152

VM update

April 1999

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 £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

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*.

Editing two files in parallel

GENERAL DESCRIPTION

On certain occasions, especially when a comparison is required, you may need to edit more than one file at a time.

The first procedure presented here shows two files side by side, either vertically or horizontally, and sets several function keys for working simultaneously in both files. The syntax is:

```
XZWO fn1 ft1 fm1 fn2 ft2 fm2 <(opt>
```

where:

- ‘fn1 ft1 fm1’ is the first file to edit.
- ‘fn2 ft2 fm2’ is the second file to edit.
- ‘opt’ can be either ‘V’ for vertical display or ‘H’ for horizontal display, where ‘H’ is the default setting.

The second procedure allows you to edit the same file simultaneously in EBCDIC and HEX. The syntax is:

```
XH fn ft <fm>
```

where ‘fn ft fm’ is the file to edit in EBCDIC and HEX.

XZWO EXEC

```
/*********************************************************************/
/* Editing two files simultaneously (vertically or horizontally)     */
/*********************************************************************/
/*********************************************************************/
/* Call:   XZWO    fn1 ft1 fm1 fn2 ft2 fm2 <(opt>                    */
/*                 fn1 ft1 fm1        : 1st file                     */
/*                 fn2 ft2 fm2        : 2nd file                     */
/*                 opt = V            : vertically                   */
/*                       H            : horizontally (default)       */
/*********************************************************************/
trace off
parse upper arg fn1 ft1 fm1 fn2 ft2 fm2 '(' opt
if fn1 = '?' then signal hilfe
if opt = '' then opt = 'H'
```

```
queue 'SCREEN 2' opt
queue 'SET PF15 MACRO XHX 15'
queue 'SET PF16 MACRO XHX 16'
queue 'SET PF19 MACRO XHX 19'
queue 'SET PF2Ø MACRO XHX 2Ø'
queue 'SOS TABCMDF'
queue 'XEDIT' fn2 ft2 fm2
queue 'SET PF15 MACRO XHX 15'
queue 'SET PF16 MACRO XHX 16'
queue 'SET PF19 MACRO XHX 19'
queue 'SET PF2Ø MACRO XHX 2Ø'
queue 'SOS TABCMDF'
queue 'MSG PF15/16 resp. PF19/2Ø were set for simultaneous paging'
'XEDIT' fn1 ft1 fm1
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
hilfe:
'VMFCLEAR'
address cms 'type xzwo    exec * 1 1Ø'
```

XH EXEC

```
/*********************************************************************/
/* Editing a file in EBCDIC and HEX simultaneously                   */
/*********************************************************************/
/* The same file is shown to the left in EBCDIC and to the right in  */
/* HEX                                                               */
/*********************************************************************/
/* Call:   XH      fn ft <fm>                                        */
/*                                 : as with XEDIT                   */
/*********************************************************************/
trace off
parse upper arg fn ft fm .
if fn = '?' then signal hilfe
queue 'SCREEN 2 V'
queue 'SOS TABCMDF'
queue 'VERIFY HEX 1 8Ø'
queue 'SET IMAGE OFF'
queue 'SOS TABCMDF'
do m = 1 to 8
   queue 'SET PF'm 'MACRO XHX' m
end
queue 'MSG PF1 until PF8 were set'
'XEDIT' fn ft fm
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
```

```
hilfe:
'VMFCLEAR'
address cms 'type xm      exec * 1 8'
```

XHX XEDIT

```
/*********************************************************************/
/* XEDIT MACRO for HX and XZWO (setting PF keys)                     */
/*********************************************************************/
parse upper arg pf
select
   when pf = '1' then do
        'N 2Ø'
        'SOS TABCMDF'
        'N 2Ø'
        'SOS TABCMDF'
        end
   when pf = '2' then do
        'U 2Ø'
        'SOS TABCMDF'
        'U 2Ø'
        'SOS TABCMDF'
        end
   when pf = '3' then do
        'N 8'
        'SOS TABCMDF'
        'N 8'
        'SOS TABCMDF'
        end
   when pf = '4' then do
        'U 8'
        'SOS TABCMDF'
        'U 8'
        'SOS TABCMDF'
        end
   when pf = '5' then do
        'RIGHT 9'
        'SOS TABCMDF'
        'RIGHT 9'
        'SOS TABCMDF'
        end
   when pf = '6' then do
        'LEFT  9'
        'SOS TABCMDF'
        'LEFT  9'
        'SOS TABCMDF'
        end
   when pf = '7' then do
        'U 2Ø'
        'SOS TABCMDF'
```

```
          'U 2Ø'
          'SOS TABCMDF'
          end
     when pf = '8' then do
          'N 2Ø'
          'SOS TABCMDF'
          'N 2Ø'
          'SOS TABCMDF'
          end
     WHEN PF = '15' THEN DO
          'N 8'
          'SOS TABCMDF'
          'N 8'
          'SOS TABCMDF'
          end
     WHEN PF = '16' THEN DO
          'U 8'
          'SOS TABCMDF'
          'U 8'
          'SOS TABCMDF'
          end
     WHEN PF = '17' THEN DO
          'RIGHT 9'
          'SOS TABCMDF'
          'RIGHT 9'
          'SOS TABCMDF'
          end
     WHEN PF = '18' THEN DO
          'LEFT 9'
          'SOS TABCMDF'
          'LEFT 9'
          'SOS TABCMDF'
          end
     WHEN PF = '19' THEN DO
          'U 2Ø'
          'SOS TABCMDF'
          'U 2Ø'
          'SOS TABCMDF'
          end
     WHEN PF = '2Ø' THEN DO
          'N 2Ø'
          'SOS TABCMDF'
          'N 2Ø'
          'SOS TABCMDF'
          end
     OTHERWISE NOP
end
```

Dr Reinhard Meyer (Germany) © Xephon 1999

Mouse-clickable XEDIT enhancements – part 2

This month, continuing the Mouse on the mainframe *series of articles on the manipulation of System/390 applications with a PC or workstation mouse, we conclude the article on writing mouse-clickable XEDIT enhancements.*

KEYWIN menus rest on the bottom of the logical screen and open upwards until all menu items are showing or the top of the logical screen is visible. Menus that are too long to be displayed in their entirety can be scrolled by pressing PF7/PF8 or by clicking on the BACK/FORW menu controls. Most practically, KEYWIN menus can be assigned to PF keys, as follows:

```
/* PROFILE XEDIT assigning KEYWIN macros to PF keys */
.
.
.
'SET PF6 MACRO KEYWIN 6 XCMDS'
.
.
.
'RESERVE -4 T N  1=Help ......................... 6=XCmds'
.
.
.
'SET ENTER BEFORE MACRO HOTKEYS'
.
.
.
```

Now:

- PF6 is assigned to invoke the KEYWIN XEDIT macro, which will display the contents of file XCMDS KEYWIN in a CMS window in position ‘6’ (rightmost) of the current XEDIT screen.
- The fourth line from the bottom of the XEDIT screen will contain a reserved line for PF keys 1 to 6.
- Pressing the ENTER key (or clicking with the mouse) will invoke the HOTKEYS XEDIT macro, which will determine what to do. If the string ‘6=XCmds’ on reserved line -4 is clicked, then the KEYWIN macro will run and display a list of XEDIT subcommands, as shown in Figure 5 (see *VMUpdate*, Issue151).

Any, or all, PF keys can be assigned to invoke the KEYWIN XEDIT macro. Each KEYWIN menu can contain an unlimited number of commands and subcommands (in practice, one or two dozen commands makes sense). So in using the KEYWIN macro and the other techniques outlined in this section, 12 PF keys and two reserved lines can serve to access several hundred subcommands, commands, and EXECs.

KEYWIN also supports the following command syntax, which includes variables to represent the file-id:

```
PRINT &fn &ft &fm
```

where &fn, &ft, and &fm are dynamically replaced with the current file's filename, filetype, and filemode.

When a menu item contains any of these variables, KEYWIN analyses the command and substitutes the necessary file-id values before issuing the command. File-id variable substitution greatly increases the variety of useful commands and EXECs that can be included in KEYWIN menus.

