BEBO WHITE

*Stanford Linear Accelerator Center,
Stanford University, Stanford, CA 94309*

Using Fullscreen CMS at CERN

Abstract

Fullscreen CMS is an optional console environment introduced in Release 5 of CMS which maintains the context of a VM session across invocations of full screen commands like XEDIT, FILLIST or MAH. In addition it allows limited scrolling and windowing capabilities.

This write-up provides CERNVM users who are interested in Fullscreen CMS with an overview of the concepts and operations which are involved. In that it is an optional environment, this write-up does not constitute an endorsement of Fullscreen CMS.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Contents

Abstract	i
Chapter 1: What is Fullscreen CMS?	1
Chapter 2: Documentation	3
Chapter 3: Getting In and Getting Out	3
Chapter 4: Windows and Virtual Screens	4
Chapter 5: When Windows POP...	4
Chapter 6: Tools in Fullscreen CMS	5
Chapter 7: Necessary Tools for the XEDIT Environment	6
Chapter 8: The Mysterious WM Window	6
Chapter 9: Suggestions and Examples	6
9.1 Possible Inclusions for a PROFILE EXEC:	7
Appendix A: Summary of CMS Windowing Commands	7

Figures

1. The Default Fullscreen CMS Environment	1
2. Example of CMS Operation within the Fullscreen Environment	2
3. Example of a "Popped" MESSAGE Window	3
4. Dropping a MESSAGE Window Using a Border Command	5

1. What is Fullscreen CMS?

Fullscreen CMS is an optional console environment introduced in Release 5 of CMS which maintains the context of a VM session across invocations of full screen commands like XFDT, FILELIST or MAIL. When returning from these commands, the results of previously executed CMS commands remain as a visible history of the operations performed prior. This history, like a console log, can be scrolled both backwards and forwards. Prior to Fullscreen CMS, this capability was not available.

The format of the Fullscreen CMS environment is similar to that of XFDT, having a command line and PF keys defined to perform common manipulations of the session file. The default format for this environment is shown in Figure 1. Commands entered from the command line are echoed in the screen followed by the CMS response to the command. Commands may also be entered in the scrollable area of the CMS Virtual Screen (defined below), or old commands may be edited; when ENTER is pressed the resulting "screen" command will be executed. All material in the screen may be simultaneously logged in a file (called CMS LOGFILE) which in some fashion replaces "spooled CONSOLE" files-- prior to Release 5 the only method of recording a terminal session. Figure 2 on page 2 illustrates the display of the CMS LISTFILE command when issued in the fullscreen environment

Figure 1: The Default Fullscreen CMS Environment

A Virtual Screen (VSCREEN) is a "presentation space" for data, specifically program input and output. It is of fixed size, and when new data is added at the bottom, old data "scrolls" off the top. Virtual Screens may also contain "reserved" areas at the top and bottom where titles or PF key definitions may be displayed.

A Virtual Screen is viewed through a Window which is a defined area on the physical screen of the terminal. Windows may have borders around them, and their size may be fixed or vary (up to some maximum) depending on the amount of data in the Virtual Screen. How much of the "presentation space" is visible depends on the size of the Window and the number of "reserved" lines in the

Figure 2: Example of CMS Operation within the Fullscreen Environment

Virtual Screen. Multiple windows may view the same Virtual Screen.

In addition to the CMS Virtual Screen (which is being viewed in this environment through the CMS Window), other Virtual Screens and Windows are defined by default: MESSAGE, NETWORK, WARNING, STATUS, etc. When there is no activity in these windows, they remain out of view, "below" the main CMS Window. When they become active (e.g., by receiving a message from another Virtual Machine or a file from RSCS), they may "pop" into view, overlaying a portion or all of the CMS window. Figure 3 on page 3 illustrates a "popped" Message Window on the CMS window of Figure 2.

To manipulate and control this full screen environment many new CMS commands have been introduced in Release 5 (e.g., SCROLL, DROP WINDOW, POP WINDOW, etc.). This document discusses some of the more useful of these commands which can make the full screen environment more comfortable for users who wish to take advantage of the continuous context provided by Fullscreen CMS. In addition there is a class of commands for writing full screen application programs (commands like DEFINE WINDOW, DELETE VSCREEN, WRITE VSCREEN, etc.). Using these commands is beyond the scope of this document. Lists of both classes of commands are provided in Appendix A to guide users through the wealth of new material.

Figure 3: Example of a "Popped" MESSAGE Window

2. Documentation

More detailed information about the full screen environment can be found in the CMS User's Guide for Release 5 of CMS. Chapter 9 of that document gives a general introduction to Windows and Virtual Screens. Chapter 15 discusses customizing this environment to specific tastes. Appendix D gives an overview of Fullscreen CMS, with emphasis on its effect on EXECs and views on writing applications.

