:

```
save                    'Save
execdrop &fn            'ExecDrop
&fn                     'Run
exec rexxc &fn          'Compile
XEDIT &fn listing       'Listing
                        '————
set num on              'Num On
set num off             'Num Off
fm A                    'FM A
                        '————
erase &fn &ft &fm       'Erase
quit                    'Quit
```

KEYWIN commands (to the left of the descriptions) may contain the file-id variables '&fn', '&ft', and '&fm'. Appropriate substitutions are made before commands are executed.

'XCMDS KEYWIN' defines a set of XEDIT subcommands that are useful in editing any CMS file. This menu is assigned to XEDIT PF key 12.

The XCMDS KEYWIN menu contains the following lines:

```
                        'Common
                        '————
add                     'Add
all                     'All
backward                'Backward
bottom                  'Bottom
delete                  'Delete
delete *                'Delete *
file                    'File
forward                 'Forward
next                    'Next
num off                 'Num  Off
num on                  'Num  On
prefix off              'Pref Off
prefix on               'Pref On
qquit                   'QQuit
quit                    'Quit
reset                   'Reset
save                    'Save
```

```
top                    'Top
x                      'X
                       'Other
                       '————
cancel                 'Cancel
duplicat               'Duplicat
file                   'FFile
get                    'Get
help XEDIT menu        'Help
hextype 1              'HexType/1
input                  'Input
left 1Ø                'Left/1Ø
lowercas               'LowerCas
powerinp               'PowerInp
put *                  'Put/*
recover 1              'Recover/1
rgtleft                'RgtLeft
right 1Ø               'Right/1Ø
scale off              'Scale Off
scale on m             'Scale On
screen 1               'Screen/1
screen 2 h             'Screen/2H
screen 2 v             'Screen/2V
ssave                  'SSave
up                     'Up
uppercas               'UpperCas
verify 1 *             'Ver 1 *
verify h 1 *           'Ver h 1 *
```

‘REXXAID XEDIT’ is an alternative XEDIT profile that links together the files and macros that constitute the basic REXX program development aid.

The REXXAID XEDIT macro follows:

```
/*********************************************************************/
/* REXXAID XEDIT Profile.                                            */
/*********************************************************************/
/* Set reserved line PF key help text colour.                        */
c = 'T' /* Options: B D G P R T W Y */
/* Enter specific XEDIT session tailoring commands below.            */
'SET CASE MIXED IGNORE'
/* Assign PF key functions and labels below. Limit label text to nine */
/* characters.                                                       */

pf1function  = 'MACRO KEYWIN 1 REXXHELP'
pf1label     = 'REXXHelp     '
pf2function  = 'SOS LINEADD  '
pf2label     = 'LineAdd      '
pf3function  = 'QUIT         '
```

```
pf3label     = 'Quit         '
Pf4function  = 'BEFORE TABKEY'
pf4label     = 'Tabkey       '
pf5function  = '             '
pf5label     = '             '
pf6function  = 'MACRO KEYWIN 6 REXXAID'
pf6label     = 'REXX Aid     '
pf7function  = 'BACKWARD     '
pf7label     = 'Backward     '
pf8function  = 'FORWARD      '
pf8label     = 'Forward      '
pf9function  = 'ONLY =       '
pf9label     = ' =           '
PF10function = 'RGTLEFT      '
PF10label    = 'RgtLeft      '
Pf11function = 'SPLTJOIN     '
pf11label    = 'SpltJoin     '
Pf12function = 'MACRO KEYWIN 12 XCMDS'
pf12label    = 'XCmds        '

/* Ensure the COMMAND LINE is on; enable XEDIT for mouse clicks;      */
/* set the PF KEYS as defined above; set RESERVED LINES.              */

'CMDLINE ON'
'ENTER BEFORE MACRO HOTKEYS'

'PF1'  pf1function; 'PF2'  pf2function; 'PF3'  pf3function
'PF4'  pf4function; 'PF5'  pf5function; 'PF6'  pf6function
'PF7'  pf7function; 'PF8'  pf8function; 'PF9'  pf9function
'PF10' pf10function;'PF11' pf11function;'PF12' pf12function
'RESERVE -4' c 'N  P',
  '1='Left(pf1label,10) '2='Left(pf2label,10) '3='Left(pf3label,10),
  '4='Left(pf4label,10) '5='Left(pf5label,10) '6='Left(pf6label,10)
'RESERVE -3' c 'N  F',
  '7='Left(pf7label,10) '8='Left(pf8label,10) '9='Left(pf9label,9),
  '10='Left(pf10label,9) '11='Left(pf11label,9) '12='Left(pf12label,9)
Exit(0)
```

The REXX development aid can be specified as the alternative XEDIT profile for all EXECs and macros through the use of the PETPROF XEDIT macro (*VM Update*, Issues 152 and 153, April and May 1999). Alternatively, the REXX development aid can be specified when a file is opened:

```
XEDIT TRYIT EXEC (PROF REXXAID
```

Figure 1 illustrates the XEDIT reserved lines and one of the pop-up menus displayed by the REXXAID XEDIT macro. In this example, PF key 6 has been pressed (or selected with a mouse click), resulting in the pop-up menu.

In summary, then, the following files are required to implement the REXX development aid:

- HOTKEYS XEDIT – enables reserved line help text to be mouse-clicked.
- KEYWIN XEDIT – displays pop-up menus of commands and subcommands.
- REXXHELP KEYWIN – contains the pop-up menu of relevant Help commands.

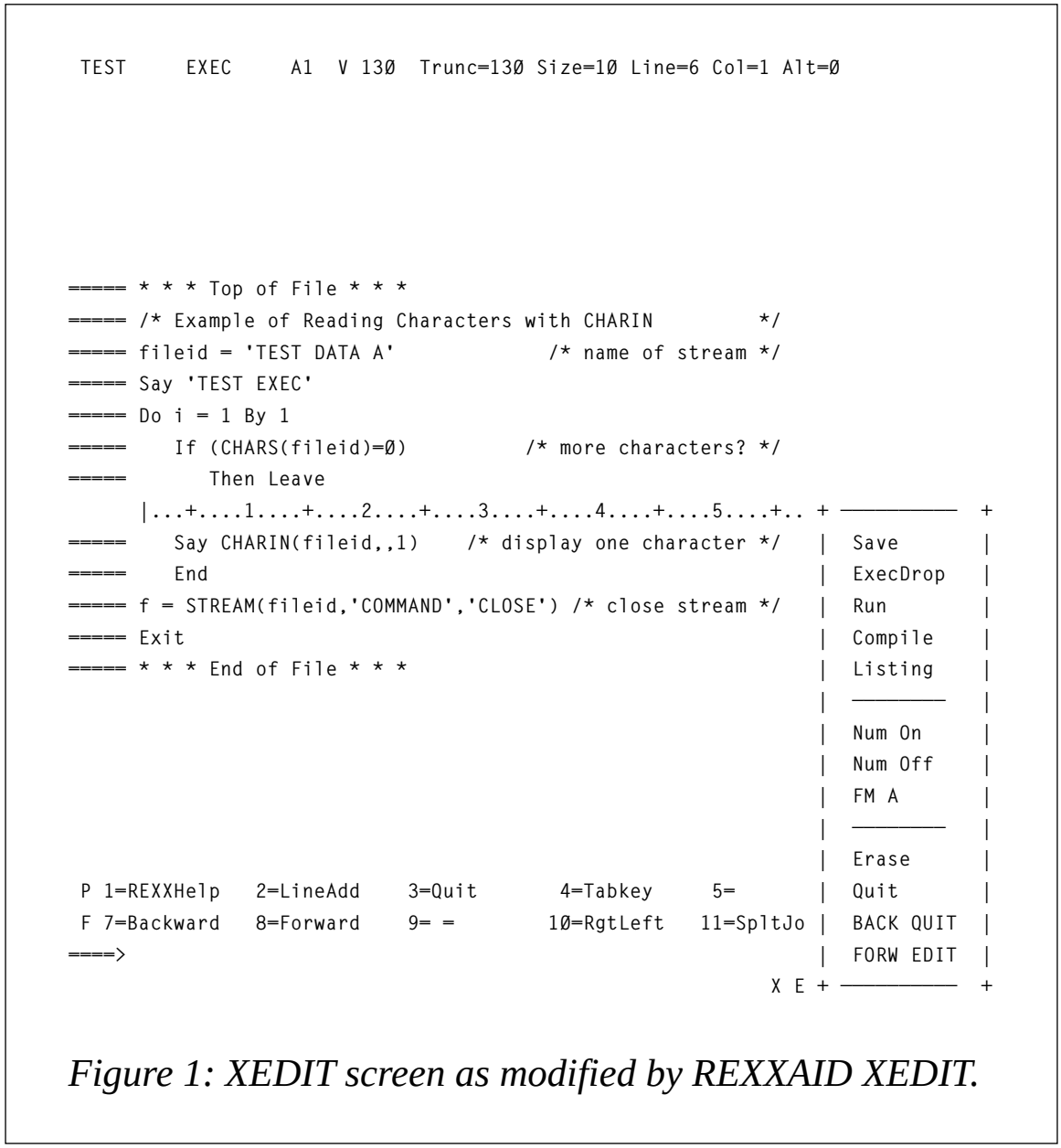

Figure 1: XEDIT screen as modified by REXXAID XEDIT.