The KEYWIN XEDIT macro follows:

```
/* KEYWIN XEDIT - Adding Pop-Up Menus to the Xedit Screen          */
winname='KEYWIN'                                /* name the window    */
Address 'COMMAND'                               /* address commands   */

/* Process arguments                                                  */
arglist=Arg(1)                                  /* get arg string     */
nargs=Words(arglist)                            /* get number of args */
Parse Upper Var arglist ,                       /* get window position*/
   winpos kwfn kwft kwfm .                      /*   & menu file name */
If (nargs < 2 | Datatype(winpos)¬='NUM' ,       /* detect bad call    */
    | winpos<1 | winpos>12)
   Then
      Do
         Address 'XEDIT' 'MSG' ,                /* queue error message*/
            Left(winname':',9) ,
            'Selected PF key is' ,
            'incorrectly defined;' ,
            'see HELP PETS' winname'.'
         Exit(97)                               /* exit with rc=97    */
         End
If (Length(Strip(kwft))=Ø)                      /* no filetype?       */
   Then kwft='KEYWIN'                           /*   then ft="KEYWIN" */
If (Length(Strip(kwfm))=Ø)                      /* no filemode?       */
   Then kwfm='*'                                /*   the fm="*"       */
```

```
'ESTATE' kwfn kwft kwfm                          /* check for menu      */
If (rc¬=Ø)                                       /* menu missing?       */
   Then
      Do
         Address 'XEDIT' 'MSG' ,                 /* queue error message*/
            Left(winname':',9) ,
            'Unable to locate' ,
            '"'kwfn kwft kwfm'"' ,
            'specified on selected PF key.'
         Exit(98)                                /* exit with rc=98     */
         End

/* Initialize virtual screen and window                                 */

borcol='W'                                      /* set border colour   */
clkcol='Y'                                      /* set help text colour*/
txtcol='G'                                      /* set menu item colour*/
f=GETCHOICES()                                  /* get menu items      */
f=DEFINEWIN()                                   /* define scrn, window */
clines=lines-5                                  /* set "scroll" value  */
topchoice=1                                     /* init top menu item  */
botchoice=Min(choices.Ø,topchoice+clines)       /* init last menu item */

/* Display window, process request(s)                                   */

Do loop=1 By 1                                   /* infinite loop       */

   /* Clear and requeue menu items to virtual screen                    */

   Do j=1 To Min(choices.Ø,lines-4)              /* clear all lines of */
      'VSCREEN WRITE' winname j ,                /*   the virtual      */
         '1 11 (PR FIELD' ' '                    /*   screen           */
      End
   k=Ø                                           /* init line counter  */
   Do j=topchoice To botchoice                   /* write menu items   */
      k=k+1                                      /* increment counter  */
      Parse Var choices.j type program ,         /* parse menu item    */
         "'"description                          /*   description      */
      'VSCREEN WRITE' winname k '1 11' ,         /* queue menu item    */
         '(NOPR' txtcol 'FIELD',                 /*   descripts to the */
         Left(Strip(description),9)              /*   virtual screen   */
      End

   /* Update the 327Ø and receive mouse click                           */

   'VSCREEN CURSOR' winname '1 2'                /* cursor on top item */
   'WINDOW POP' winname                          /* menu in front/top  */
   'VSCREEN WAITREAD' winname                    /* update 327Ø, get   */
                                                 /*   selection info   */
   /* Retrieve keystroke and cursor location                          */

   Parse Var waitread.2 . vline vcol area       /* get cursor position*/
   choice=vline+topchoice-1                     /* get menu choice #  */
   keystroke=Strip(waitread.1)                  /* identify keystroke */
   xcmd=''                                      /* reset cmd string   */

   /* Evaluate where the menu was clicked                             */

   Select;
      When (vline=-1 & vcol=-1)                  /* close window?     */
         Then keystroke='PFKEY  3'
      When (vline=-2 & vcol<6 & area='RESERVED') /* scroll backward?  */
         Then keystroke='PFKEY  7'
      When (vline=-2 & vcol=6 & area='RESERVED') /* indeterminate?    */
         Then keystroke='NULL'
      When (vline=-2 & vcol>6 & area='RESERVED') /* quit?             */
         Then keystroke='PFKEY  3'
      When (vline=-1 & vcol<6 & area='RESERVED') /* scroll forward?   */
         Then keystroke='PFKEY  8'
      When (vline=-1 & vcol=6 & area='RESERVED') /* indeterminate?    */
         Then keystroke='NULL'
      When (vline=-1 & vcol>6 & area='RESERVED') /* edit menu file?   */
         Then keystroke='PFKEY 11'
      Otherwise NOP                              /* else no change    */
      End

   /* Process function                                                */

   Select;
      When (keystroke='NULL')                    /* no function?      */
         Then NOP
      When (keystroke='PFKEY  1')                /* PF1 pressed?      */
         Then Address 'CMS' 'HELP PETS KEYWIN'   /*  then display help*/
      When (keystroke='PFKEY  3')                /* PF3 pressed?      */
         Then Leave loop                         /*   then end loop   */
      When (keystroke='PFKEY  7')                /* PF7 pressed?      */
         Then
            If (topchoice=1)                     /* if at menu top,   */
               Then                              /*    scroll to end  */
                  Do
                     topchoice=choices.Ø
                     botchoice=choices.Ø
                     End
               Else                              /* else              */
                  Do                             /*   scroll backward */
                     topchoice= ,                /* set top menu item */
                        Max(topchoice-clines,1)
                     botchoice= ,                /* set end menu item */
                        Min(topchoice+clines, ,
                        choices.Ø)
                     End
```

```
When (keystroke='PFKEY  8')              /* PF8 pressed?      */
   Then
      If (topchoice=choices.Ø)           /* if at menu end,   */
         Then                            /*    scroll to top  */
            Do
               topchoice=1               /* set top menu item */
               botchoice= ,              /* set end menu item */
                  Min(topchoice+clines, ,
                  choices.Ø)
               End
         Else                            /* else              */
            Do                           /*   scroll forward  */
               topchoice=Max(Min( ,      /* set top menu item */
                      topchoice+clines, ,
                         choices.Ø),1)
               botchoice=Min(topchoice+ ,  /* set end menu item */
                         clines,choices.Ø)
               End
When (keystroke='PFKEY 11')              /* PF11 pressed?     */
   Then
      Do
         'XEDIT' kwfn kwft kwfm         /* edit menu file     */
         f=GETCHOICES()                 /* refresh choices.   */
         Address 'XEDIT' 'REFRESH'      /* refresh screen     */
         'WINDOW POP' winname           /* put menu on top    */
         End
When (keystroke='PFKEY 12')             /* PF12 pressed?      */
   Then Leave loop                      /*   then quit        */
When (Left(keystroke,5)='PFKEY')        /* another PF key?    */
   Then NOP                             /*   then do nothing  */
When (Left(keystroke,5)='ENTER' ,       /* menu item selected?*/
      & area='DATA')
   Then
      Do
         Parse Upper Var ,              /* parse menu item    */
            choices.choice ,
            command"'"description"'"
         command=Strip(command)
         Address 'XEDIT' 'EXT/FT/FN/FM'  /* get fn, ft, fm     */
         Do sloop=1 By 1                /* fix all variables  */
            posfn=Pos('&FN',command)    /* &fn variable?      */
            posft=Pos('&FT',command)    /* &ft variable?      */
            posfm=Pos('&FM',command)    /* &fm variable?      */
            Select;
               When (posfn>Ø)           /* substitute fname   */
                  Then command= ,
                     Left(command,Max(posfn-1,Ø)) ,
                     ||fname.1|| ,
                     Substr(command,posfn+3)
               When (posft>Ø)           /* substitute ftype   */
                  Then command= ,
                           Left(command,posft-1) ,
                           ||ftype.1|| ,
                           Substr(command,posft+3)
                     When (posfm>Ø)             /* substitute fmode   */
                        Then command= ,
                           Left(command,posfm-1) ,
                           ||fmode.1||,
                           Substr(command,posfm+3)
                     Otherwise Leave sloop      /* exit if done       */
                     End
                  End

               /* Process command                                     */

               Select;
                  When (Length(command)=Ø)      /* if blank, pass     */
                     Then NOP
                  When (Pos('QUIT',' 'command)>Ø,
                     | Pos(' QQ',' 'command)>Ø,     /*      one       */
                     | Pos(' FILE',' 'command)>Ø,   /*      of        */
                     | Pos(' FF',' 'command)>Ø,     /*      these     */
                     | Pos(' CANCEL',' 'command)>Ø, /*      commands  */
                     | Pos(' KEYWIN',' 'command)>Ø, /*      ?         */
                     | Pos(' XAP',' 'command)>Ø)
                     Then                       /* if yes, then       */
                        Do
                           xcmd=command         /*    set cmd         */
                           Leave loop           /*    exit            */
                           End
                  Otherwise
                     Do
                        Address 'XEDIT'         /* commands to Xedit  */
                        command                 /* issue command      */
                        Do sloop=1 By 1         /* infinite loop      */
                           If (Queued()=Ø)      /* anything queued?   */
                              Then Leave sloop  /* no, leave          */
                           Pull stkcmd          /* yes, pull command  */
                           stkcmd               /* issue command      */
                           End
                        'REFRESH'               /* update 327Ø screen */
                        Address 'COMMAND'       /* commands revert    */
                        End
                  End
            End
      Otherwise NOP                             /* unknown keystrokes */
      End
   End

/* Exit processing                                                    */
```

```
KEYWINEND:
Address 'COMMAND'                                /* address commands   */
'SET CMSTYPE HT'                                 /* hide messages      */
'WINDOW DELETE' winname                          /* delete window      */
'VSCREEN DELETE' winname                         /* delete v. screen   */
'SET CMSTYPE RT'                                 /* show messages      */
If (xcmd='')                                     /* if no command,     */
   Then NOP                                      /*    do nothing      */
   Else Push xcmd                                /* else queue command */
Exit(Ø)                                          /* return to Xedit    */

/* Read commands from disk file into stem variable CHOICES.            */

GETCHOICES:
choices.='';choices.Ø=Ø                          /* init stem variable */
Address 'COMMAND' 'EXECIO * DISKR' ,             /* read file          */
   kwfn kwft kwfm '(STEM CHOICES. FINIS'
If (choices.Ø=Ø)                                 /* if no commands     */
   Then
      Do
         Address 'XEDIT' 'MSG' ,                 /* issue error msg    */
            Left(winname':',9) ,
            'Unable to read' ,
            'KEYWIN "'kwfn kwft kwfm'".'
         Exit(99)                                /* exit, rc=99        */
         End
Return('')

/* Define menu virtual screen and window                               */

DEFINEWIN:
f=CMSTYPE('HT')                                  /* hide messages      */
'QUERY DISPLAY (LIFO';Parse Pull . slines .      /* get # 327Ø lines   */
Address 'XEDIT' 'EXT/LSCREEN/'                   /* logical screen size*/
lines=lscreen.1
cols=lscreen.2
line1=lscreen.3
col1=lscreen.4
bline=line1+lines-1
winrows=Min(lines-2,2+choices.Ø)                 /* calc size of window*/
'QUERY VSCREEN' winname                          /* v. screen exists?  */
If (rc¬=Ø)                                       /* if v. scrn missing,*/
   Then 'VSCREEN DEFINE' ,                       /*    then define v.  */
      winname winrows '11 Ø 2'                   /*    screen          */
'QUERY WINDOW' winname                           /* window exists?     */
If (rc=Ø)                                        /* if window exists,  */
   Then 'WINDOW DELETE' winname                  /*   then delete win  */
pl=slines-bline+2                                /* line location      */
pl=-pl
pc=Word('3 16 29 42 55 66 3 16 29 42 55' ,       /* column location    */
   '66',winpos)+col1-1
If (pc>cols+11) Then pc=3+col1-1              /* limit col location */
If (cols<12 | winrows<6)                      /* if window is too   */
   Then                                       /*    small           */
      Do
         Address 'XEDIT' 'MSG' ,              /* issue error msg    */
            Left(winname':',9) ,
            'Cannot create a' ,
            'window.'
         f=CMSTYPE()                          /* reset message mode */
         Signal KEYWINEND                     /* go to exit process */
         End
'WINDOW DEFINE' winname winrows 12 pl pc ,    /* define window      */
   '(POP TOP'
'WINDOW SHOW' winname 'ON' winname '1 1'      /* connect win to scrn*/
'SET BORDER' winname 'ON ('borcol             /* set border colour  */
'VSCREEN WRITE' winname '-2 1 11 (RES PR' ,   /* queue line -2 help */
   clkcol 'FIELD BACK QUIT'
'VSCREEN WRITE' winname '-1 1 11 (RES PR' ,   /* queue line -1 help */
   clkcol 'FIELD FORW EDIT'
f=CMSTYPE()                                   /* reset message mode */
Return('')

/* Handle CMS Message Typing                                        */

CMSTYPE:
Procedure Expose $cmstype                     /* retain cmstype val */
Address 'COMMAND'                             /* address commands   */
type=Strip(Translate(Left(Arg(1),2)))         /* get request        */
If (type¬='' & Wordpos(type,'HT RT')>Ø)       /* if specific request*/
   Then
      Do
         'QUERY CMSTYPE (LIFO'                /* query cmstype      */
         Parse Pull . . $cmstype .            /* get current cmstype*/
         'SET CMSTYPE' type                   /* set new cmstype    */
         End
   Else If (Wordpos($cmstype,'HT RT')>Ø)      /* else if previous,  */
         Then Address 'COMMAND' ,             /* reset cmstype      */
               'SET CMSTYPE' $cmstype
Return('')                                    /* return             */
```

THE PETPROF XEDIT MACRO

KEYWIN command menus can contain any XEDIT subcommand, CMS or CP command, macro, or EXEC. Commands such as 'SAVE' and 'PRINT &fn &ft &fm' may be appropriate for any file opened to XEDIT. However, a command like 'SCRIPT &fn' is appropriate only for files containing Script commands. If a user routinely edits different kinds of files (eg Assembler, EXECs, HTML, Script) it may be appropriate to create several different 'profile' macros, each invoking KEYWIN menus appropriate to a different filetype.

The PETPROF XEDIT macro can be used to invoke an XEDIT macro appropriate to the filetype of the active file. The mapping of filetype to macro is contained in a file named DEFAULT PETPROF. A sample DEFAULT PETPROF follows:

```
*        PROFILEA
ASSEMBLE PROFILEB
SCRIPT   PROFILED
```

The left column contains a list of filetypes, with '*' signifying any filetype not otherwise listed. The right column contains a list of XEDIT macros containing XEDIT initialization subcommands. The logic of PETPROF is as follows:

- PETPROF determines the filetype of the active file.
- PETPROF determines whether one of two files exists: 'userid PETPROF' or 'DEFAULT PETPROF'; if so, PETPROF inspects the file to determine whether a macro has been specified for the filetype.
- If no macro has been specified for the filetype, PETPROF inspects the macro mapping file to determine whether a default macro has been specified.
- If an appropriate macro can be identified, PETPROF determines whether that macro exists, and, if it exists, that macro is invoked.