3. Getting In and Getting Out

The Fullscreen CMS environment is entered by issuing the command:

```
SET FULLSCREEN ON
```

(No abbreviation is permitted for the word FULLSCREEN.) PF3 is used to exit this environment. This is equivalent to the command

```
SET FULLSCREEN SUSPEND
```

In the suspended state, all Virtual Screens remain intact. If Fullscreen CMS is re-entered (via either SET FULLSCREEN ON or SET FULLSCREEN RESUME), the context is preserved at the point of suspension.

The command

```
SET FULLSCREEN OFF
```

returns to "normal" CMS without saving any of the Fullscreen environment.

4. Windows and Virtual Screens

One may view different portions of the Virtual Screen by using the SCROLL command. SCROLL positions the Window to different places within the Virtual Screen. This command has 9 possible arguments: BAcKward, FoRward, Up, Down, RiGht, LeFt, ToP, BotToM, and NeXt (capitalization indicates the minimum abbreviation). The spatial context of these commands (BAckward, etc.) is the same as that used in XFEDIT. The following are examples of the use of these commands:

```
SCROLL DOWN CMS 3
(moves the CMS Window down three lines in its Virtual Screen)
```

```
SCROLL FORWARD =
(moves the "topmost" window, denoted by '=', one window
forward in its Virtual Screen).
```

Different windows may be brought into view with the POP WINDOW command. If the new window is smaller than the one presently in view, it will appear as an overlay. Conceptually one can imagine all windows as being stacked one on top of the other. The ones visible are those at this top of this stack (not to be confused with the CMS stack). The POP command may bring a window in the stack to the top (i.e., make it visible).

The DROP WINDOW command has the opposite effect. It moves windows down in the stack, and as a result they may disappear from view.

A parameter in both the POP WINDOW and DROP WINDOW commands controls how far a window will move relative to its present position. If this parameter is omitted, the window POPs all the way to the top or DROPs all the way to the bottom.

5. When Windows POP...

Frustrations can occur in Fullscreen CMS when windows unexpectedly "pop" and obscure part of the CMS Window or part of an XFEDIT file. Fullscreen CMS provides a collection of tools specifically for controlling "popped" windows. In the following section are suggested PF key settings which users can define in their PROFILE XFEDIT to make XFEDIT less vulnerable to this problem.

6. Tools in Fullscreen CMS

The PA2 key provides the most immediate relief in Fullscreen CMS. This key is used by Fullscreen CMS to scroll forward the "topmost" window (i.e., the one which just intruded upon the terminal session.) When the "popped" window has been scrolled to the end of the data in its Virtual Screen, it disappears from view. Thus one can scroll through the material which caused the screen to "pop" into view and then have the window vanish.

A second method (which works from any full screen environment) is to issue the command

DROP WINDOW =

This causes the "topmost" window (here denoted by '=') to drop from "topmost" importance and hence vanish "below" the window you were viewing.

A final method is to use a so-called "border" command. If the "popped" window has a border around it, the window can be manipulated by positioning the cursor on one of the corners of the border and typing a valid, one-letter border command (D="drop window", F="scroll forward", etc.--the complete set of border commands is summarized in Appendix D of the CMS User's Guide). When ENTER is pressed, the window will then respond to the border command and be dropped or scrolled accordingly. Figure 4 illustrates the dropping of a MESSAGE Window which has "popped" into view on the CMS Window in Figure 3 on page 3. After this command is executed, the CMS Window will return to the state as in Figure 2 on page 2.

Figure 4: Dropping a MESSAGE Window Using a Border Command

7. Necessary Tools for the XEDIT Environment

With Fullscreen CMS in effect, XEDIT is simply another full screen application which defines a Virtual Screen named XEDIT and a Window named XEDIT within which it displays (using CMS Windowing Commands) the file being edited. One can handle a "popped" window in this context by using border commands, as described above, or by issuing CMS window commands from the XEDIT command line. However, the most natural method, supplied by PA2 in Fullscreen CMS requires that the user set a PF key (or perhaps even PA2) to issue the command 'DROP WINDOW = '. This can be done in PROFILE XEDIT, thereby becoming a permanent part of a user's XEDIT environment.

Which key to set is a matter of personal taste. Many users don't use PF13 through PF24, so these are potential candidates. For people who set NULLS ON or who use ASCII terminals in emulation mode, PA2 is an excellent candidate. Once the choice is made, one simply inserts a line like 'SET PA2 BEFORE DROP WINDOW = ' into their PROFILE XEDIT.

Many users may want to set a second key to issue the command 'SCROLL FORWARD = '. This can be useful in situations where the user wants to scroll through the "popped" window rather than perfunctorily dropping it. This setting exactly mirrors the functionality of PA2 in Fullscreen CMS.

Users may also want to alter their RDRLIST or FILELIST profiles in a similar fashion if they use these full screen applications extensively.

8. The Mysterious WM Window

Users may occasionally encounter the Window Manipulating (or WM) window while using Fullscreen CMS. Although it may be invoked deliberately with the command 'POP WINDOW WM', it may also occur spontaneously when a window accidentally overlays all possible command areas in XEDIT or Fullscreen CMS. The WM window serves as an "escape valve" for situations where no area on the screen can be used for entering commands.