- REXXAID KEYWIN – contains the pop-up menu of REXX-related commands.
- XCMDS KEYWIN – contains the pop-up menu of XEDIT subcommands.
- REXXAID XEDIT – activates the REXX development aid.

AID FOR DEVELOPING SCRIPT (DCF) DOCUMENTS

The following files are required to implement a simple Script document development aid:

- DCFHELP KEYWIN
- DCFAID KEYWIN
- 3270 OPTIONS
- FILE OPTIONS
- 3820 OPTIONS
- XCMDS KEYWIN
- DCFAID XEDIT.

'DCFHELP KEYWIN' defines a menu of calls to on-line help that are most relevant to creating Script documents. This menu is assigned to XEDIT PF key 1. DCFHELP KEYWIN should be modified to point to the proper on-line help which is locally available, including BookManager books. The following lines constitute a 'sample' menu which may not work 'as-is' at all installations:

```
                        'DCF Help
help script   task      '* SCRIPT
help XEDIT    menu      '* XEDIT
help bookmgr            'BookMgr
help printers           'Printers
open dcf                'DCF Book
help glossary           'Glossary
open standards          'Standards
```

'DCFAID KEYWIN' defines a menu of CMS and Script commands that are relevant to manipulating Script documents. This menu is assigned to XEDIT PF key 6. Modifications to this menu should be

made as appropriate to suit local document development practices.

```
save                          'Save
                              '————
XEDIT  3270 options           '3270 Opts
script &fn  (options(3270)    'Scrpt3270
                              '————
XEDIT  file options           'File Opts
erase  $&fn script a          'EraseFile
script &fn  (options(file)    'ScrptFile
XEDIT  $&fn script a          'XEDITFile
print  $&fn script A          'PrintFile
                              '————
XEDIT  3820 options           '3820 Opts
erase  &fn  list3820 A        'Erase3820
script &fn  (options(3820)    'Scrpt3820
print  &fn  list3820 A        'Print3820
                              '————
prefix off                    'PrefOff
prefix on                     'PrefOn
quit                          'Quit
```

The Script development aid menu, as defined above, supports three kinds of formatting:

- Document is formatted and displayed on a 3270 screen.
- Document is formatted for a 3270 screen, but the formatted copy is saved to a disk file.
- Document is formatted for printing on a 3820 class printer, and the formatted copy is saved to a disk file.

The DCF options required to accomplish these and other run-time specific formatting (eg TWOPASS, INDEX) are contained in the appropriate options files:

- 3270 OPTIONS
- FILE OPTIONS
- 3820 OPTIONS.

Further discussion of DCF options files is beyond the scope of this article.

The ‘XCMDS KEYWIN’ menu file has been discussed previously. Since this menu consists of standard XEDIT subcommands, it may be

appropriate to include XCMDS KEYWIN with every file development aid that is constructed with these techniques.

'DCFAID XEDIT' is an alternative XEDIT profile that links together the files and macros which constitute the basic Script document development aid.

The DCFAID XEDIT macro follows:

```
/*********************************************************************/
/* DCFAID XEDIT Profile.                                             */
/*********************************************************************/

/* Set reserved line PF key help text colour.                        */
c = 'T' /* Options: B D G P R T W Y */
/* Enter specific XEDIT session tailoring commands below.            */
'SET CASE MIXED IGNORE'
/* Assign PF key functions and labels below. Limit label text to nine */
/* characters.                                                       */

pf1function  = 'MACRO KEYWIN 1 DCFHELP'
pf1label     = 'DCFHelp     '
pf2function  = 'SOS LINEADD  '
pf2label     = 'LineAdd      '
pf3function  = 'QUIT         '
pf3label     = 'Quit         '
Pf4function  = 'BEFORE TABKEY'
pf4label     = 'Tabkey       '
pf5function  = '             '
pf5label     = '             '
pf6function  = 'MACRO KEYWIN 6 DCFAID'
pf6label     = 'DCF Aid      '
pf7function  = 'BACKWARD     '
pf7label     = 'Backward     '
pf8function  = 'FORWARD      '
pf8label     = 'Forward      '
pf9function  = 'ONLY =       '
pf9label     = ' =           '
PF10function = 'RGTLEFT      '
PF10label    = 'RgtLeft      '
Pf11function = 'SPLTJOIN     '
pf11label    = 'SpltJoin     '
Pf12function = 'MACRO KEYWIN 12 XCMDS'
pf12label    = 'XCmds        '

/* Ensure the COMMAND LINE is on; enable XEDIT for mouse clicks;     */
/* set the PF KEYS as defined above; set RESERVED LINES.             */

'CMDLINE ON'
```

```
'ENTER BEFORE MACRO HOTKEYS'

'PF1'  pf1function; 'PF2'  pf2function; 'PF3'  pf3function
'PF4'  pf4function; 'PF5'  pf5function; 'PF6'  pf6function
'PF7'  pf7function; 'PF8'  pf8function; 'PF9'  pf9function
'PF10' pf10function;'PF11' pf11function;'PF12' pf12function

'RESERVE -4' c 'N  P',
  '1='Left(pf1label,10) '2='Left(pf2label,10) '3='Left(pf3label,10),
  '4='Left(pf4label,10) '5='Left(pf5label,10) '6='Left(pf6label,10)
'RESERVE -3' c 'N  F',
  '7='Left(pf7label,10) '8='Left(pf8label,10) '9='Left(pf9label,9),
  '10='Left(pf10label,9) '11='Left(pf11label,9) '12='Left(pf12label,9)

Exit(0)
```

The Script development aid can be specified as the alternative XEDIT profile for all Script documents through the use of the PETPROF XEDIT macro (described in *VM Update*, Issue 152, April 1999). Alternatively, the Script development aid can be specified when a file is opened:

```
XEDIT TRYIT SCRIPT (PROF DCFAID
```

Figure 2 illustrates the XEDIT reserved lines and one of the pop-up menus displayed by the DCFAID XEDIT macro. In this example, PF key 6 has been pressed (or selected with a mouse click), resulting in the pop-up menu.

In summary, the following files are required to implement the basic Script document development aid:

- HOTKEYS XEDIT – enables reserved line help text to be mouse-clicked.
- KEYWIN XEDIT – displays pop-up menus of commands and subcommands.
- DCFHELP KEYWIN – contains the pop-up menu of relevant on-line help.
- DCFAID KEYWIN – contains the pop-up menu of Script-related commands.
- 3270 OPTIONS – DCF formatting options.
- FILE OPTIONS – DCF formatting options.