In summary, to enable automatic selection of XEDIT initialization subcommands tailored to files by filetype, one must do the following:

- Create one or more XEDIT macros containing initialization subcommands, including calls to the HOTKEYS and KEYWIN macros, if desired; each macro should be tailored to a specific filetype.
- Create a file named 'userid PETPROF' or 'DEFAULT PETPROF', which maps filetypes to macros.

- Ensure access to the PETPROF XEDIT macro.
- Create a file named PROFILE XEDIT containing the following lines:

```
/* PROFILE XEDIT */
'MACRO PETPROF'
Exit(Ø)
```

The PETPROF XEDIT macro follows:

```
/* PETPROF XEDIT - Invoke an XEDIT macro according to filetype        */
Address 'XEDIT'                                   /* address commands   */
'EXT/FT/'                                         /* get filetype       */
profile=GETPROF(ftype.1)                          /* recommended macro? */
If (profile='') Then profile=GETPROF('*')         /* no, get default    */
Select;
   When (profile='') Then NOP                     /* no macro, zip      */
   When (profile='PROFILE') Then profile=''       /* if standard, null  */
   When (PESTATE(profile 'XEDIT *')=Ø) Then NOP /* macro found?       */
   Otherwise profile=''                           /* hmmm....           */
   End
If (profile¬='') Then 'MACRO' profile             /* issue macro        */
Exit(Ø)                                           /* return             */

/* Look for a recommended XEDIT macro, according to filetype          */

GETPROF:
Address 'COMMAND'                                 /* address commands  */

/* Look for a profile-to-filetype mapping file.                       */

Select;
   When (PESTATE(Userid() 'PETPROF *')=Ø)         /* user-id PETPROF?   */
      Then petprofid=Userid() 'PETPROF'
   When (PESTATE('DEFAULT PETPROF *')=Ø)          /* DEFAULT PETPROF?   */
      Then petprofid='DEFAULT PETPROF'
   Otherwise petprofid=''                         /* hmmmm....          */
   End

$$prof$$='$$PROF$$'                               /* set profile dummy  */
If (petprofid='')                                 /* if no mapping file,*/
   Then NOP                                       /*    do nothing      */
   Else
      Do
         If (Arg()=Ø)                             /* filetype supplied? */
            Then filetype='*'                     /*    no, set default */
            Else Parse Upper Arg filetype .       /*    yes, get ft     */
         'PIPE <' petprofid ,                     /* scan mapping file  */
            '| STRIP LEADING' ,                   /* (no leading blanks)*/
            '| FIND' filetype'_| TAKE 1' ,        /* for filetype;      */
            '| SPEC WORDS 2-2 1' ,             /* get profile name   */
            '| VAR $$PROF$$ | HOLE'            /* save in variable   */
         If ($$prof$$='$$PROF$$')              /* if no profile,     */
            Then $$prof$$=''                   /*    then clear var  */
         End
Return($$prof$$)                               /* return profile name*/

/* Check for profile-to-filetype mapping file                        */

PESTATE: Procedure
Address 'COMMAND' 'ESTATE' Arg(1)
Return(rc)
```

FURTHER INFORMATION

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

Editor's note: in a future article, the author will discuss adding new functions to XEDIT with alternative XEDIT customization macros.

Richard G Ellis
Director of Computing and Information Systems
University of Connecticut (USA)