The WM window appears as a small window at the bottom of the physical screen containing PF key definitions, a command line, and the message to "Enter a windowing command or press a PF key". The correct response is usually to drop the window which caused the fatal overlay ('DROP WINDOW = ' will suffice) or use PF8 to scroll to the end of the offending window and make it vanish. Pressing PF3 will then remove the WM window and return to the previous environment.

9. Suggestions and Examples

Below are some commands which could be placed in a PROFILE EXEC. Finally, there is an EXEC example which scrolls a window half a screen forward or backward.

9.1 Possible Inclusions for a PROFILE EXEC:

The following statements are examples of how the Fullscreen environment may be tailored from a user PROFILE EXEC.

```

/* Redefine the size and position of the MESSAGE window and
Virtual Screen so they land more aesthetically on a 43 line
terminal
*/

```

```

'DEFINE VSCREEN MESSAGE 40 70 2 0 (PROTECT'
'DEFINE WINDOW MESSAGE 33 71 -6 3 (VARIABLE BORDER POP
TOP'

```

```

/* Initiate Fullscreen CMS
*/

```

```

'SET FULLSCREEN ON'
'CP TERM BRKIFY PA1' /* FULLSCREEN ON sets it to NONE.
*/

```

```

/* Turn on a LOGFILE for the CMS Virtual Screen and name
the file by date
*/

```

```

'SET LOGFILE CMS ON' date('Sorted') 'LOGFILE A'

```

```

/* Enlarge the CMSOUT window to fill the whole screen
(CMSOUT is the window XEDIT uses to display the CMS
Virtual Screen when command responses from CMS occur
in XEDIT)
*/

```

```

'MAXIMIZE WINDOW CMSOUT'

```

```

/* Set the less useful CMS PF keys to something personally
useful
*/

```

```

/* Establish a PF key for dropping "popped" windows
*/

```

```

'SET CMSPF 18 Clear_Top NOECHO #WM CLEAR WINDOW ='

```

EXEC example:

```
HAFSCROL EXEC:  called as HAFSCROL FORWARD or HAFSCROL BACKWARD

/* EXEC to scroll the "topmost" window up or down half a window */
   arg direction .

/* Find window dimensions */
   address command 'QUERY WINDOW = (STACK'
   pull . . size .

/* Scroll in correct direction or issue a syntax message */
select
  when abbrev('FORWARD',direction,1)=1 then do
    address command 'SCROLL DOWN =' size%2
    end
  when abbrev('BACKWARD',direction,1)=1 then do

/* Maintain unitarily so FORWARD followed by BACKWARD
   lands you where you started for an odd sized screen */
   size=size-size%2
   address command 'SCROLL UP =' size
   end
otherwise do
  say "Invalid scrolling direction:" direction
  exit 24
end
end /* select */
exit
```

Appendix A

Summary of CMS Windowing Commands

Windowing commands can be categorized roughly into three groups:

1. those necessary to use Fullscreen CMS,
2. those which can occasionally be quite useful, and
3. those needed only by persons wanting to write full screen applications.

For categories 1) and 2), the following is a terse description of the command. The command HELP file and/or the CMS Command Reference manual should be consulted for the complete description. For completeness, category 3) commands are listed without description.

- **Necessary Windowing Commands**

DROP WINDOW	Move a window behind the others.
POP WINDOW	Move a window in front of others.
SCROLL	Move a window up, down, left, or right in its Virtual Screen.

- **Useful Windowing Commands**

CLEAR WINDOW	Scroll past all data in a window.
PUT VSCREEN	Write the data in a Virtual Screen to a file.
MAXIMIZE WINDOW	Expand a window to the physical size of the screen.
MINIMIZE WINDOW	Reduce the size of a window to one line.
POSITION WINDOW	Change the location of a window on the physical screen.
QUERY WINDOW	Find characteristics of the Windowing environment.
QUERY VSCREEN	
QUERY CMSPF	
QUERY FULLREAD	
QUERY LOGFILE	
QUERY FULLSCREEN	
RESTORE WINDOW	Return a maximized/minimized window to its previous size.
SET FULLSCREEN	Invoke or suspend the FULLSCREEN CMS.

SET LOGFILE	Establish or discontinue logging of a Virtual Screen into a LOGFILE.
SET CMSPF	Place a command on a CMS PF key.
SIZE WINDOW	Change the size of a window.

- **Advanced Windowing Commands**

ALARM SCREEN	DELETE WINDOW	SHOW WINDOW
CLEAR VSCREEN	GET VSCREEN	WAITREAD VSCREEN
CURSOR VSCREEN	HIDE WINDOW	WAITT VSCREEN
DEFINE VSCREEN	PUT SCREEN	WRITE VSCREEN
DEFINE WINDOW	REFRESH	
DELETE VSCREEN	ROUTE	

and a horde of SET and QUERY commands