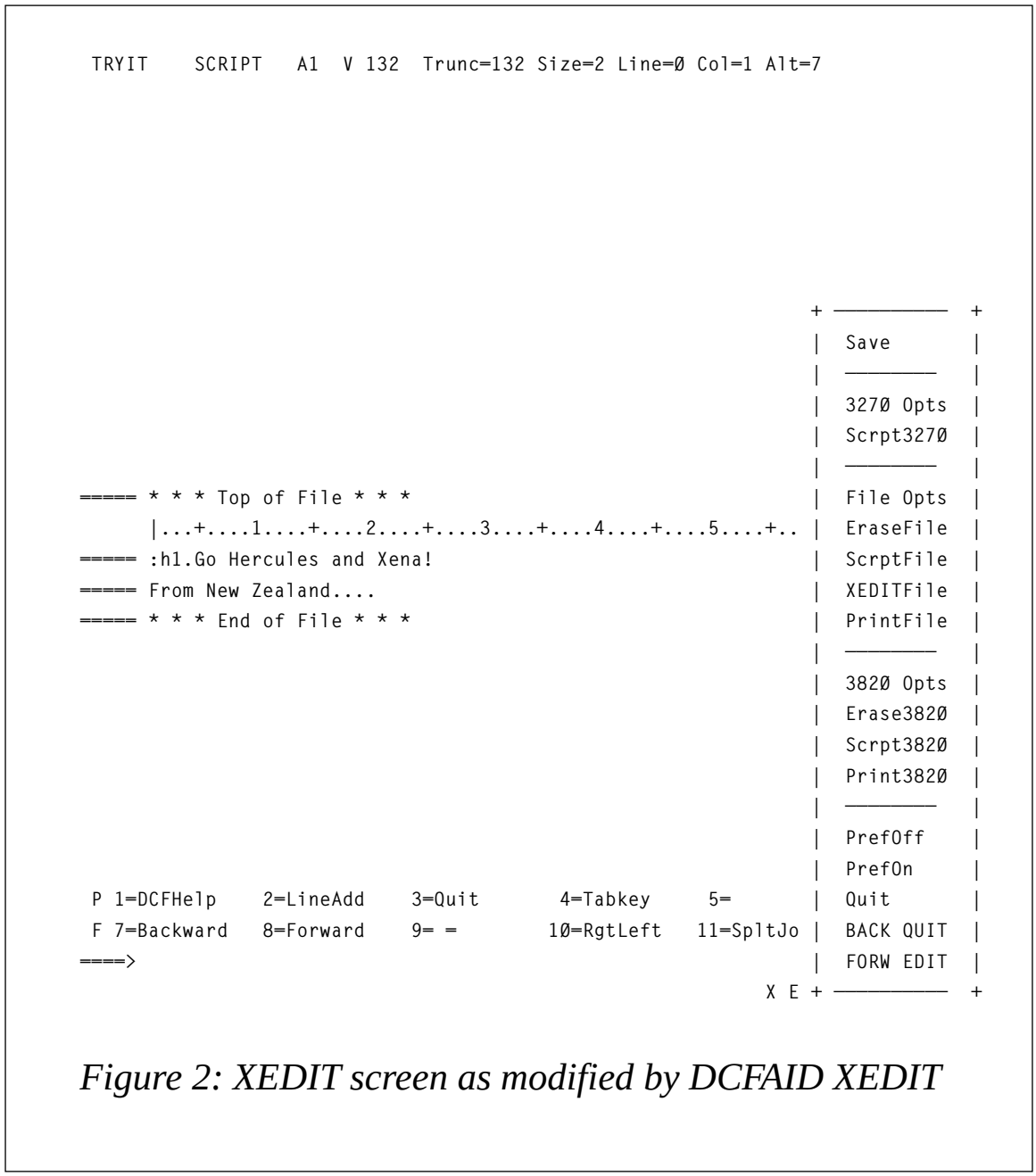

Figure 2: XEDIT screen as modified by DCFAID XEDIT

- 3820 OPTIONS – DCF formatting options.
- XCMDS KEYWIN – contains the pop-up menu of XEDIT subcommands.
- DCFAID XEDIT – activates the Script development aid.

AUTOMATIC XEDIT PROFILE SELECTION

The 'PETPROF XEDIT' macro (presented in *VM Update*, Issue 152, April 1999) selects and executes an appropriate XEDIT customization macro based on the filetype of the current file. The mapping of filetype to customization macro is contained in a file named 'DEFAULT PETPROF' (or 'userid PETPROF'). To implement macro mapping for the types of files discussed in this article, the DEFAULT PETPROF file should contain the following lines:

```
EXEC     REXXAID
XEDIT    REXXAID
SCRIPT   DCFAID
```

Automatic macro mapping can then be achieved by invoking XEDIT as follows:

```
X NEW SCRIPT (PROF PETPROF
```

Alternatively, the PROFILE XEDIT file can be created with the following lines:

```
/* Profile XEDIT */
'MACRO PETPROF'
Exit(0)
```

SUMMARY

A number of XEDIT macro programming techniques can be combined to produce helpful, productivity-enhancing program and document development aids. Developers can build on the examples presented here to create development aids for files containing any programming or markup language.

FURTHER INFORMATION

Further information about the PETs project can be found at the following Web location: http://vm.uconn.edu/~pets/index.html.

Richard G Ellis
Director of Computing Services
University of Connecticut (USA)

The REXX Language Association Web site

Continuing our series of VM Web site reviews, we visit the REXX Language Association Web site. The site can be accessed at http://www.rexxla.org/. 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.

REXX – created in 1979 (see below for 20th birthday information!) by Mike Cowlishaw, has spread to most current computing platforms. But it's still not as widely known or used as other, more trendy languages, such as (Visual) BASIC or Java. In fact, it's sometimes difficult for enthusiastic REXX practitioners to articulate reasons for their positive feelings about the language. A few paragraphs from the link to Ian Collier's 'REXX Introduction' begin to tell the story:

What is REXX?

REXX is a procedural programming language that allows programs and algorithms to be written in a clear and structured way. It is easy to use by experts and casual users alike. REXX has been designed to make easy the manipulation of the kinds of symbolic objects that people normally deal with such as words and numbers. Although REXX has the capability to issue commands to its host environment and to call programs and functions written in other languages, REXX is also designed to be independent of its supporting system software when such commands are kept to a minimum.

General programming using REXX

REXX provides powerful character and arithmetical abilities in a simple framework. It can be used to write simple programs with a minimum of overhead, but it can also be used to write robust large programs. It can be used for many of the programs for which BASIC would otherwise be used, and its layout may look somewhat similar to that of a structured BASIC program. Note, however, that REXX is not BASIC!

Macro programming using REXX

Many applications are programmable by means of macros. Unfortunately, in the Unix world, almost every application has a different macro language. Since REXX is essentially a character manipulation language, it could provide the macro language for all these applications, providing an easy-to-use and consistent interface across all applications. The best examples of such systems are on CMS (IBM's mainframe operating system which was the birthplace of REXX) and on the Amiga. However, IBM's OS/2 is catching up, and now that REXX is freely available on Unix it cannot be long before applications start to appear which have REXX as their macro language. Two products already exist. They are the Workstation Group's uni-XEDIT and Mark Hessling's THE (a link to which is displayed on my REXX title page).

Other applications of REXX

REXX can be used as an 'application glue' language, in a manner similar to that in which shell scripts are often used. Since REXX is able to pass arbitrary command strings for execution by its environment, it can be used to execute Unix programs as well as providing the control language necessary for testing things such as parameters and return codes and acting accordingly.

REXX is often executed by an interpreter, and this permits rapid program development. This productivity advantage makes the language very suitable for modelling applications and products - in other words, for prototype development. REXX is also fairly easy to debug. Once a design has been shown to work satisfactorily, it can be easily recoded in another language if that is required for performance or other reasons.

The design of REXX is such that the same language can effectively and efficiently be used for many different applications that would otherwise require the learning of several languages.

A bonus for visiting Ian's page – and an illustration of the enthusiasm REXX users feel for the language – is viewed by clicking the REXX logo at the bottom of the 'About REXX' page. This leads to his short REXX page, listing favourite and useful links. Many other REXXers

build similar lists, both for their use as bookmark pages and to guide visitors to particular points of interest.

While user groups such as SHARE offer REXX education and news, and myriad books and magazines offer different sorts of information such as tutorial and advanced techniques, another sort of resource exists in the REXX Language Association (REXXLA). Briefly described on the opening page as "an independent organization dedicated to promoting the use of the REXX programming language", it's more fully described on the 'About REXXLA' page:

The REXX Language Association was established to further the understanding and use of the REXX programming language. With the release of Object REXX and NetREXX, REXXLA has expanded its mission to include these languages as well. The Association is headquartered at the Research Triangle Park in North Carolina, and is international in scope with members throughout the world. Supported by a dedicated group of volunteers, the REXX Language Association provides many benefits for its members.

REXXLA sponsors the annual REXX Symposium which features presentations by recognized REXX experts and practitioners, interesting products, and an opportunity to network and to learn from people who share your enthusiasm for REXX. All of the major REXX vendors attend, giving you the opportunity to interact directly with them and provide them with valuable feedback from the field. A REXXLA membership also entitles you to a discounted registration fee at the Symposium.