A full screen console interface – part 9

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.*

```
BOTTOM   EQU   *                   PFØ5 Bottom
         ST    R14,PFKSV14
         TM    UIDOPT2,UIDAUTO          Is user in Refresh mode
         BZ    BOT1ØØ                   No, do it
         NI    UIDOPT2,X'FF'-UIDAUTO    Reset AUTO Refresh option
         OI    UIDOPT4,UIDBHDR          Remember to refresh Header line
         B     BOT2ØØ                   Refresh the screen, no beeps
         SPACE
BOT1ØØ   GO    CSCWRPGB                Locate bottom line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO             Is record number valid?
         BNE   BOT2ØØ                  No, must be null, TOF, or EOF
         OI    UIDOPT4,UIDBALM         Just sound the alarm
         B     BOT9ØØ
         SPACE
BOT2ØØ   BAS   R14,BOTSCR              Build BOTTOM screen
*        B     BOT9ØØ
         SPACE
BOT9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
CSCUSCBL RELOC                         BOTLINE (external call)
         BAS   R14,BOTLINE
         BACK
         SPACE
BOTSCR   EQU   *
         ST    R14,BOTSV14
         L     R7,UIDBUFF2             Address bottom line
         LINK  DELETE                  Delete it
         LINK  ADDEOFT                 Add EOF as first record
         ST    R7,NEWBOT               Save as new bottom line
         L     R7,UIDBUFF2             Delete the last line
         LINK  DELETE
         GO    CSCRDFLT                Read last record
         BNZ   BOTBLANK                Not found, add blank lines
         SR    R1,R1                   Add as first line
         LINK  ADD
         B     BOTL1ØØ                 Move line to the top
         SPACE
BOTLINE  EQU   *
         ST    R14,BOTSV14
         ST    R7,NEWBOT               Save as new top line
BOTL1ØØ  L     R7,UIDBUFF2             Address bottom line
         C     R7,NEWBOT               Is it the required new bot line
         BE    BOTEND                  Yes, screen completed
         LINK  DELETE                  No, delete top line
         L     R7,UIDBUFF1             Address top line
         GO    CSCRDFPR                Get previous record
         BNZ   BOTBLANK                Not found, add blank lines
         SR    R1,R1                   Add as first line
         LINK  ADD
         B     BOTL1ØØ
         SPACE
BOTBLANK EQU   *
         LINK  ADDTOFT                 Add TOF as first record
BOTB1ØØ  L     R7,UIDBUFF2             Address bottom line
         C     R7,NEWBOT               Is it the EOF message
         BE    BOTEND                  Yes, screen completed
         LINK  DELETE                  No, delete bottom line
         LINK  ADDBLKT                 Add blank line as first record
         B     BOTB1ØØ
         SPACE
BOTEND   EQU   *
         OI    UIDOPT4,UIDBSCR         Option to build user screen
         MVI   CCHLINE2,X'FF'          Make Bottom and Top lines valid
         L     R7,UIDBUFF1                in case WRAP is turned Off
         MVI   CCHLINE2,X'FF'
         TM    UIDOPT3,UIDWRAP         Is WRAP switch On?
         BZ    BOTE9ØØ                 No, done
         GO    CSCWRPBT                Yes, build partial lines
BOTE9ØØ  L     R14,BOTSV14
         BR    R14
         SPACE 3
*
* Process BACKWARD command   (PFØ7, PF19 or BWD)
*
*
BWDCMD   EQU   *                       Backward (input command)
         ST    R14,CMDSV14
         SR    RØ,RØ                   No table to search
         GO    CSCSCN                  Scan parameter
         LA    R2,1                    Default is 1
         BNZ   BWDC1ØØ                 Nothing found, use default
         GO    CSCSCNVN                Validate parameter
         BNZ   BWDC5ØØ                 Not numeric, that's an error
         SR    RØ,RØ                   Do not search any table
         GO    CSCSCN                  Anything left?
         BZ    BWDC6ØØ                 Yes, bad news, only one parm
BWDC1ØØ  LTR   R6,R2                   Copy repetition factor
         BZ    BWDC9ØØ                 It is zero, all done
         GO    CSCWRPGT                Locate top line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO             Is it TOF already?
         BNE   BWDC2ØØ                 No, do the work
         OI    UIDOPT4,UIDBALM         Yes, beep beep
         B     BWDC9ØØ                 Done...
         SPACE
BWDC2ØØ  BAS   R14,BWD                 Go back one screen
         TM    UIDOPT4,UIDBALM         Was alarm set?
         BO    BWDC3ØØ                 Yes, give up, we are at TOF
         BCT   R6,BWDC2ØØ              Go back lots of screens
BWDC3ØØ  NI    UIDOPT4,X'FF'-UIDBALM   We did something, no beeps
BWDC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
BWDC5ØØ  MSG   Ø311,USER               We got an invalid operand
         B     BWDC9ØØ                 That's all
         SPACE
BWDC6ØØ  MSG   Ø312,USER                Too many operands
         B     BWDC9ØØ                  That's all
         SPACE
BWD      EQU   *                   PFØ7 Backward
         ST    R14,PFKSV14
         GO    CSCWRPGT                 Locate top line on screen
         C     R7,UIDBUFF1              Is it the same?
         BE    BWD1ØØ
         SR    R1,R1                    No, add it as top line
         LINK  ADD
         L     R7,UIDBUFF2              Address last line
         LINK  DELETE                   Delete it
         L     R7,UIDBUFF1              Address top line again
BWD1ØØ   SR    RØ,RØ
         C     RØ,CCHRECNO              Is record number valid?
         BE    BWD8ØØ                   No, impossible to do it, beep
         TM    UIDOPT2,UIDAUTO          Is user in Refresh mode
         BZ    BWD2ØØ                   No, do it
         NI    UIDOPT2,X'FF'-UIDAUTO    Reset AUTO Refresh option
         OI    UIDOPT4,UIDBHDR          Remember to refresh Header line
BWD2ØØ   BAS   R14,BOTLINE              Move line to the bottom
         B     BWD9ØØ
         SPACE
BWD8ØØ   OI    UIDOPT4,UIDBALM          Null, TOF or EOF reference, beep
BWD9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
*
* Process FORWARD command   (PFØ8, PF2Ø or FWD)
*
*
FWDCMD   EQU   *                        Forward (input command)
         ST    R14,CMDSV14
         SR    RØ,RØ                    No table to search
         GO    CSCSCN                   Scan parameter
         LA    R2,1                     Default is 1
         BNZ   FWDC1ØØ                  Nothing found, use default
         GO    CSCSCNVN                 Validate parameter
         BNZ   FWDC5ØØ                  Not numeric, that's an error
         SR    RØ,RØ                    Do not search any table
         GO    CSCSCN                   Anything left?
         BZ    FWDC6ØØ                  Yes, bad news, only one parm
FWDC1ØØ  LTR   R6,R2                    Copy repetition factor
         BZ    FWDC9ØØ                  It is zero, all done
         GO    CSCWRPGB                 Locate bottom line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO              Is it EOF already?
         BNE   FWDC2ØØ                  No, do the work
         OI    UIDOPT4,UIDBALM          Yes, beep beep
         B     FWDC9ØØ                 Done...
         SPACE
FWDC2ØØ  BAS   R14,FWD                 Go back one screen
         TM    UIDOPT4,UIDBALM         Was alarm set?
         BO    FWDC3ØØ                 Yes, give up, we are at TOF
         BCT   R6,FWDC2ØØ              Go back lots of screens
FWDC3ØØ  NI    UIDOPT4,X'FF'-UIDBALM
FWDC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
FWDC5ØØ  MSG   Ø311,USER               We got an invalid operand
         B     FWDC9ØØ                 That's all
         SPACE
FWDC6ØØ  MSG   Ø312,USER               Too many operands
         B     FWDC9ØØ                 That's all
         SPACE
FWD      EQU   *                  PFØ8 Forward
         ST    R14,PFKSV14
         GO    CSCWRPGB                Locate bottom line on screen
         C     R7,UIDBUFF2             Is it the same?
         BE    FWD1ØØ
         L     R1,UIDBUFF2             No, add it as bottom line
         LINK  ADD
         L     R7,UIDBUFF1             Address first line
         LINK  DELETE                  Delete it
         L     R7,UIDBUFF2             Address bottom line again
FWD1ØØ   SR    RØ,RØ
         C     RØ,CCHRECNO             Is record number valid?
         BE    FWD8ØØ                  No, impossible to do it, beep
         TM    UIDOPT2,UIDAUTO         Is user in Refresh mode
         BZ    FWD2ØØ                  No, do it
         NI    UIDOPT2,X'FF'-UIDAUTO   Reset AUTO Refresh option
         OI    UIDOPT4,UIDBHDR         Remember to refresh Header line
FWD2ØØ   BAS   R14,TOPLINE             Move line to the top
         B     FWD9ØØ
         SPACE
FWD8ØØ   OI    UIDOPT4,UIDBALM         Null, TOF or EOF reference, beep
FWD9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
*
* Process CURRENT command   (PFØ9 or PF21)
*
*
CURCMD   EQU   *                       Current (input command)
         ST    R14,CMDSV14
         SR    RØ,RØ                   No table to search
         GO    CSCSCN
         BNZ   CURC1ØØ                 Nothing found, that's good
         MSG   Ø312,USER               No parameters allowed
         B     CURC9ØØ
         SPACE
CURC1ØØ  BAS   R14,CURRENT
CURC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
CURRENT  EQU   *                   PFØ9 Current
         ST    R14,PFKSV14
         TM    UIDOPT2,UIDAUTO          Is user in Refresh mode
         BZ    CUR1ØØ                   No, refresh screen
         OI    UIDOPT4,UIDBALM          Yes, beep beep
         B     CUR9ØØ
         SPACE
CUR1ØØ   OI    UIDOPT2,UIDAUTO          Set AUTO Refresh option
         BAS   R14,ADDHDR               Create new Header line
         OI    UIDOPT4,UIDBHDR          Refresh the Header line
         BAS   R14,REBUILD              Refresh the screen
CUR9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
*
* Process LEFT and RIGHT commands (SHIFT omitted)
*
*
SHIFTLE  EQU   *                        Left (Shift omitted)
         ST    R14,CMDSV14
         LA    R4,SHFLEFT               Load shift direction
         B     SHF#GO
         SPACE
SHIFTRI  EQU   *                        Right (Shift omitted)
         ST    R14,CMDSV14
         LA    R4,SHFRIGHT              Load shift direction
         B     SHF#GO
         SPACE
*
* Process SHIFT command   (PF1Ø or PF22 or SHIFT)
*
*        SHF#GO is invoked by SHIFTLE and SHIFTRI (SHIFT omitted)
*
*
SHIFTCMD EQU   *                        Shift (input command)
         ST    R14,CMDSV14
         LA    RØ,SHFTABLE              Address table to search
         GO    CSCSCN                   Scan option
         BNZ   SHFC5ØØ                  Nothing, something is missing
         LTR   R4,R15                   Save and test value
         BZ    SHFC6ØØ                  Not on table, invalid option
SHF#GO   SR    RØ,RØ                    No more tables to search
         GO    CSCSCN                   Get shift value
         BNZ   SHFC5ØØ                  Nothing, more bad news
         GO    CSCSCNVN                  Verify if numeric
         BNZ   SHFC6ØØ                   No, it is not, more bad news
         LR    R5,R2                     Save shift value for now
         SR    RØ,RØ                     No more tables to search
         GO    CSCSCN                    Check for something extra
         BZ    SHFC7ØØ                   Something found, more bad news
         LTR   RØ,R5                     Test shift value
         BZ    SHFC2ØØ                   It is zero, back to column one
         SR    RØ,RØ                     Required by next IC
         IC    RØ,UIDCOL1                Get current first column
         LA    R1,SHFLEFT                Check for left/right shift
         CR    R1,R4                     Compare with left shift
         BNE   SHFC1ØØ                   No good, it must be right shift
         SR    RØ,R5                     Subtract... shift left
         BNM   SHFC2ØØ                   Check if negative
         B     SHFC8ØØ                   It is, no virtual columns, error
         SPACE
SHFC1ØØ  AR    RØ,R5                     Add... shift right
         LA    R1,L'CCHDATA              Compare with maximum
         CR    RØ,R1
         BH    SHFC8ØØ                   Too much, out of range
SHFC2ØØ  STC   RØ,UIDCOL1                Store new offset
         OI    UIDOPT4,UIDBSCR           Remember to rebuild screen DS
         TM    UIDOPT3,UIDWRAP           Is WRAP switch On?
         BZ    SHFC9ØØ
         MSG   Ø32Ø,(USER,NOCMD)         Yes, display warning message
SHFC9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE
SHFC5ØØ  MSG   Ø31Ø,USER                 Missing operands
         B     SHFC9ØØ
         SPACE
SHFC6ØØ  MSG   Ø311,USER                 Invalid operands
         B     SHFC9ØØ
         SPACE
SHFC7ØØ  MSG   Ø312,USER                 Unexpected operands
         B     SHFC9ØØ
         SPACE
SHFC8ØØ  MSG   Ø321,USER                 Shift value too big
         B     SHFC9ØØ
         SPACE
SHIFT    EQU   *                    PF1Ø Shift
         ST    R14,PFKSV14
         CLI   UIDCOL1,Ø                 Is current value zero
         MVI   UIDCOL1,Ø                 Assume it is not
         BNE   SHF1ØØ                    No, we were right
         MVI   UIDCOL1,64                It was zero, make it 64
SHF1ØØ   OI    UIDOPT4,UIDBSCR           Refresh the user screen
         TM    UIDOPT3,UIDWRAP           Is WRAP switch On?
         BZ    SHF9ØØ
         MSG   Ø32Ø,(USER,NOCMD)       Yes, display warning message
SHF9ØØ   L     R14,PFKSV14
         BR    R14
         SPACE 3
*
* Process ENTER key
*
*
REFRESH  EQU   *                 ENTER No function at this moment
*        ST    R14,PFKSV14
*        L     R14,PFKSV14
         BR    R14
         SPACE 3
*
* Process END command   (PFØ3 / PF15 is local to the user)
*
*
END      EQU   *                       End (PFØ3 is local to user)
         TM    UIDOPT1,UIDRMTE         Is user remote?
         BO    DISCONN                 Yes, process as Disconnect
         ST    R14,CMDSV14
         SR    RØ,RØ                   No table to search
         GO    CSCSCN
         BNZ   END1ØØ                  Nothing found, good news
         MSG   Ø312,USER               No parameters allowed
         L     R14,CMDSV14
         BR    R14
         SPACE
END1ØØ   L     RØ,UIDPID               Get PATHID (first two bytes)
         GO    CSCSEV                  Terminate session
         B     USERBYE                 That's all, user is gone
         SPACE 3
*
* Process DISConnect command
*
*
DISCONN  EQU   *                       Disconnect
         ST    R14,CMDSV14
         GO    CSCUSADN
         LTR   R8,R8                   User maybe be gone
         BZ    USERBYE                 Yes it is...
         L     R14,CMDSV14
         BR    R14
         SPACE 3
*
* Process SWAP/SWITCH command
*
*
SWAP     EQU   *                  SWAP command
         ST    R14,CMDSV14
         LA    RØ,SWPTABLE            Table to search
         GO    CSCSCN
         BNZ   SWAP6ØØ                Nothing, we must have at least 1
SWAP1ØØ  LTR   R15,R15                Is parameter valid?
         BZ    SWAP7ØØ                No, error
         LR    R1,R15                 Copy value
         SRL   R1,8                   Get index to option byte
         IC    RØ,UIDOPTS(R1)         Load correct byte
         XR    RØ,R15                 Switch bit
         STC   RØ,UIDOPTS(R1)         Put it back
         B     SWAP8ØØ                Next please...
         SPACE
SWAP6ØØ  MSG   Ø31Ø,USER              No parameters entered
         B     SWAP9ØØ
         SPACE
SWAP7ØØ  MSG   Ø311,USER              Invalid parameter found
SWAP8ØØ  LA    RØ,SWPTABLE            Address table (again)
         GO    CSCSCN
         BZ    SWAP1ØØ                We got something, check it
         OI    UIDOPT4,UIDBHDR        All done, refresh Header line
         NI    UIDOPT4,X'FF'-UIDBSCR  Rebuild the screen
SWAP9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE 3
*
* Clear user screen. Only valid if in Refresh mode and CMS scroll is ON
*
*
CLEARCMD EQU   *                      Clear user screen
         ST    R14,CMDSV14
         SR    RØ,RØ                  No table to search
         GO    CSCSCN                 Scan parameter
         BNZ   CLEA1ØØ                Nothing found, that's good news
         MSG   Ø312,USER              No parameters allowed
         B     CLEA9ØØ
         SPACE
CLEA1ØØ  TM    UIDOPT2,UIDAUTO        User in Refresh mode?
         BO    CLEA2ØØ                Yes, so far so good
         MSG   Ø33Ø,(USER,NOCMD)      No, display error message
         B     CLEA9ØØ
         SPACE
CLEA2ØØ  TM    UIDOPT3,UIDCMS         Is CMS scroll active?
         BO    CLEA3ØØ                Yes, very good indeed
         MSG   Ø331,(USER,NOCMD)      Too bad, tell the user
         B     CLEA9ØØ
         SPACE
CLEA3ØØ  LINK  CLEAR                  Clear user screen
         OI    UIDOPT3,UIDCLEAR       Remember we did it
         OI    UIDOPT4,UIDBSCR        Rebuild user screen
         TM    UIDOPT3,UIDWRAP        Is WRAP switch On?
         BZ    CLEA9ØØ                  No, done
         GO    CSCWRPTP                 Yes, build partial lines
CLEA9ØØ  L     R14,CMDSV14
         BR    R14
         SPACE 3
*
* Create Header override
*
*
CSCUSCRH RELOC                          Change header (external call)
         BAS   R14,ADDHDR               Perform function
         BACK                           Go back to caller
         SPACE
ADDHDR   EQU   *                        Change Header line
         MVC   HDRDATA,HDRINIT          Start with default line
         LA    R1,HDRDATA               Address data
         ST    R1,SCRHDR                Store address for CSCBLD
         MVC   Ø(L'HDRPREF,R1),HDRPREF Move Prefix indicator
         LA    R1,L'HDRPREF(,R1)
         TM    UIDOPT2,UIDDATE          Is DATE to be displayed?
         BZ    ADDH1ØØ
         MVC   Ø(L'HDRDATE,R1),HDRDATE Yes, move it and advance pointer
         LA    R1,L'HDRDATE(,R1)
ADDH1ØØ  TM    UIDOPT2,UIDTIME          Is TIME to be displayed?
         BZ    ADDH2ØØ
         MVC   Ø(L'HDRTIME,R1),HDRTIME
         LA    R1,L'HDRTIME(,R1)
ADDH2ØØ  TM    UIDOPT2,UIDUSER          Is USER to be displayed?
         BZ    ADDH3ØØ
         MVC   Ø(L'HDRUSER,R1),HDRUSER
         LA    R1,L'HDRUSER(,R1)
ADDH3ØØ  TM    UIDOPT2,UIDAUTO          Are we in Refresh mode?
         BZ    ADDH4ØØ                  No, must be Browse
         MVC   Ø(L'HDRCURR,R1),HDRCURR Move Current Header
         LA    R1,L'HDRCURR(,R1)
         B     ADDH5ØØ
         SPACE
ADDH4ØØ  MVC   Ø(L'HDRBRSE,R1),HDRBRSE Move Browse Header
         LA    R1,L'HDRBRSE(,R1)
ADDH5ØØ  TM    UIDOPT2,UIDINC           Selective Include?
         BZ    ADDH6ØØ                  No, try Exclude
         MVC   Ø(L'HDRINCL,R1),HDRINCL Move Include Header
         LA    R1,L'HDRINCL(,R1)
         B     ADDH7ØØ
         SPACE
ADDH6ØØ  TM    UIDOPT2,UIDEXC           Selective Exclude?
         BZ    ADDH8ØØ                  No, try special options
         MVC   Ø(L'HDREXCL,R1),HDREXCL Move Exclude Header
         LA    R1,L'HDREXCL(,R1)
ADDH7ØØ  MVC   Ø(L'UIDSEL,R1),UIDSEL   Move Include/Exclude prefixes
```

```
         LA    R2,L'UIDSEL-1(,R1)      Last possible byte
ADDH71Ø  CLI   Ø(R2),C' '              Back-up blanks
         BNE   ADDH72Ø
         MVI   Ø(R2),C'_'              Replace them with underscores
         BCT   R2,ADDH71Ø
ADDH72Ø  LA    R1,1(,R2)               Adjust pointer
         MVC   Ø(L'HDRIECL,R1),HDRIECL Close Include/Exclude
         LA    R1,L'HDRIECL(,R1)
ADDH8ØØ  TM    UIDOPT3,UIDFLTR         Is Filter active?
         BZ    ADDH81Ø
         MVC   Ø(L'HDRFLTR,R1),HDRFLTR Yes, Move filter indicator
         LA    R1,L'HDRFLTR(,R1)
ADDH81Ø  TM    UIDOPT3,UIDWRAP         Is Wrap active?
         BZ    ADDH82Ø
         MVC   Ø(L'HDRWRAP,R1),HDRWRAP
         LA    R1,L'HDRWRAP(,R1)
ADDH82Ø  TM    UIDOPT3,UIDCMS          Is CMS active?
         BZ    ADDH9ØØ
         MVC   Ø(L'HDRCMS,R1),HDRCMS
ADDH9ØØ  LA    R1,L'HDRDATA            We are done, get Header length
         ST    R1,SCRHDRL              Store it for CSCBLD
         BR    R14
         SPACE 3
*
* Rebuild the user screen
*
*
CSCUSCRB RELOC                         Rebuild (external call)
         BAS   R14,REBUILD             Perform function
         BACK                          Go back to caller
         SPACE
REBUILD  EQU   *                       Rebuild user screen
         ST    R14,REBSV14
         TM    UIDOPT2,UIDAUTO         Is user in Refresh mode
         BZ    REB1ØØ
         GO    CSCUIN                  Yes, build initial screen
         B     REB9ØØ
         SPACE
REB1ØØ   GO    CSCWRPGT                Locate top line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO             Is it TOF?
         BNE   REB2ØØ                  No...
         CLI   CCHPREF,C' '            Is it TOF or just a blank line?
         BE    REB2ØØ                  Just a blank...
         BAS   R14,TOPSCR              Rebuild "TOP" screen
         B     REB9ØØ
         SPACE
REB2ØØ   GO    CSCWRPGB                Locate bottom line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO             Is it EOF line
         BNE   REB3ØØ
         CLI   CCHPREF,C' '              Is it EOF or just a blank line?
         BE    REB3ØØ                    Just a blank...
         BAS   R14,BOTSCR                Rebuild "BOTTOM" screen
         B     REB9ØØ
         SPACE
REB3ØØ   GO    CSCWRPGT                  Locate top line on screen
         SR    RØ,RØ
         C     RØ,CCHRECNO               Is it a normal DF record?
         BNE   REB32Ø                    Yes, adjust screen from top
         CLI   CCHPREF,C' '              Is it EOF or just a blank line?
         BE    REB5ØØ                    Just a blank...
REB32Ø   LINK  SELECT                    Is it expected by the user?
         BZ    REB4ØØ
         GO    CSCRDFNT                  No, get the next record
         BZ    REB4ØØ
         BAS   R14,BOTSCR                Not found, build "BOTTOM" screen
         B     REB9ØØ
         SPACE
REB4ØØ   L     R1,UIDBUFF2               Copy first to after the last
         LINK  ADD
         L     R7,UIDBUFF1               Address first line
         LINK  DELETE                    Delete it
         L     R7,UIDBUFF2               Address last line just added
         BAS   R14,TOPLINE               Shift it to the top
         B     REB9ØØ
         SPACE
REB5ØØ   GO    CSCWRPGB                  Locate bottom line on screen
         LINK  SELECT                    Is it expected by the user?
         BZ    REB6ØØ
         GO    CSCRDFPR                  No, get the previous record
         BZ    REB6ØØ
         BAS   R14,TOPSCR                Not found, rebuild "TOP" screen
         B     REB9ØØ
         SPACE
REB6ØØ   SR    R1,R1                     Add as first line
         LINK  ADD
         L     R7,UIDBUFF2               Address last line
         LINK  DELETE                    Delete it
         L     R7,UIDBUFF1               Address first line just added
         BAS   R14,BOTLINE               Shift it to the bottom
*        B     REB9ØØ
         SPACE
REB9ØØ   L     R14,REBSV14
         BR    R14
         SPACE 3
         DS    ØD
CMDSV14  DS    F                         Save area for input commands
PFKSV14  DS    F                         Save area for PF Key commands
REBSV14  DS    F                         Save area for REBUILD routine
```

```
TOPSV14  DS    F                        Save area for TOPSCR routine
BOTSV14  DS    F                        Save area for BOTSCR routine
NEWBOT   DS    F                        New bottom line when scrolling
NEWTOP   DS    F                        New top line when scrolling
         SPACE
@SCUSASD DC    V(CSCUSASD)              Send user data to remote node
@SCUSADN DC    V(CSCUSADN)              Process disconnect command
         SPACE
         DS    ØD
HDRDATA  DS    CL6Ø                     Field to build new Header
HDRINIT  DC    6ØC'_'                   Default Header
HDRPREF  DC    C'>_'                    Prefix header
HDRDATE  DC    C'__Date___'                Date
HDRTIME  DC    C'__Time___'                Time
HDRUSER  DC    C'__User___'                User
HDRCURR  DC    C'Current___'               Current screen (refresh)
HDRBRSE  DC    C'Browse___'                Browse
HDRINCL  DC    C'Inc('                     Selective Include
HDREXCL  DC    C'Exc('                     Selective Exclude
HDRIECL  DC    C')_'                       Close Selective Inc/Exc
HDRFLTR  DC    C'F_'                       Filter active
HDRWRAP  DC    C'W_'                       Wrap
HDRCMS   DC    C'C_'                       CMS
         SPACE
* Command classes in use
*
* PA/PF Key classes (until someone codes a macro)
*
*    ØØ - X'ØØØØØØ'     *** PA/PF classes should match the          ***
*    Ø1 - X'8ØØØØØ'     *** equivalent command class                ***
*    Ø2 - X'4ØØØØØ'     *** ie PFØ7 and BWD are both class Ø3       ***
*    Ø3 - X'2ØØØØØ'     ***                                         ***
*
*    Ø1 - General commands
*    Ø2 - Commands associated with refresh mode
*    Ø3 - Commands associated with browse  mode
*    Ø4 - Commands associated with Data File search
*         Special commands
*    Ø5 -   Release
*    Ø6 -   OP
*    Ø7 -   COnnect / DIsconnect
*
PFTABLE  DS    ØD
         DC    AL1(ENTER),X'ØØØØØØ',A(REFRESH)   Refresh the screen
*
         DC    AL1(PF4),X'2ØØØØØ',A(TOP)         PFØ1-PF12
         DC    AL1(PF5),X'2ØØØØØ',A(BOTTOM)
         DC    AL1(PF7),X'2ØØØØØ',A(BWD)
         DC    AL1(PF8),X'2ØØØØØ',A(FWD)
         DC    AL1(PF9),X'4ØØØØØ',A(CURRENT)
         DC    AL1(PF1Ø),X'8ØØØØØ',A(SHIFT)
*
         DC    AL1(PF16),X'2ØØØØØ',A(TOP)        PF13-PF24
         DC    AL1(PF17),X'2ØØØØØ',A(BOTTOM)
         DC    AL1(PF19),X'2ØØØØØ',A(BWD)
         DC    AL1(PF2Ø),X'2ØØØØØ',A(BWD)
         DC    AL1(PF21),X'4ØØØØØ',A(CURRENT)
         DC    AL1(PF22),X'8ØØØØØ',A(SHIFT)
         DC    X'FFFFFFFFFFFFFFFF'
         SPACE
LOCATE   CMMD  (E,Ø4,Ø1,'/          ',CSCULC)    Default Locate command
MATCH    CMMD  (E,Ø4,Ø1,'\          ',CSCULCMT)  Default Match  command
         SPACE
SWPTABLE CMMD  (B,ØØ,Ø1,'DATE       ',X'Ø1ØØ'+UIDDATE),    UIDOPT2     *
               (B,ØØ,Ø1,'TIME       ',X'Ø1ØØ'+UIDTIME),                *
               (B,ØØ,Ø1,'USER       ',X'Ø1ØØ'+UIDUSER),                *
               (B,Ø3,Ø1,'FILTER     ',X'Ø2ØØ'+UIDFLTR),    UIDOPT3     *
               (B,ØØ,Ø1,'WRAP       ',X'Ø2ØØ'+UIDWRAP),                *
               (B,Ø2,Ø1,'CMS        ',X'Ø2ØØ'+UIDCMS)
         SPACE
SHFTABLE CMMD  (B,ØØ,Ø1,'LEFT       ',SHFLEFT),  Shift options         *
               (B,ØØ,Ø1,'RIGHT      ',SHFRIGHT)
SHFLEFT  EQU   X'8Ø'                                Left
SHFRIGHT EQU   X'4Ø'                                Right
         SPACE
USCTABLE CMMD  (I,Ø3,Ø1,'BACKWARD   ',BWDCMD),   User Commands Table   *
               (I,Ø3,Ø3,'BWD        ',BWDCMD),                         *
               (I,Ø3,Ø3,'BOTTOM     ',BOTCMD),                         *
               (I,Ø2,Ø1,'CLEAR      ',CLEARCMD),                       *
               (E,Ø7,Ø2,'CONNECT    ',CSCUSACN),                       *
               (I,Ø2,Ø3,'CURRENT    ',CURCMD),                         *
               (I,Ø7,Ø2,'DISCONNECT ',DISCONN),                        *
               (E,Ø3,Ø1,'DOWN       ',CSCUSBDN),                       *
               (I,ØØ,Ø3,'END        ',END),                            *
               (E,ØØ,Ø1,'EXCLUDE    ',CSCUEX),                         *
               (I,Ø3,Ø1,'FORWARD    ',FWDCMD),                         *
               (I,Ø3,Ø3,'FWD        ',FWDCMD),                         *
               (E,Ø4,Ø1,'GO         ',CSCULCGO),                       *
               (E,ØØ,Ø1,'INCLUDE    ',CSCUEXIN),                       *
               (I,Ø1,Ø2,'LEFT       ',SHIFTLE),                        *
               (E,Ø4,Ø1,'LOCATE     ',CSCULC),                         *
               (E,Ø4,Ø5,'DOWNLOCATE ',CSCULCDL),                       *
               (E,Ø4,Ø2,'DLOCATE    ',CSCULCDL),                       *
               (E,Ø4,Ø1,'MATCH      ',CSCULCMT),                       *
               (E,Ø4,Ø5,'DOWNMATCH  ',CSCULCDM),                       *
               (E,Ø4,Ø2,'DMATCH     ',CSCULCDM),                       *
               (E,Ø3,Ø1,'NEXT       ',CSCUSBDN),                       *
               (E,Ø6,Ø2,'OP         ',CSCUOP),                         *
               (E,Ø3,Ø1,'PRINT      ',CSCUPR),                         *
               (E,Ø5,Ø1,'RELEASE    ',CSCURL),                         *
               (I,Ø1,Ø2,'RIGHT      ',SHIFTRI),                    *
               (E,Ø1,Ø2,'SET        ',CSCUST),                     *
               (I,Ø1,Ø2,'SHIFT      ',SHIFTCMD),                   *
               (I,Ø1,Ø1,'SWAP       ',SWAP),                       *
               (I,Ø1,Ø1,'SWITCH     ',SWAP),                       *
               (I,Ø3,Ø1,'TOP        ',TOPCMD),                     *
               (E,Ø3,Ø1,'UP         ',CSCUSBUP),                   *
               (E,Ø3,Ø1,'WRITE      ',CSCUPRWR)
         SPACE
         CSCDATA
         CSCDS (UID,CCH,PFX,CMD)
         SPACE
         DMSDSBLK
         REGEQU
         END
```

CSCUIN ASSEMBLE

```
         TITLE 'CSCUIN - CSC Process User INIT command'
CSCUIN   START X'Ø1842Ø'
         PRINT NOGEN
         CSCHDR                        Process INIT command
*
* Process <CSC>INI User command (INIT)
*
*
         USING UIDSECT,R8              UID (user) Block
         USING CCHSECT,R7              CCH (cache) Block
         USING CQYSECT,R6              CQY Console Query Block
         TM    UIDOPT2,UIDINIT         Already initialized?
         BO    UIN2ØØ                  Yes, rebuild screen
         OI    UIDOPT2,UIDINIT         No, remember we did it next time
         LA    R6,CSCBUFF+L'COMMCMD    Address Console Query Block
         LH    R1,CQYDQRRW             Number of screen lines
         LA    RØ,5                    Lines for Header and Trailer
         SR    R1,RØ                   Number of detail lines
         STC   R1,UIDSCRL              Save number of detail lines
         TM    CQYDQRFL,CQYDQREC+CQYDQREH Check Extended Data Stream
         BNO   UIN1ØØ                  Not supported
         OI    UIDOPT2,UIDEDS          Colours and EH supported
         DROP  R6
         SPACE
UIN1ØØ   OI    UIDOPT2,UIDAUTO+UIDTIME Default options
         OI    UIDOPT4,UIDBTTL         Set option to refresh Title
UIN2ØØ   L     RØ,UIDCLASS             Load user classes
         SLL   RØ,1                    Make bit 1 (class 2) the first
         LTR   RØ,RØ                   Does user have class 2?
         BM    SCREEN                  Yes, good enough
         NI    UIDOPT2,X'FF'-UIDAUTO   Reset refresh option
         OI    UIDOPT4,UIDBHDR          Rebuild Header line
SCREEN   SR    RØ,RØ                    Clear User buffer
         ST    RØ,UIDBUFF1
         ST    RØ,UIDBUFF2
         L     R1,UIDBUFF               Rebuild User Free List
         ST    R1,UIDFREE1              Save address of first entry
         LA    R2,UIDBUFSZ              Load buffer size in double words
         SRL   R2,5                     Number of entries (256 = 32 dw)
SCR1ØØ   LR    R7,R1                    Address entry
         XC    CCHSECT(CCHSIZEB),CCHSECT Clear all entry
         LA    R1,CCHSIZEB(,R1)         Next entry
         ST    R1,CCHFWD                Store forward pointer
         ST    RØ,CCHBWD                Store backward pointer
         LR    RØ,R7                    Save address of current entry
         BCT   R2,SCR1ØØ                Do all entries
         SR    RØ,RØ
         ST    RØ,CCHFWD                Zero fwd pointer of last entry
         ST    R7,UIDFREE2              Save address of last Free entry
         SPACE
         SR    R6,R6                    Clear R6 for next IC
         IC    R6,UIDSCRL               Load number of screen lines
         L     R7,CACHEPTR              Address current record
         B     SCR21Ø                   Enter the loop, use back door
         SPACE
SCR2ØØ   L     R7,CCHBWD                Move last records to user buffer
         C     R7,CACHEPTR              End of cache
         BE    SCR3ØØ                   Yes, try read from disk
SCR21Ø   OI    UIDOPT3,UIDNODSP         Reject NoDisplay messages
         LINK  SELECT                   Expected by the user?
         BNZ   SCR2ØØ                   No, ignore it
         SR    R1,R1                    Add as first record
         LINK  ADD
         BCT   R6,SCR2ØØ
         B     SCR6ØØ
         SPACE
SCR3ØØ   L     R7,CCHFWD                Address first (oldest) cache rec
SCR31Ø   GO    CSCRDFDP                 Read previous record from disk
         BNZ   SCR4ØØ                   Not found, clear user buffer
         OI    UIDOPT3,UIDNODSP         Reject NoDisplay messages
         LINK  SELECT                   Expected by the user?
         BNZ   SCR31Ø                   No, ignore it
         SR    R1,R1                    Add as first record
         LINK  ADD
         BCT   R6,SCR31Ø
         B     SCR6ØØ
         SPACE
SCR4ØØ   L     R7,UIDFREE1              Address first user free record
         MVC   CCHUSER,BLANKS
         MVC   CCHDATA(L'TOF),TOF       Build Top-Of-File message
         LA    RØ,L'TOF
         STC   RØ,CCHRLEN
         LINK  PREFIX                  Get prefix and attributes
         SR    R1,R1                   Add as first record
         OI    UIDOPT1,UIDFFREE        Record is from Free list
         LINK  ADD
         BCTR  R6,Ø                    Count this line
         LTR   R6,R6
         BZ    SCR6ØØ
SCR5ØØ   L     R7,UIDFREE1             Get another Free record
*        LA    RØ,1                    Build length field
*        STC   RØ,CCHRLEN
         SR    R1,R1
         OI    UIDOPT1,UIDFFREE        Record is from Free list
         LINK  ADD
         BCT   R6,SCR5ØØ
SCR6ØØ   IC    R6,UIDSCRL              Load number of screen lines
         L     R5,UIDBUFF1             First user record
         L     R4,HLDPTR               Address messages on Hold
SCR7ØØ   LTR   R7,R4
         BZ    SCR8ØØ                  Nothing left
         L     R4,CCHFWD               Next message on Hold
         CLC   CCHDATE,CCHDATE-CCHSECT(R5) Compare date and time
         BH    SCR8ØØ                  Message already on screen
         BL    SCR79Ø
         CLC   CCHTIME,CCHTIME-CCHSECT(R5)
         BH    SCR8ØØ
         BL    SCR79Ø
         CLC   CCHRECNO,CCHRECNO-CCHSECT(R5) Check also record number
         BE    SCR8ØØ                  It should be the one
         LR    R3,R5                   Scan user screen
SCR71Ø   L     R3,CCHFWD-CCHSECT(,R3)
         LTR   R3,R3                   Last buffer record?
         BZ    SCR79Ø                  Yes, message is not on screen
         CLC   CCHRECNO,CCHRECNO-CCHSECT(R3)
         BNE   SCR71Ø                  Not this one, try next
         CLC   CCHTIME,CCHTIME-CCHSECT(R3)
         BNE   SCR79Ø
         CLC   CCHDATE,CCHDATE-CCHSECT(R3)
         BE    SCR8ØØ                  It is there, no need to add it
SCR79Ø   LINK  SELECT                  Expected by the user?
         BNZ   SCR7ØØ                  No, next please...
         LR    R1,R5                   Address line to replace
         LINK  ADD                     Add new line after this one
         LR    R7,R5                   Address line to replace (delete)
         L     R5,CCHFWD-CCHSECT(,R5)  Next line, the one added
         L     R5,CCHFWD-CCHSECT(,R5)  Next line, possible replace
         LINK  DELETE                  Delete line
         BCT   R6,SCR7ØØ               Check all messages on Hold
SCR8ØØ   TM    UIDOPT3,UIDCMS          CMS scrolling?
         BZ    SCR9ØØ                  No, all done
         TM    UIDOPT3,UIDCLEAR         Was screen cleared?
         BO    SCR860                   Yes, so do it again
         L     R7,UIDBUFF1              Address first screen line
SCR810   L     R4,CCHFWD                Save address of next line
         L     R1,CCHRECNO              Is it a blank line?
         LTR   R1,R1
         BNZ   SCR820                   No, almost done
         CLI   CCHUSER,X'00'            Blank or just TOF
         BNE   SCR900
         LINK  DELETE                   Real a blank line, delete it
         LINK  ADDBLKB                  Add a new blank at the bottom
         LR    R7,R4                    Remove all top blank lines
         B     SCR810
         SPACE
SCR820   L     R0,UIDCMSTP              Last CMS top line
         L     R1,UIDBUFF1              First screen line
SCR830   LTR   R7,R1
         BZ    SCR900                   End of buffer, record not found
         L     R1,CCHFWD                Address next line
         C     R0,CCHRECNO              Check record number
         BNE   SCR830                   Not this one, keep trying
         L     R4,CCHBWD                Found it, delete lines above
SCR840   LTR   R7,R4                    All lines checked?
         BZ    SCR900                   Yes, now we are done
         L     R4,CCHBWD                Address previous line
         TM    CCHOPTS,CCHHOLD          Message on hold?
         BO    SCR840                   Yes, don't touch it
         LINK  DELETE                   Delete line
         LINK  ADDBLKB                  Add blank at the bottom
         B     SCR840
         SPACE
SCR860   LINK  CLEAR                    Clear screen again
*        B     SCR900
         SPACE
SCR900   TM    UIDOPT3,UIDWRAP          Is Message WRAP active?
         BZ    SCR990
         GO    CSCWRP                   Yes, build partial lines
SCR990   OI    UIDOPT4,UIDBSCR          Option to rebuild user screen
         BACK
         SPACE 3
         CSCDATA
         CSCDS (UID,CCH)
         CQYSECT
         REGEQU
         END
```