From the very first, the REXX Language Association has been an active member of the committee working on an ANSI Standard for REXX. That Standard has been approved, and we are now involved in drafting an ANSI Standard for Object REXX and NetREXX. Your membership gives you a voice in this very important work.

If you write sophisticated applications in Object REXX, design Java-enhanced webpages with NetREXX, or simply use REXX to make your life easier on any platform, the REXX Language Association has much to offer you.

As of this writing, REXX has just passed a significant milestone, described on the REXXLA opening page:

Happy 20th Birthday to REXX!

The REXX language turned 20 years old on March 20th. Why is March 20th considered REXX's birthday? According to Mike Cowlishaw, 'it was the day I woke up at 3am with a clear idea of what was needed, and by the end of the day had the initial specification off around the world for comment.'

For those exploring REXX or already committed to it, the 'Links' link provides diverse resources, in categories General, OS/2, Windows, Unix, DOS, Amiga, NetWare, VM, and MVS. The General category begins with links to three IBM REXX pages: 'Mike Cowlishaw's REXX Language Page', the 'Object REXX homepage', and the 'NetREXX homepage'. (Mike's page was reviewed in *VM Update*, Issue 140, April 1998, and the OREXX and NetREXX pages were mentioned.) These three pages are also linked by the graphics appearing at the top of REXXLA pages: the logo from Mike's REXX books, the Object REXX logo, and the stylized network.

The next link, Dave Martin's 'REXX Frequently Asked Questions (FAQ)' page, opens by default in two frames: navigation on the left, content on the right. Before clicking to specific questions, note the opening content page's links to additional FAQs on TSO and VM/ESA REXX. Dave describes his FAQ as "*intended to serve as a useful reference for REXX-related information, aiming for breadth as opposed to depth, with references to other material given where appropriate*". Dave has a slightly different answer to the question 'What Is REXX?':

REXX is a programming language designed by Michael Cowlishaw of IBM UK Laboratories. In his own words: 'REXX is a procedural language that allows programs and algorithms to be written in a clear and structured way.' REXX doesn't look that different from any other procedural language. Here's a simple REXX program:

```
/* Count some numbers */

say "Counting..."
do i = 1 to 10
say "Number" i
end
```

What makes REXX different from most other languages is that it is also designed to be used as a macro language by arbitrary application programs. The idea is that application developers don't have to design

their own macro languages and interpreters. Instead they use REXX as the macro language and support the REXX programming interface.

If a REXX macro comes across an expression or function call that it cannot resolve, it can ask the application to handle it instead. The application only has to support the features that are specific to it, freeing the developer from handling the mundane (and time-consuming) task of writing a language interpreter. And if all applications use REXX as their macro language, the user only has to learn one language instead of a dozen.

Other question areas are REXX and the Internet, Free REXX Products, Commercial Products, REXX and ANSI, NetREXX, The REXX Language Association, REXX Bibliography, Common REXX Coding Errors, Frequently Asked Questions (including 'Is REXX better than <some other language>' and several technical queries), and Copyright Information. Other general links deal with various REXX implementations, describe several REXX books, and offer articles and utilities.

Several links are provided for OS/2 resources; OS/2 was one of the earliest systems to include integrated REXX function and has developed a widespread contingent of REXX fans. VisPro/REXX from Hockware and VX-REXX from Watcom were among the earliest visual development environments. Windows, Unix, DOS, Amiga, VM, Netware, and MVS are all represented, with free and priced tools and services available.

The next major link from the main page deals with standards. Just as REXXLA was born at a REXX Symposium (a yearly event formerly loosely connected with REXXLA and now operated by it), the industry standard for REXX also originated at a Symposium. Brian Marks' 'A brief history of the REXX Standard', begins:

If you are unfamiliar with how standards are developed, the history of the REXX Standard may give some insight. The idea of a standard was first promoted by Linda Green, the IBM representative to the SHARE organization at the time. There was enthusiasm for the idea at the very first REXX Symposium, which allowed Linda to make a case to the authorities, who allowed Linda to convene the first meeting. This was attended by several of the parties with a producer or user interest in

REXX and they 'bootstrapped' themselves to being a committee by suggesting Brian Marks as chairman, a choice subsequently endorsed by the Information Technology Industry Council which administers this class of standard.

Ultimately, decisions of standardizing groups are the result of majority voting. In practice, the consensus achieved meant that there was almost no voting, apart from the formally required vote that the draft standard was ready for public review.

Standardization seems to be an elusive goal – while a standard exists, "There are not yet any implementations that claim full conformance with the ANSI standard, although some implementations now contain features that are also new to the standard, for example date conversions and extra built-in functions."

It's often a chore sorting theoretical wisdom from old-fashioned experience-based advice. The REXXLA newsletter, linked from the main page, lets people using REXX in day-to-day projects, or developers implementing REXX language facilities or add-ons, report on lessons learned, projects underway, or part-baked ideas for future efforts.

The first newsletter posted, September 1998, includes six columns and three articles. The columns, entitled 'Dr Rx', 'Editorial Ramblings', 'The Scriptor's Maid', 'Tip of the Month', 'Train/REXX', and 'REXXwishes', present short and different REXX perspectives. As one might guess, Train/REXX describes a less-than-pleasant REXX problem, beginning:

There are several places in my code where I use stem variables indexed by something other than numbers and I have discovered an instance where seemingly proper code can be just imprecise enough. The particular example I ran afoul of used an ISPF table, but this could happen almost anywhere.

and ending:

Surprise ! ct 'ABCDE' does NOT equal ct 'ABCDE'. The first one has a value > 0, but it's lost forever, because we no longer know how many blanks to add to produce the right result; the second version has the

default value of zero unless we're REALLY unlucky, in which case it will have a value that looks like it might be right but is actually dead wrong.

Mike Cowlishaw's article 'Easier Java Programming [part 1]', which introduces the NetREXX programming language, combining REXX's power and programming ease with Java's portability, begins:

The great strength of Java is the Java Virtual Machine and class libraries, which together form a programmable environment that is available on a wide variety of computers. The Java programming language is one way of creating programs and classes for that environment – but it is not necessarily the only way. Just as different artists prefer different tools and techniques, so different programming languages suit different programmers.

NetREXX is a new programming language that is designed specifically to make it easier for programmers to take advantage of the Java environment, without losing any of the capabilities offered by the Java language. The productivity gains of using NetREXX are available to anyone currently programming in Java, and the ease-of-use of NetREXX will encourage more new users to exploit the advantages of the Java environment.

The January 1999 newsletter, the most recently posted at the time of writing, shows enlarged structure and scope, with a news item about the November 1998 standards meeting, excerpts from a 1998 REXX Symposium presentation, reviews and reports (of a free REXX interpreter for DOS and UNIX and an OS/2 SIG, respectively), four columns, and four articles.

The Dr Rx column provides its regular reader/member interaction, with expert information helping with common and obscure REXX queries. The REXXwishes column by F Scott Ophof (the newsletter's editor), which deals with a popular commercial product written in REXX, begins by explaining the product's strength and user friendliness:

This month's first subject is that great MUA by Richard Schafer called 'Mailbook' (originally available as freeware 'RiceMail'), which runs on IBM'S VM/CMS operating system. ('MUA' stands for 'Mail User

Agent', meaning the application used to read, write, sort, and index your e-mail. To actually send and receive e-mail items, an MTA (Mail Transfer Agent) is used; they usually work in the background, ie invisible to the user.)

Mailbook is an application that originated in the VM/CMS world as a set of REXX macros built around the IBM editor called XEDIT. Anyone used to XEDIT has little – if any – problem using Mailbook, because almost any application in the CMS world will use the XEDIT editor. (Yes, there is also the ISPF editor; for all practical purposes it is a rather close relative of XEDIT).

Since Mailbook was written in REXX, it was – and still is – quite easy to write modifications for by any CMSer familiar with REXX. Users can also (re)configure Mailbook with their own preferences. The configuration options span a huge range of possibilities. Mailbook also makes use of the CMS NAMES utility, which is the CMS equivalent of an addressbook on a PC. Although that is not doing NAMES real justice; NAMES is actually a free-form database and can be used for much more than just addresses. In combination with Mailbook it does things like expanding nicknames to full names and e-mail addresses, automatically filing both sent and received items in the relevant mailbook (folder, notebook file), and so forth.