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

Fernando Duarte
Analyst (Canada)

VM:Secure enhancement rules

Object Rules are special macros that enhance VM:Secure rules to allow additional resource access control. Building on the VM:Secure rules logic, Object Rules can be added to secure external resources such as MVS datasets accessed from VM, or any resource used or owned by a VM product. Object Rules allow products that have user-written or RACF-like security exits to use VM:Secure for access control.

BACKGROUND

Our VM security was under audit review and it was determined that there was a severe problem controlling access to MVS data from VM using CMS FILEDEF and OS simulation (we share all data between our VM and MVS mainframe systems). There was also a concern about the number of products we have on VM that have security exits with in-house written code. All take a considerable amount of administration because most use hard-coded security tables.

For our security, RACF is used on MVS and VM:Secure with rules is used on VM. VM/RACF was reviewed as a possible solution to the audit issues, but it was determined that VM/RACF would not solve the problems because of certain limitations in the product. Furthermore, because we were heavily invested in VM:Secure, it would have been a tremendous effort to convert users, applications, and administration to use VM/RACF – the disadvantages outweighed the benefits. Inquiries into other security products proved to be fruitless because nothing on the market seemed to provide the services we needed.

Discussions with the VM:Secure technical staff determined that rules *may* be enhanced to handle external product security, but there was no immediate direction indicated. Reviewing the Object Rules concept with them indicated that the idea was quite feasible.

We ended up with two choices – either eliminate all external MVS data access from VM and continue administration of security tables for the various VM products, or build our own solution. We chose to use VM:Secure and its robust macro primitives to build a solution.

OBJECT RULES CONCEPT

Object Rules are a set of VM:Secure macros that have functions such as adding, updating, deleting, and querying. The macros can be used on the primary VM:Secure server, or placed on a stand-alone VM:Secure server.

An ‘object’ is considered to be any resource that needs to be secured. Each object can have parameters (or tokens) associated with it to further define the resource. These Object Definitions are built and loaded into the VM:Secure server. Then, using ACCEPT and REJECT rule structures, these objects or external resources can be secured for any user-id.

For example, an external product needs to check for READ or UPDATE authority on accessing reports. The report names can be up to 12 characters long.

The Object Name could be called REPORT. Object REPORT would have two parameters/tokens associated with it – the report name and READ or UPDATE. A user’s Object Rule file could look something like the following:

```
ACCEPT REPORT STATUS* READ
ACCEPT REPORT TEMPLATE UPDATE
REJECT REPORT SALARY* *
```

The Object Rules would allow READ access for this user to any report name that began with STATUS and UPDATE access to any report called TEMPLATE. It would reject ALL access (the ‘*’ as the second parameter denotes this) to any report name beginning with SALARY.

Note that Object Rules are intended for securing resources ‘owned’ by the system or a particular product, not the user. Unlike VM:Secure rules, users should not have authority to edit their own Object Rules. A security administrator should be assigned the task of maintaining the Object Rules for users.

The validation process begins with a global default access setting of ACCEPT or REJECT. This access is used for objects that have no rules associated with them in the User or SYSTEM file. The process will then check for a USERID OBJECTS file. If that file is found, it will look for the object and its tokens and, if an entry is found, it will use the ACCEPT or REJECT as the result. If not found, the process looks for a SYSTEM OBJECTS file. If that file is found, it will look for the object and its parameters and, if an entry is found, it will use the ACCEPT or REJECT as the final access to that object.

Object Rules are checked from the top downwards. The first occurrence that matches a request is used for the access (ACCEPT or REJECT). A simplistic flow of control is as follows:

- A request to validate access for an object arrives:
  - The Object Rules environment is verified.
  - The global default access is obtained.
  - The object is validated.
  - The object's default access is obtained.
  - If found, it replaces the global default access.
- Look for a USERID OBJECTS file in storage:
  - If found, search for an Object Rule that matches the request.
  - If found, return the result.
- Look for a SYSTEM OBJECTS file in storage:
  - If found, search for an Object Rule that matches the request.
  - If found, return the result.
- If there are no matches, then use the default access and return the result.

The returned result can be either ACCEPT (zero Return Code and no message) or REJECT (298 Return Code with an error message).

OBJECT RULE SEARCH LOGIC

In general, the searching for a particular Object Rule matches what VM:Secure rules does. It also handles RACF-like rules if the object parameter is built like a RACF resource (xxx.yyy.zzz) with ‘dots’ separating words. This feature allows the best of both worlds – setting up both VM:Secure rule and RACF rule structures in one package.

Object Rules’ parameters can use pattern match (%) and wildcard (*) characters. For example:

```
ACCEPT REPORT SALARY%%98 READ
```

This would allow the user to READ any report that started with SALARY, had any two characters following it, and then 98 – such as SALARYJV98.

The pattern and characters can be mixed:

```
REJECT MVSDATA PAYROLL.PROD%%%.DATA*
```

This would reject access to an object called MVSDATA that has a parameter looking like a RACF resource (or MVS dataset name) if the name begins with PAYROLL.PROD then has any four characters, then .DATA, and ends with any character(s).

All Object Rule search logic is as follows (for User and SYSTEM object rules). For each case, searching begins at the top and searches down the list of Object Rules found (ie the order you put the Object Rules in really matters!).

Specific matches

If an Object Rule exactly matches the request, that rule is used. If a specific SYSTEM Object Rule is found and a user has an Object Rule for that same object, the SYSTEM rule takes precedence.

For example, with:

```
SYSTEM Rule:  REJECT REPORT SARARYVP READ
User Rule:    ACCEPT REPORT SARARY* READ
Attempted:    VMSECURE OBJCHK SALARYVP READ
```

In this case, the user attempting to access Object REPORT to look at (READ) report name SALARYVP would not get access. The specific SYSTEM entry overrides the (generic) user entry. The user would need the entry ACCEPT REPORT SALARYVP READ added, to be allowed to use that report.

With a second example:

```
SYSTEM Rule:   REJECT REPORT SARARYVP *
User Rule:     ACCEPT REPORT SARARY* READ
Attempted:     VMSECURE OBJCHK SALARYVP READ
```

the user, attempting to access object REPORT to look at (READ) report name SALARYVP, would get access. The SYSTEM entry is technically a (generic) entry since the second parameter is an '*'. The user entry allows READ access to any report beginning with SALARY and ending in anything.

Pattern matches

If an Object Rule matches with one or more pattern-match characters (%), that rule is used.

Wild card matches

If an Object Rule is found that matches the request (with or without pattern characters), that rule is used.

For additional logic, review the code.

TECHNICAL INFORMATION

VM:Secure is used as the base for these macros because it has an excellent multitasking nucleus. The multitasking is needed to provide adequate and fair response times to security queries.

The VM:Secure macros are written in REXX and use CMS Pipelines for the majority of the data manipulation. New VM:Secure messages are required for error and informational messages.

Because CMS Pipelines are heavily used, CMS Version 9 (VM/ESA) or later will be required to use these macros.

SEPARATE SERVER

If you plan to install Object Rules on a separate VM:Secure server, Release 1.1C or later is recommended to be used. There is a VMXSYS parameter called DEBUG NODIAG3C which can be used so that the

VM:Secure server does not attempt to update the on-line system directory. Because the the main function of VM:Secure is to maintain the on-line directory, you will want to disable that function on the separate server.

Although it is perfectly acceptable to run Object Rules on the primary VM:Secure server, it is recommended that you use a second VM:Secure server. This is to ensure that the primary VM directory security and rules are not impacted and it also allows Object Rules to get better performance.

Using a separate server also gives you the ability to have one server per application. This allows you to keep security impact to a minimum if there is a problem with one of the servers.

STORAGE USE

The Object Definitions, user objects, and some global settings are loaded into the server's storage. Defining more Object Rules will require more virtual storage of the server. It is recommended that you run the server in XA/ESA mode and define storage over 16MB.

Local benchmarks showed that less than 5MB was required to load over 90,000 Object Rules for around 1,000 user-ids. Your storage requirements will vary depending on your usage.

The macros validate all data being loaded and format it so that it can be loaded above or below the 16MB line if in XA/ESA mode. The process utilizes CMS EXECLOAD to load the data. Since EXECLOAD will load any file into storage, it was the easiest way to keep the data in storage for quicker access.

One aspect of EXECLOAD is that, if it detects the file being loaded is a REXX EXEC, it will attempt to load it above the 16MB line (if available). The loading processes build the data files beginning with '/* */' so that they appear as REXX files even though they are, in most cases, just plain data.

CMS Pipelines is used to read data loaded by EXECLOAD. The stage command '<' will check for EXECLOADed files if no filemode is specified on that stage (see the CMS Pipelines Reference for more information). This allows multiple processes to share in-storage data and get superb I/O response times.

OBJECT DEFINITION FILES

The Object Definition File (OBJDEF) contains the information that defines the attributes (default access, tokens, etc) of an object you want to protect. You need one OBJDEF file for each object you want to define.

The filename of the OBJDEF file needs to be the name of the object – the filetype of the file has to be OBJDEF. The OBJDEF files need to reside on the Object Rules Disk you defined for them to be effective at initialization.

Each OBJDEF file should contain a Default_Action, Tokens, and at least one Token.x/Default.x pair of entries. It can include as many comments (line beginning with a '*') or blank lines as you like. A simple example follows:

```
|...+....1....+....2....+....3....+.
* * * Top of File * * *
*  Object definition of REPORT

Default_Action    ACCEPT

Tokens            2

Token.1           (1,12)
Default.1

Token.2           READ|UPDATE
Default.2         READ
```

The following sections cover each of the entries in detail.

Default_Action

'Default_Action' defines the default access granted if no rule is found for that object when access is requested. The only values that can be assigned are ACCEPT or REJECT.

Tokens

‘Tokens’ defines the number of tokens that the object has and must be included before any Token.n or Default.n entries. In the example above, the REPORT object has two tokens to define the object (’Tokens 2’). For each token defined, there should be a corresponding Token.n entry. For example, with ‘Tokens 2’, there should be ‘Token.1’ and ‘Token.2’ entries.

Token.n

The ‘Token.n’ entry defines that token for the object. It can be one of two formats as seen above.

The first format (word1|word2...) allows you to define specific words to that parameter. The list of words defines the only values that the token can be when defined in the SYSTEM or a User OBJECTS file, or when an OBJCHK or OBJFOR command is issued. For example:

```
    Token.2  READ|UPDATE
```

allows token 2 to be the word READ or UPDATE. Any other value is invalid on an access request or Object Rule.

The second format defines the minimum (‘n’) and maximum (‘nn’) number of characters allowed in the token. This allows you to have any string allowed for an object’s token.

‘n’ can be from one to 70. ‘nn’ can be from one to 70 and must be greater than or equal to ‘n’.

For example:

```
    Token.1  (1,12)
```

allows the token to have at least one character and no more than 12 characters. If the token is less than one or greater than 12 characters, it will be considered an error.

Default.n

‘Default.n’ defines the default value for the Token.n (where ‘n’ is the same for each) if that token is not supplied on a Object Rule in the

SYSTEM or User OBJECTS file only. The Default.n entry must follow any Token.n entry; Default.n entries are optional.

Note: in general, only the last Default.n can have a value assigned to it since intermediate tokens cannot be left out (ie you cannot leave out token 2 of a four- token Object Rule – the code would count the tokens and assume that token 4 was missing). So if you define a Default.n for any token except the last one, it is basically ignored.

The value can either be a string or null (nothing entered after Default.n). If the Token.n entry used the 'word1|word2...' format, then the Default.n entry must be one of the words listed.

If the Token.n entry used the (n,nn) format, it can be any string with at least 'n' characters and no more than 'nn' characters and cannot include any '*' or '%' characters.

USER AND SYSTEM OBJECT RULES FILE

The User and SYSTEM Object Rules File (OBJECTS) contains the information that defines the Object Rules to govern the whole system and individual users.

The filename of the OBJECTS file will be either SYSTEM or a user-id – the filetype will be OBJECTS. The OBJECTS files need to reside on the Object Rules Disk you defined for them to be effective at initialization.

Generally, the file includes any ACCEPT or REJECT Object Rules and any number of comments (lines beginning with '*') or blank lines.

The format is:

```
    ACCEPT|REJECT objname token1 token2...
```

For example:

```
      |...+....1....+....2....+....3....+..
00000 * * * Top of File * * *
00001 * REPORT RULES
00002 ACCEPT REPORT STERLING READ
00003 ACCEPT REPORT STERL19* UPDATE
00004 ACCEPT REPORT CONF####CE UPDATE
00005 REJECT REPORT CA *
```

Where ‘REPORT’ is the object name and is followed by its token values for that rule (in the previous OBJDEF example, it has two tokens).

Note that for general token strings (defined with the (n,nn) format), wildcard and pattern-matching characters are valid.

INSTALLATION

Object Rules are relatively simple to install. It is completely up to you how you want to manage the source files and executables. This section is intended to give you some suggested methods for installing the Object Rules code.

The installation has two basic options – installing on your primary VM:Secure server or installing on a separate server. Although the separate server installation path is longer, it is far better than using the same server used for all other VM security.