Articles include 'I think I understand how the FORMAT builtin function works' by Brian Marks and 'Safe REXX (part 2)' by Shmuel (Seymour J) Metz.

While the REXXLA Web site content is available to the public, REXXLA operation depends on support of members and friends – by paying dues and attending the yearly symposium. Membership, costing $24/year, is available by clicking the cheerful 'Join!' button on the 'About REXXLA', page, which provides a form to be mailed to the group's headquarters. And full information on the annual Symposium is provided by links in the body of the main page. Sadly, the event – the tenth! – from 3-5 May 1999 in Florida will be over by the time this article is published. But information on the event and news of next year's gathering will appear on the Web site.

The final main page link discusses REXX Year 2000 considerations, beginning:

The REXX language itself does not contain any 'Year 2000' (Y2K) problems and thus can be considered Y2K compliant. However, there are Y2K considerations in the use of certain REXX built-in functions. Be sure to use the correct Date() functions to prevent Y2K problems within your code. For example, options E (European), U (USA), J (Julian), and O (Ordered) use two- digit years. The options S (Sorted), L (local), and N (Normal) use four-digit years. Use of four-digit years in programs would be mandatory to ensure Y2K compliance.

The REXXLA Web site offers tactical and strategic resources for REXX users. The former provide coding advice, debugging tips, design pointers, interface techniques. The latter include awareness of the evolving REXX language, access to REXX experts and compiler/ interpreter implementers, and participation in the world-wide REXX community. It's a site well worth visiting, and an organization worth supporting with membership dues.

Gabe Goldberg
Computers and Publishing (USA)

A full screen console interface – part 11

Editor's note: the following 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.*