INSTALLATION OPTION 1

The following steps outline the installation on a separate VM:Secure server (user-id). This is the recommended method for using Object Rules:

1 Create a user-id to be the server. In the following examples, it is assumed that a user-id of VMSECUR2 has been created to hold the ObjectRules code. The ID should have the same directory statements as stated in the *Installation Guide for VM:Secure*, with the following exceptions:

   – The directory disk (1B0), back-up disk (1B1), and hold disk (1B2) can all be one cylinder each. The 1B0 disk is the only one that will actually have data on it. Optionally, you can create TDISKs for the 1B1 and 1B2 in the PROFILE EXEC. Since they are not used for anything, a temporary disk is fine.

   – Add a 1A0 disk as two cylinders. This will be CPFMTXAd as a dummy directory area for the server to write to.

   – Create a disk to hold the Object Rules files.

Use any address you like, but a 1B4 disk address is fine. This matches what VM:Secure uses for its RULE disk. The OBJDEF, User, and SYSTEM files will be placed on this disk.

2 CPFMTXA the 1A0 disk. Allocate from cylinder 1 to the end as DRCT, and label it NODIR.

3 Load the Object code to the 191 disk. This includes the macros, new messages, etc.

4 Copy the OBJECT TEMPLATE to the 1B4 (Object Rule) disk.

5 Update the AUTHORIZ CONFIG file to include any user-ids you want to authorize to access the Object code. The user-id of the SERVER must be granted authority for all OBJ commands!

6 Update the VMSECURE MANAGERS file to include all user-ids you want to authorize to access VMSECURE CONFIG commands

7 Create a dummy entry on the 1B0 disk for the server-id and any other user-ids you want to be able to do VMSECURE CONFIG commands, like the following:

```
USER userid NOPASS
*1 - ØØØØØØ NOT_A_USER
ACCOUNT accnt-num dist-code
CONSOLE ØØ9 327Ø Ø
SPOOL ØØC 254Ø READER A
SPOOL ØØD 254Ø PUNCH A
SPOOL ØØE 14Ø3
```

8 Create any OBJDEF and SYSTEM or User OBJECT files on the 1B4 disk (or the disk you chose to hold the Object Rules).

9 Update the VMRMAINT CONFIG on VMRMAINT's 192. Duplicate the VM:SECURE entry to look like:

```
    PRODUCT  VM:SECURE  userid  x.x
```

Where 'userid' is the user-id of the server and 'x.x' is the release.

10 Update the VMISTART COMMANDS file on VMRMAINT's 192 disk to include:

```
COMMAND VM:SECURE userid VMXSYS DEBUG NODIAG3C
```

Where ‘userid’ is the user-id of the server.

11 Create a duplicate of VMSECURE MDISKS on VMRMAINT 192’s disk and name it userid MDISKS (where ‘userid’ is the user-id of the server). Change the OwnerId fields to match where the disks are located.

12 Start the server with a SOURCE start to initialize the 1A0 directory. The Object Rules should initialize after VM:Secure is up.

INSTALLATION OPTION 2

The following steps outline the installation on the same VM:Secure server (user-id) that you have running to protect VM log-ons, links, etc.

1 Create a disk to hold the Object Rules files. Use any address you like; however, using an existing VM:Secure disk is not recommended.

2 Load the Object code to the 191 disk. This includes the macros, new messages, and VMRMAINT files – back up the existing 191 files first!

Do not replace the following files if they exist on the 191 disk:

– AUTHORIZ CONFIG

– DASD CONFIG

– SECURITY CONFIG

– VMSECURE MANAGERS

– SYSTEM VMSECURE.

3 Copy the OBJECT TEMPLATE to the 1B4 (Object Rule) disk.

4 Update the AUTHORIZ CONFIG file to include all user-ids you want to authorize to access the Object code. The user-id of the SERVER must be granted authority for all OBJ commands! Use

the AUTHORIZ CONFIG file with this package as a template.

5 Create any OBJDEF and SYSTEM or User OBJECT files on the 1B4 disk (or the disk you decided to hold the Object Rules).

6 Add an UPDATE to the SYSTEM VMSECURE macro to start Object Rules (see the SYSTEM VMSECURE published below as a template).You will need to add:

```
TEST EXEC OBJSTART cuu mode default
```

where:

– ‘cuu’ is the address of the Object disk.

– ‘mode’ is any open mode the disk can be accessed at.

– ‘default’ is ACCEPT or REJECT.

This line of code should be placed before the ‘call housekeeping’ line.

7 Start up the server as you normally would. The Object Rules should initialize after VM:Secure is up.

OBJADD COMMAND

The OBJADD command allows an authorized user to use an Object Rules file on their disk to add or replace an existing Object Rules file for a user. Only valid Object Rules or comments can be included in the file being used. The format is:

```
OBJADD  userid  [  filetype  [  filemode  [  (options
```

where:

- ‘userid’ specifies the user whose Object Rules file is to be added or replaced. If the user Object Rules file already exists, specify the REPLACE option.

- ‘filetype’ specifies the filetype of the Object Rules file to be used. This file must exist on one of the issuing user’s accessed mini-disks. This parameter is optional. If not specified, the default filetype is OBJECTS.

- ‘filemode’ specifies the filemode of the Object Rules file to be used. The filemode must be one of the issuing user’s accessed mini-disks. This parameter is optional and, if not specified, the default filemode is A.
- The option ‘REPLACE’ replaces an already existing Object Rules file for the user-id.

For example, to add the user-id FRANK’s Object Rules file, called FRANK OBJFILE on your A disk, enter:

```
vmsecure objadd frank objfile a
```

To replace user-id FRANK’s Object Rules file using the file called FRANK REPLACE on your A disk, enter:

```
vmsecure objadd frank replace a (replace
```

EXAMPLES OF RESPONSES

Examples of responses follow:

- When VM:Secure successfully adds or replaces the Object Rules file, it will display the following message:

```
VMXSYS8ØØ2I The User Objects have been loaded for user-id
```

- If VM:Secure finds an Object Rules file for the user and the REPLACE option was not used, then the following error is displayed:

```
VMXSYS8Ø21E Object Rules file already exists for user-id
```

- If VM:Secure cannot find the file you specified, the following error message will display:

```
VMXSYSØØ21E File 'userid filetype filemode' not found.
```

- Other situations may occur during OBJADD that may produce other messages.

Return codes and error messages

Return codes and error messages (immediate termination) for OBJADD are shown in Figure 1.

Return code	Message number	Text
2	ØØ38E	Missing parameter
4	ØØ39E	Invalid parameter 'parm'
1Ø	8Ø21E	Objects file already exists for userid
12	Ø265E	Not authorized for: command parms
14	Ø364E	File 'fn ft fm' is being updated
28	ØØ21E	File 'fn ft fm' not found
3Ø	Ø621E	Unexpected return code rc from macro
299	7ØØØE	The OBJECT RULES are not active

Figure 1: OBJADD return codes (immediate termination)

THE OBJCHK COMMAND

The OBJCHK command allows users or applications to query Object Rules authorization.

The format is:

```
OBJCHK  objectname  [token-1]  ... [token-n] [(options]
```

Where:

- ‘objectname’ is the name of the object or resource being checked. The objectname should have been previously defined in an OBJDEF file and loaded for it to be valid.
- ‘token-1...token-n’ is the list of tokens associated with the object. The number of tokens specified must match the defined number in the Object Definition File (OBJDEF).
- The option ‘Quiet’ suppresses the REJECT message issued with return code 298.

Use of OBJCHK

The primary use of the OBJCHK command is from applications that have Object Definitions set up. The applications can use a compiled Module, REXX EXEC, or an EXEC loaded into a segment, in order to ‘hide’ the OBJCHK command. OBJCHK can be used in security exits provided by other VM products to simulate the required security authorization checking.

For the following examples, assume an object has been defined called REPORTS. It has two tokens. One is the report name with a minimum of four and a maximum of 12 characters. The second is either READ or UPDATE. Also assume the server machine name is VMSECURE.

To determine whether the user (on which the OBJCHK command is being issued) has UPDATE authority for a report named MGRSALARY, enter:

```
vmsecure objchk reports mgrsalary update
```

To determine whether the user has READ authority for a report named STATUS, enter:

```
vmsecure objchk reports status read
```

Return codes

The OBJCHK command is primarily a yes or no check for access to a resource. A zero return code denotes access is granted; any non-zero return code denies access (see Figure 2). OBJCHK error messages are shown in Figure 3.

```
Return     Text
code

Ø          An ACCEPT Object Rule was found; access is allowed.
298A       REJECT Object Rule was found; access is denied
               to that resource.
non-zero   A processing error occurred. Access should be denied
               until the error is corrected.
```

Figure 2: OBJCHK return codes

OBJDEL COMMAND

The OBJDEL command allows an authorized user to delete an Object Rules file from the VM:Secure server.

This command should be used with extreme caution! Deleting an Object Rules file for a user removes all Object Rules for that user. If

Return code	Message number	Text
2	8201E	Tokens invalid or missing for object 'objectname'.
4	8206E	Token count does not match for object objectname. Tokens allowed is count
6	8006E	Object name not specified.
28	8200E	Object objectname does not exit.
298	9001E	Access rejected for: 'objectname objectparms...'
299	7000E	The OBJECT RULES are not active.
300	8202E	Severe error rc reading 'file' from storage.

Figure 3: OBJCHK return Codes (Immediate Termination)

you remove the SYSTEM Object Rules file, all default Object Rules are deleted. It has the format:

```
OBJDEL   userid  [  (options
```

Where:

- ‘userid’ specifies the user whose Object Rules file you want to delete.
- The option ‘noprompt’ specifies that you do not want to be prompted to verify the deletion. Be very careful if using this option.

For example, to delete the user-id FRANK’s Object Rules file with no verification prompt, enter:

```
vmsecure objdel frank (noprompt
```

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

James S Vincent
Software Specialist
Nationwide Insurance (USA)

VM news

IBM has announced Version 1.2 of its VisualAge Generator Server for VM. Version 1.2 provides runtime libraries for VisualAge Generator programs (developed with VisualAge Generator Version 3.1 or earlier). The libraries provide support for communication, text user interface applications, and common subroutines that can be shared by all VisualAge Generator programs.

For further information contact your local IBM representative.

* * *

Serena Software has announced Release 8.2.2 of its Comparex software, available for VM, OS/390, MVS, and VSE. This includes a new Euroexit option for conversions from the euro to the local currency unit, or from the local currency unit to the euro.

Comparex performs single-step comparisons of the contents of any two libraries, directories, files, or databases to detect differences between files of like and dissimilar content, structure, or record length, and can isolate changes and generate a difference report.

Release 8.2.2 is also designed to improve the ease of use and efficiency of the existing copybook parsing utility. This lets users define the data for comparison by generating keywords and options directly from copybook field definitions. Besides MVS PDSs, users can now directly access CA-Panvalet or CA-Librarian copybooks when using the parsing utility.

For further information contact:
Serena Software International, 500 Airport Boulevard, Second Floor, Burlingame, CA 94010-1904, USA.
Tel: (650) 696 1800.
Serena Software International (UK), The Courtyard, 60 Station Road, Marlow, Bucks, SL7 1NX, UK.
Tel: ((01628) 481200.
URL: http://www.serena.com.

* * *

Xephon has just launched four weekly news services covering the following subject areas:

- Data Centre
- Distributed Systems
- Networks
- Software

Each week, subscribers receive, by e-mail, a short news bulletin consisting of a list of items; each item has a link to the page on our Web site that contains the corresponding article. Each news bulletin also carries links to the main industry news stories of the week.

To subscribe to one or more of these news services, or review recent articles, point your browser at http://www.xephon.com/newz.html.

* * *

xephon