CSCULC ASSEMBLE

```
         TITLE 'CSCULC - CSC Process User Locate command'
CSCULC   START X'Ø1ED78'
         PRINT NOGEN
         CSCHDR                        User Locate command
*
* Process LOCATE command
*
```



```
         USING UIDSECT,R8              UID (user) Block
         USING CCHSECT,R7              CCH (cache) Block
         USING MSGSECT,R5              MSG (message) block
         SPACE
         LA    R5,ULCMSG               Address message work area
         MVI   MSGARBCH,ULCARBCH       Move unprintable arbitrary char
         MVI   MSGANYCH,ULCANYCH       Move unprintable any character
         CLI   Ø(R6),C'/'              Was command omitted?
         BE    ULC1ØØ                  Yes, just build search mask
         SR    RØ,RØ                   No table to search
         GO    CSCSCN                  Skip command name and blanks
         BNZ   ULC5ØØ                  Nothing found, too bad
ULC1ØØ   BAS   R14,BLDMASK             Build mask
         BNZ   ULC9ØØ                  Something went wrong, forget it
         BAS   R14,LOCATEUP            Locate data
ULC9ØØ   BACK
         SPACE
ULC5ØØ   MSG   Ø31Ø,USER               Missing string to locate
         B     ULC9ØØ
         SPACE 3
*
* Process MATCH command
*
CSCULCMT RELOC
         LA    R5,ULCMSG               Address message work area
         MVI   MSGARBCH,C'*'           Move arbitrary and any character
         MVI   MSGANYCH,C'%'           *** * TO BE CHANGED LATER * ***
         CLI   Ø(R6),C'\'              Was command omitted?
         BE    MAT1ØØ                  Yes, build search mask
         AR    R6,R1                   Skip over command name
MAT1ØØ   LA    R6,1(,R6)               Allow for one space after name
         BAS   R14,BLDMASK             Build mask
         BNZ   MAT9ØØ                  Something went wrong, forget it
         BAS   R14,LOCATEUP            Locate data
MAT9ØØ   BACK
         SPACE 3
*
* Process DOWNLOCATE command (DLOCATE)
*
CSCULCDL RELOC
         LA    R5,ULCMSG               Address message work area
         MVI   MSGARBCH,ULCARBCH       Move unprintable arbitrary char
         MVI   MSGANYCH,ULCANYCH       Move unprintable any character
         SR    RØ,RØ                   No table to search
         GO    CSCSCN                  Skip blanks after command name
         BZ    DL1ØØ
         MSG   Ø31Ø,USER               Nothing found, too bad
         B     DL9ØØ
         SPACE
DL1ØØ    BAS   R14,BLDMASK             Build mask
         BNZ   DL9ØØ                   Something went wrong, forget it
```

```
         BAS   R14,LOCATEDN            Locate data
DL900    BACK
         SPACE 3
*
* Process DOWNMATCH command (DMATCH)
*
CSCULCDM RELOC
         LA    R5,ULCMSG              Address message work area
         MVI   MSGARBCH,C'*'          Move arbitrary and any character
         MVI   MSGANYCH,C'%'          *** * TO BE CHANGED LATER * ***
         LA    R6,1(R1,R6)            Allow for one space after name
         BAS   R14,BLDMASK            Build mask
         BNZ   DM900                  Something went wrong, forget it
         BAS   R14,LOCATEDN           Locate data
DM900    BACK
         SPACE 3
*
* Process GO command
*
CSCULCGO RELOC
         LA    R1,DIAG000C            Address DIAG work area
         DIAG  R1,R0,X'000C'          Get current date and time
         MVC   DATECURR(2),DIAG000C+6 Convert date to yy/mm/dd
         MVI   DATECURR+2,C'/'
         MVC   DATECURR+3(5),DIAG000C
         MVC   TIMECURR,DIAG000C+8    Copy current time
         MVC   DATEYY(8),DATECURR     Copy current date to build area
         MVC   TIMEHH(8),TIMEZERO     Start with time 00:00:00
         BAS   R14,DATE               Get new date and time from input
         BAS   R14,VALIDATE           Validate both date and time
         L     R7,UIDFREE1            Get a record from Free List
         SR    R0,R0
         ST    R0,CCHRECNO            Invalidate entry
         MVC   CCHDATE,DATEYY         Copy reference Date and Time
         MVC   CCHTIME,TIMEHH
         GO    CSCRDFGO               Search for the record
         BZ    GO400                  Found it...
         MSG   0356,USER              Record not found
         B     GO900
         SPACE
GO400    BAS   R14,FMTSCRN            Record found, format user screen
GO900    BACK
         SPACE
*
* Build new Date and Time as modified by the user
*
DATE     EQU   *
         ST    R14,ULCSV14
         BAS   R14,GETVAL             Get first value
         BZ    DATE100
         MSG   0310,USER              Nothing found, it is missing
```

```
         B     GO9ØØ                  That's all
         SPACE
DATE1ØØ  CLI   Ø(R6),C':'             Check for a ":" separator
         BE    TIME1ØØ                Found it, start with hours
         MVC   DATEDD,ULCVAL          Move day to build area
         CLI   Ø(R6),C'/'             Is date complete
         BNE   TIME                   Yes, build time
         BAS   R14,GETVAL             No, we may have a mm/dd
         MVC   DATEMM,DATEDD          Move month
         MVC   DATEDD,ULCVAL          Move day
         CLI   Ø(R6),C'/'             Is it complete now?
         BNE   TIME
         BAS   R14,GETVAL             No, we must have yy/mm/dd
         MVC   DATEYY,DATEMM          Move year
         MVC   DATEMM,DATEDD               month
         MVC   DATEDD,ULCVAL               day
TIME     C     R6,CSCBUFFE            Anything left?
         BNL   TIME9ØØ                No, all done
         CLI   Ø(R6),C' '             Yes, a blank separates date/time
         BNE   TIME3ØØ                If not, that's an error
         BAS   R14,GETVAL             Get hours
TIME1ØØ  MVC   TIMEHH,ULCVAL          Move hours to build area
         CLI   Ø(R6),C':'             Check for a following ":"
         BNE   TIME2ØØ                Not there, anything else is bad
         CLI   1(R6),C' '             No space allowed after the ":"
         BE    TIME2ØØ
         BAS   R14,GETVAL             Get minutes
         MVC   TIMEMM,ULCVAL          Move minutes, we had hh:mm
         CLI   Ø(R6),C':'             Another ":"?
         BNE   TIME2ØØ                No, almost done
         CLI   1(R6),C' '             No space allowed after the ":"
         BE    TIME3ØØ
         BAS   R14,GETVAL             Get seconds
         MVC   TIMESS,ULCVAL          Move seconds
         CLI   Ø(R6),C':'             Another ":"?
         BE    TIME3ØØ                That's too much
TIME2ØØ  C     R6,CSCBUFFE            Any unexpected data
         BNL   TIME9ØØ                No, return
         SR    RØ,RØ                  No table to search
         GO    CSCSCN                 Locate next value
         BNZ   TIME9ØØ                Only blanks, that's OK
TIME3ØØ  MSG   Ø312,USER              Display unexpected data
         B     GO9ØØ                  Goodbye
         SPACE
TIME9ØØ  L     R14,ULCSV14
         BR    R14
         SPACE
*
* Get next value from Date or Time entered
*
*         A non-zero cc is return if data not found
*
```

```
GETVAL   EQU   *
         ST    R14,GETSV14
         SR    RØ,RØ                    No table to search
         GO    CSCSCN                   Locate next value
         BNZ   GET9ØØ                   Not found
         GO    CSCSCNVN                 Make sure it is numeric
         BNZ   GET6ØØ                   Bad news, it is not
         LA    RØ,2                     Maximum length is 2
         CR    RØ,R1
         BL    GET7ØØ                   Too long, display error
         AR    R6,R1                    Fool the SCAN routine
         LA    RØ,1                     Address the separator "/" or ":"
         ST    RØ,SCANLEN               Store a length of 1
         C     R6,CSCBUFFE              Anything left?
         BNL   GET1ØØ                   No, that's ok, it is ...nn
         CLI   Ø(R6),C' '               Is there a blank after?
         BE    GET1ØØ                   That's OK too
         CLI   Ø(R6),C':'               Is it a ":"?
         BE    GET1ØØ                   That's good, nn: or nn:mm
         CLI   1(R6),C' '               Is there a blank after
         BE    GET6ØØ                   Yes, not allowed
         CLI   Ø(R6),C'/'               Last chance, is it a "/"?
         BNE   GET6ØØ                   No, too bad, display error
GET1ØØ   CVD   R2,ULCCONV               Convert value to decimal
         OI    ULCCONV+7,X'ØF'          Remove sign
         UNPK  ULCVAL,ULCCONV           Unpack to work field
         CR    R14,R14                  Generate a zero cc
GET9ØØ   L     R14,GETSV14              Return
         BR    R14
         SPACE
GET6ØØ   MSG   Ø311,USER                Invalid data found
         B     GO9ØØ
         SPACE
GET7ØØ   MSG   Ø352,USER                Value too long
         B     GO9ØØ
         SPACE
*
* Validate Date and Time entered by the user
*
VALIDATE EQU   *                        Validate Date and Time
         CLC   DATEMM,MAXMM             Quick and dirty Date validation
         BH    VAL5ØØ                   Month is over 12
         CLC   DATEDD,MAXDD
         BH    VAL5ØØ                   Day is over 31
         CLC   TIMEHH,MAXHH             Hours must be ØØ-23
         BH    VAL6ØØ                   Too big, bad time
         CLC   TIMEMM,MAXMMSS           Minutes must be ØØ-59
         BH    VAL6ØØ
         CLC   TIMESS,MAXMMSS           Seconds must be ØØ-59
         CLC   DATECURR,DATEYY          Is Date into the future
         BL    VAL7ØØ                   Yes, get the time machine
         BHR   R14
```

```
         CLC   TIMECURR,TIMEHH
         BL    VAL7ØØ                  We need a small time machine
         BR    R14
         SPACE
VAL5ØØ   MSG   Ø353,USER                Invalid Date
         B     GO9ØØ
         SPACE
VAL6ØØ   MSG   Ø354,USER                Invalid Time
         B     GO9ØØ
         SPACE
VAL7ØØ   MSG   Ø355,USER                Future Date/Time
         B     GO9ØØ
         SPACE 3
*
* Build mask to locate
*
BLDMASK  EQU   *                        Build search mask
         ST    R14,ULCSV14
         LA    RØ,L'MSGMASK            Maximum length
         L     R1,CSCBUFFE             Address end of mask
         BCTR  R1,Ø                    Address mask's last character
         LR    R2,R1
         SR    R1,R6                   Mask length - 1
         CR    R6,R2                   Check first and last byte addrs
         BH    BLDM5ØØ                 Nothing, no mask supplied
         CR    R1,RØ                   Compare length of mask and field
         BNL   BLDM6ØØ                 Too long
         EX    R1,BLDMMVC              Move mask to message work area
         EX    R1,BLDMTR               Translate to uppercase
         LA    R2,MSGMASK(R1)          Address mask's last byte
         CLI   MSGARBCH,ULCARBCH       Is this a LOCATE or DLOCATE
         BNE   BLDM1ØØ
         CLC   MSGMASK(1),Ø(R2)        Yes, check first and last bytes
         BE    BLDMØ1Ø
         LA    R1,1(,R1)               They are different, add last one
         CR    R1,RØ                   If we have space
         BNL   BLDM6ØØ                 No, too long
         LA    R2,1(,R2)               Advance pointer past added byte
BLDMØ1Ø  MVI   MSGMASK,ULCARBCH        Change first arbitrary character
         MVI   Ø(R2),ULCARBCH          Change or add last one too
BLDM1ØØ  LA    R2,1(,R2)               Address end of mask
         ST    R2,MSGMASKE             Store into work area
         CR    R14,R14                 Generate a zero cc
BLDM9ØØ  L     R14,ULCSV14
         BR    R14
         SPACE
BLDM5ØØ  MSG   Ø31Ø,USER                No mask specified?
         LTR   R14,R14                 Generate non-zero cc
         B     BLDM9ØØ
         SPACE
BLDM6ØØ  MSG   Ø35Ø,USER                Mask too long
         LTR   R14,R14                 Generate non-zero cc
```

```
         B     BLDM900
         SPACE
BLDMMVC  MVC   MSGMASK(*-*),0(R6)      Move mask
BLDMTR   TR    MSGMASK(*-*),CSCUPP     Uppercase mask
         SPACE 3
*
* Perform LOCATE or MATCH
*
LOCATEUP EQU   *                       Prepare to go Up
         ST    R14,ULCSV14
         L     R15,@SCRDFPR            Load RDF routine - Read Previous
         A     R15,0(,R15)             Skip timestamp
         ST    R15,ULCREAD             Store it for LOCATE
         L     R7,UIDBUFF1             Address top line
         SR    R0,R0
         C     R0,CCHRECNO             Is record number valid?
         BNE   LOCATE                  Yes...
*        CLI   CCHUSER,X'00'           Is it TOF line?
*        BE    LOC200                  Yes, nothing to do... not found
*        GO    CSCRDFLT                Must be EOF, start with last
*        BZ    LOC300                  Found something, check it
         TM    UIDOPT2,UIDAUTO         Is user in refresh mode?
         BZ    LOC200
         GO    CSCRDFLT                Yes, someone cleared the screen
         BZ    LOC300                  Found something, check it
         B     LOC200                  Nothing found
         SPACE
LOCATEDN EQU   *                       Prepare to go Down
         ST    R14,ULCSV14
         L     R15,@SCRDFNT            Load RDF routine - Read Next
         A     R15,0(,R15)             Skip timestamp
         ST    R15,ULCREAD             Store it for LOCATE
         L     R7,UIDBUFF1             Address top line
         SR    R0,R0
         C     R0,CCHRECNO             Is record number valid?
         BNE   LOCATE                  Yes...
*        CLI   CCHUSER,X'00'           Is it TOF line?
*        BNE   LOC200                  No, must be EOF... not found
         GO    CSCRDFFT                Yes, start with first record
         BZ    LOC300                  Found something, check it
         B     LOC200                  Nothing found
         SPACE
LOCATE   EQU   *
         L     R15,ULCREAD               RDF routine, Previous or Next
         GO                            , Read record
         BZ    LOC300                    We found it, process
LOC200   MSG   0351,USER                 String or Mask not found
         B     LOC900
         SPACE
LOC300   SR    R1,R1                     Required by next IC
         IC    R1,CCHRLEN                Load message length
```

```
         LA    R2,ULCCCH               Move to cache work area
         STC   R1,CCHRLEN-CCHSECT(,R2) Store record length
         BCTR  R1,Ø                    Adjust for EXecute
         EX    R1,LOCMVC               Move message text
         EX    R1,LOCTR                Uppercase everything
         ST    R7,ULCADDR              Save or record address (cache)
         LR    R7,R2                   Copy to R7 for LINK LOCATE
         LA    R5,ULCMSG               Address message work area
         LINK  LOCATE                  Search data
         L     R7,ULCADDR              Restore record address (cache)
         BNZ   LOCATE                  Record does not match string
         BAS   R14,FMTSCRN             Record found, format user screen
         SPACE
LOC9ØØ   L     R14,ULCSV14
         BR    R14
         SPACE
LOCMVC   MVC   CCHDATA-CCHSECT(*-*,R2),CCHDATA
LOCTR    TR    CCHDATA-CCHSECT(*-*,R2),CSCUPP
         SPACE 3
*
* Format user screen
*
*        Input R7 addresses reference record (cache image)
*
FMTSCRN  EQU   *                       Format user screen
         ST    R14,FMTSV14
         L     R1,UIDBUFF2             Address last screen line
         LINK  ADD                     Add it as last buffer record
         L     R7,UIDBUFF1             Address first line
         LINK  DELETE                  Delete it
         TM    UIDOPT2,UIDAUTO         Is user in refresh mode?
         BZ    FMT1ØØ
         NI    UIDOPT2,X'FF'-UIDAUTO   Yes, reset option
         OI    UIDOPT4,UIDBHDR         Remember to refresh Header line
FMT1ØØ   OI    UIDOPT4,UIDBSCR         Option to build user screen
         L     R7,UIDBUFF2             Get address of bottom line
         ST    R7,NEWTOP               Save as new top line
FMT2ØØ   L     R7,UIDBUFF1             Address top line
         C     R7,NEWTOP               Is it the old bottom line
         BE    FMT8ØØ                  Yes, all done
         LINK  DELETE                  No, delete top line
         L     R7,UIDBUFF2             Address bottom line
         GO    CSCRDFNT                Get next record
         BNZ   FMT3ØØ                  Not found, build EOF line
         L     R1,UIDBUFF2             Add as last record
         LINK  ADD
         B     FMT2ØØ
         SPACE
FMT3ØØ   LINK  ADDEOFB                 Add EOF after last record
FMT4ØØ   L     R7,UIDBUFF1             Address top line
         C     R7,NEWTOP               Is it the old bottom line
```

```
         BE    FMT8ØØ                  Yes, all done
         LINK  DELETE                  No, delete top line
         LINK  ADDBLKB                 Add blank line after last line
         B     FMT4ØØ                  Clear all lines after EOF line
         SPACE
FMT8ØØ   TM    UIDOPT3,UIDWRAP         Is WRAP switch On?
         BZ    FMT9ØØ                  No, done
         GO    CSCWRPTP                Yes, build partial lines
FMT9ØØ   L     R14,FMTSV14
         BR    R14
         SPACE 3
ULCMSG   DS    ØD                      Work area for message entry
         ORG   *+MSGSIZEB              Entry size in bytes
ULCCCH   DS    ØD                      Work area for cache entry
         ORG   *+CCHSIZEB              Entry size in bytes
         SPACE
ULCSV14  DS    F                       Save area for R14
GETSV14  DS    F                       Save area for GETVAL R14
FMTSV14  DS    F                       Save area for FMTSCRN R14
ULCADDR  DS    F                       Save area for cache address
ULCREAD  DS    F                       CSCRDF routine to use
NEWTOP   DS    F                       Area to identify top line
ULCARBCH EQU   X'FF'                   Unprintable arbitrary character
ULCANYCH EQU   X'FE'                   Unprintable any character
ULCCONV  DS    D
DATECURR DS    CL8                     Current date from CP
TIMECURR DS    CL8                     Current time from CP
TIMEZERO DC    C'ØØ:ØØ:ØØ'             Reference time
DATEYY   DS    CL2                     Date modified by the user
         DS    C
DATEMM   DS    CL2
         DS    C
DATEDD   DS    CL2
TIMEHH   DS    CL2                     Time modified by the user
         DS    C
TIMEMM   DS    CL2
         DS    C
TIMESS   DS    CL2
ULCVAL   DS    CL2
MAXMM    DC    C'12'                   Maximums for Date validation
MAXDD    DC    C'31'
MAXHH    DC    C'23'                   Maximums for Time validation
MAXMMSS  DC    C'59'
         SPACE 3
         CSCDATA
         CSCDS (UID,CCH,MSG)
         REGEQU
         END
```

Add class 04 to the USER statements and regen CSCSVP to have the GO, LOCATE, and MATCH commands working.

CSCURL ASSEMBLE

Add class 05 to the USER statements and regen CSCSVP to have the RELEASE command working. This command releases messages on Hold. It also supports messages with the UNIQUE attribute and self-releasing messages.

```
         TITLE 'CSCURL - CSC Process User Release command'
CSCURL   START X'Ø1EA7Ø'
         PRINT NOGEN
         CSCHDR                          User Release command
*
* Process RELEASE command
*
         USING UIDSECT,R8                UID (user) Block
         USING CCHSECT,R7                CCH (cache) Block
         SPACE
         LA    RØ,1                      Default is RELEASE 1 1
         ST    RØ,URLREL1                First line to release
         ST    RØ,URLREL2                Last line to release
         SR    RØ,RØ                     Do not search any table
         GO    CSCSCN                    Get first operand
         BNZ   URL1ØØ                    Nothing, process with defaults
         GO    CSCSCNVN                  Check if numeric
         BNZ   URL7ØØ                    No, bad news...
         SR    R3,R3                     Required by next IC
         IC    R3,UIDSCRL                Load number of screen lines
         LTR   R2,R2                     Check operand
         BZ    URL8ØØ                    It is zero, not enough
         CR    R2,R3                     Check with number of lines
         BH    URL8ØØ                    Too much, also bad
         ST    R2,URLREL1                Now assume RELEASE n n
         ST    R2,URLREL2
         SR    RØ,RØ                     Do not search any table
         GO    CSCSCN                    Get second operand
         BNZ   URL1ØØ                    Nothing, process RELEASE n n
         GO    CSCSCNVN                  Check for numeric
         BNZ   URL7ØØ                    No, bad news again...
         SR    R3,R3                     Required by next IC
         IC    R3,UIDSCRL                Load number of screen lines
         ST    R2,URLREL2                Now we have RELEASE n1 n2
         C     R2,URLREL1                Compare n1 with n2
         BL    URL8ØØ                    n2 < n1, that's magic or error
         CR    R2,R3                     Check with number of lines
         BH    URL8ØØ                    Too much, no magic, just bad
         SR    RØ,RØ                     Do not search any table
         GO    CSCSCN                    Anything unexpected?
         BZ    URL81Ø                    Yes, that was really unexpected
URL1ØØ   SR    R1,R1                     Zero line counter
         L     R7,UIDBUFF1               Start with first screen line
URL11Ø   LTR   R7,R7                     Is it the last?
```

```
         BZ    URL300                 Yes, all user screen was checked
         TM    CCHOPTS,CCHHOLD        Is this message on Hold?
         BZ    URL200
         LA    R1,1(,R1)              Yes, count it
         C     R1,URLREL1             Message to release?
         BL    URL200                 No, keep going
         SR    R0,R0
         L     R1,PFXPTR              Address Prefix table
         USING PFXSECT,R2
URL120   LTR   R2,R1                  Check for End-Of-Table
         BZ    URL130                 Found it, release message
         L     R1,PFXFWD              Address next entry
         CLC   CCHUSER,PFXUSER        Compare user-ids
         BNE   URL120                 Not this one, check next
         IC    R0,PFXCLASS            Get class from Prefix table
         O     R0,UIDCLASS            Check against user classes
         C     R0,UIDCLASS
         BE    URL130                 Good, release message
         LA    R2,CCHUSER             Address user-id from message
         MSG   0360,USER              User not authorized
         B     URL140                 Keep going
         DROP  R2
         SPACE
URL130   BAS   R14,RELEASE            Yes, release this message
         OI    UIDOPT1,UIDRLSE        Remember to rebuild the screen
URL140   L     R1,URLREL1             Get number of message released
         LA    R0,1(,R1)              Increment
         C     R0,URLREL2             Check with last line to release
         BH    URL400                 All done, refresh user screens
         ST    R0,URLREL1             Store next line to release
URL200   L     R7,CCHFWD              Address next screen line
         B     URL110
         SPACE
URL300   L     R2,URLREL1             At least one message not found
         MSG   0361,USER              Tell the user about it
URL400   TM    UIDOPT1,UIDRLSE        Any message released?
         BZ    URL500
         NI    UIDOPT1,X'FF'-UIDRLSE  Yes, reset option
         GO    CSCUSCRB               Rebuild user screen
         SPACE
URL500   BAS   R14,REFRESH            Update users in Refresh mode
         B     URL900                 All done, return
         SPACE
URL700   MSG   0311,USER              Invalid non numeric operand
         B     URL900
         SPACE
URL800   L     R4,URLREL1             Value out of range
         MSG   0362,USER
         B     URL900
         SPACE
URL810   MSG   0312,USER              Unexpected operand
```

```
*        B     URL900
         SPACE
URL900   BACK                          Return
         SPACE 3
*
* Release message
*
*        Input R7 addresses message to release (cache image)
*
CSCURLPR RELOC                         Release (External call)
         BAS   R14,RELEASE             Perform Release
         BACK                          Go back to caller
         SPACE
RELEASE  EQU   *
         ST    R14,RELSV14
         STM   R6,R9,RELSAVE
         L     R0,CCHRECNO             Get record number to release
         SR    R1,R1
         L     R2,HLDPTR               Address first message on Hold
REL100   C     R0,CCHRECNO-CCHSECT(,R2) Check record number
         BNE   REL200
         CLC   CCHUSER,CCHUSER-CCHSECT(R2)    user id
         BNE   REL200
         CLC   CCHDATE,CCHDATE-CCHSECT(R2)    date
         BNE   REL200
         CLC   CCHTIME,CCHTIME-CCHSECT(R2)    time
         BE    REL300                  Must be the one to release
REL200   LR    R1,R2                   Save address of previous entry
         L     R2,CCHFWD-CCHSECT(,R2)  Address next entry
         B     REL100
         SPACE
REL300   L     R0,CCHFWD-CCHSECT(,R2)  Address entry to follow
         LTR   R1,R1                   Is message to release the first?
         BNZ   REL310
         ST    R0,HLDPTR               Yes, change list pointer
         B     REL320
         SPACE
REL310   ST    R0,CCHFWD-CCHSECT(,R1)  Chain previous with next
REL320   LTR   R0,R0                   Is message to release the last?
         BNZ   REL330
         ST    R1,HLDLAST              Yes, update last entry pointer
REL330   LR    R7,R2                   Address entry to release
         NI    CCHOPTS,X'FF'-CCHHOLD   Reset Hold option
         LINK  PREFIX                  Restore default attributes
         L     R0,CCHRECNO             Record number to release
         L     R2,CACHEPTR             Current record from cache
REL400   L     R2,CCHFWD-CCHSECT(,R2)  Start with first (oldest)
         CLC   CCHDATE,CCHDATE-CCHSECT(R2) Is record still in cache
         BL    REL500                  No...
         BH    REL400                  Maybe...
         CLC   CCHTIME,CCHTIME-CCHSECT(R2)
         BL    REL500
```

```
         BH    REL400
         C     R0,CCHRECNO-CCHSECT(,R2)
         BNE   REL400
         NI    CCHOPTS-CCHSECT(R2),X'FF'-CCHHOLD Found cache record
         MVC   CCHATTR-CCHSECT(1,R2),CCHATTR Reset attributes
REL500   L     R8,SSSPTR               Now, let's check the world
         LTR   R8,R8                   Anybody home?
         BZ    REL900                  No, there are no active sessions
         SR    R0,R0
         L     R1,CCHRECNO             Record number we are releasing
REL600   L     R2,UIDFREE1             Check Free List first
REL610   C     R1,CCHRECNO-CCHSECT(,R2) Check only record number
         BNE   REL620
         ST    R0,CCHRECNO-CCHSECT(,R2) Zero record number, invalidate
REL620   L     R2,CCHFWD-CCHSECT(,R2)  Search all table
         LTR   R2,R2
         BNZ   REL610
         L     R2,UIDBUFF1             Now check the user buffer
REL700   C     R1,CCHRECNO-CCHSECT(,R2) Check record number
         BNE   REL710
         CLC   CCHUSER,CCHUSER-CCHSECT(R2)    user id
         BNE   REL710
         CLC   CCHDATE,CCHDATE-CCHSECT(R2)    date
         BNE   REL710
         CLC   CCHTIME,CCHTIME-CCHSECT(R2)    time
         BNE   REL710                  Found it
         NI    CCHOPTS-CCHSECT(R2),X'FF'-CCHHOLD Reset option
         MVC   CCHATTR-CCHSECT(1,R2),CCHATTR Restore default attributes
         TM    UIDOPT2,UIDAUTO         If user is in Refresh mode
         BZ    REL800
         OI    UIDOPT1,UIDRLSE         Remember to update the screen
         B     REL800
         SPACE
REL710   L     R2,CCHFWD-CCHSECT(,R2)  Check all until you find one
         LTR   R2,R2
         BNZ   REL700
REL800   L     R8,UIDFWD               Check all sessions
         LTR   R8,R8
         BNZ   REL600
REL900   LA    R0,CCHSIZE              Now prepare to release storage
         LR    R1,R7                   Load length and address
         LINK  RELEASE                 Release the storage
         LM    R6,R9,RELSAVE           Restore work registers
         L     R14,RELSV14
         BR    R14
         SPACE 3
*
* Update screens for users in refresh mode
*
CSCURLRF RELOC                         Refresh (External call)
         BAS   R14,REFRESH             Perform Refresh
         BACK                          Go back to caller
```

```
         SPACE
REFRESH  EQU   *
         ST    R14,REFSV14
         ST    R8,URLUID               Save address of UID block
         L     R8,SSSPTR               Now check all active sessions
         LTR   R8,R8                   Anybody home?
         BZ    REF400                  No, not a single session
         SPACE
REF100   TM    UIDOPT1,UIDRLSE         Anything released?
         BZ    REF300                  No, check next session
         NI    UIDOPT1,X'FF'-UIDRLSE   Yes, reset option
         GO    CSCUIN                  Refresh user screen
         TM    UIDOPT1,UIDCONN         Is user connected?
         BO    REF300                  Yes, forget it for now
         TM    UIDOPT1,UIDRMTE         Is user remote?
         BZ    REF200                  Yes, send data back
         GO    CSCUSADP                Send data back to user
         B     REF300
         SPACE
REF200   GO    CSCBLD                  Build data stream
         LINK  SEND                    Send it
REF300   L     R8,UIDFWD               Address next session
         LTR   R8,R8                   Make sure there is one
         BNZ   REF100                  Yes, at least one more
REF400   L     R8,URLUID               Restore address of UID block
         L     R14,REFSV14
         BR    R14
         SPACE 3
RELSV14  DS    F                       Save area for RELEASE R14
RELSAVE  DS    4F                                        R6-R9
REFSV14  DS    F                                REFRESH
URLREL1  DS    F                       First line to release
URLREL2  DS    F                       Last line to release
URLUID   DS    F                       Address of current UID block
         SPACE 3
         CSCDATA
         CSCDS (CCH,UID,PFX)
         REGEQU
         END
```

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

Fernando Duarte
Analyst (Canada)

IBM has announced two Entry Enterprise Server Offerings (ESO) for VM and VSE.

ESOs represent a packaged solution comprising the latest System/390 hardware and software, together with the services to integrate and exploit existing applications and peripherals.

For further information contact your local IBM representative.

* * *

Sterling Software has announced Version 3.0 of its VM:Webgateway Web-to-host software for using legacy applications from a Web browser while maintaining end-to-end security. Users can Web-enable and Web-enhance all existing mainframe applications on VM, OS/390, MVS, and VSE and include full-screen applications.

It uses Secure Sockets Layer technology to encrypt data transmitted between Web browsers and the mainframe and it uses client and server certificates that authenticate Web browser users.

Version 3.0 has new support for multi-tier security standards, and trusted third-party Certificate Authorities, such as VeriSign, will soon offer standard, digital certificates that use multi-tier certificate chaining for additional security. This will enable VM:Webgateway users to implement the new multi-tier encryption technology.

Version 3.0 uses 20% less CPU resources and it supports HTTP 1.1 for persistent connections.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, Sterling Court, Eastworth Road, Chertsey, Surrey, KT16 8DF, UK.
Tel: (01932) 587000.
URL: http://www.vm.sterling.com.

* * *