HEWLETT PACKARD

HP Visual User Environment

System Administration Manual

INDUSTRIAL DESIGNERS SOCIETY OF AMERICA
IDEA
INDUSTRIAL DESIGN EXCELLENCE AWARD
GOLD

HP Visual User Environment

System Administration Manual

HP Part No. B1171-90044
Printed in U.S.A. November 1991

Edition 2

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MANUAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Printing History

The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive these updates or new editions, see your HP sales representative for details.

November 1991 ... Edition 2

Interface Technology Operation
1000 N.E. Circle Blvd.
Corvallis, OR 97330

Contents

1

Introduction

This chapter describes:

- The contents and organization of this manual.
- The typographical conventions used.
- Other documentation you may need to read.

Should You Be Reading this Manual?

The primary audience for this manual is system administrators of systems running the HP Visual User Environment (HP VUE). Since HP VUE provides many interactive customization capabilities, most end-users do not need to know the system in the detail described here.

However, there are some things that cannot be done interactively that an end-user might some day need to do—for example, converting a screen dump from one graphics format to another, or adding a keybinding. These things aren't particularly hard to do; they're just not done very often by typical users, and they are covered in this manual rather than in the *HP Visual User Environment User's Guide*. So, if you are not a system administrator but nevertheless have found yourself faced with having to read a section of this manual, don't panic.

Note

System administration information for systems running the X Window System without HP VUE is contained in the manual *Using the X Window System*, HP part number B1171-90043.

How this Manual Is Organized

Chapter	Contents
1	Introduces this manual and describes the other documentation available to you.
2	Describes the hardware and software that make up a typical system and explains how they fit together.
3	Covers configuring the system for multiple screens and how HP VUE implements Native Language Support.
4	Covers how HP VUE implements distributed operation.
5	Covers the HP VUE Login Manager, which is responsible for starting the X server and displaying the login screen.
6	Covers the HP VUE Session Manager, which provides the ability to save and restore previous sessions.
7	Describes how to construct command lines to execute programs locally or remotely.
8	Explains how clients obtain resources and how to modify the resource database in an environment of managed sessions.
9	Introduces the HP VUE clients, explains how HP VUE manages color resources, and explains client caching.
10	Describes how HP VUE implements fonts.
11	Describes the HP VUE Window Manager, including menus, button bindings, key bindings, and workspace backdrops.
12	Describes the file types and action definition databases that provide the ability to start processes from the file manager and workspace manager.
13	Explains how to customize the HP VUE Workspace Manager (front panel).
14	Discusses how to run sessions from a startup script and how to run a non-HP VUE environment from a system console.

15 Covers compatibility and porting issues for Softbench and HP VUE.

Typographical Conventions

Table 1-1. Typographical Conventions.

If you see ...	It means ...
`computer text`	File names, directory names, displayed text, or text that you type exactly as shown. For example, **`inittab`** is a file name and **`$HOME/.vue/types`** is a directory path.
italic text	A book title, emphasized text, or a parameter for which you must supply an allowable entry. For example, **`xterm -title`** *name* means you type **`xterm -title`** followed by a name.
▢	Represents a key on the keyboard. For example, CTRL L means you hold down the CTRL key and press L.
[ ]	Brackets enclose optional text, usually parameters or command-line options. For example, **`xload [-rv] &`** means that you must type **`xload`** but don't have to type **`-rv`** unless you need its particular functionality.
{ }	Braces enclose a list of *mutually exclusive* options. For example, **`xset r`** { **`on`** / **`off`** } means that option **`r`** can be set to either **`on`** or **`off`**, but not both.
bold text	The definition of this term follows. Additionally, the term may be defined in the glossary.

Case Sensitivity and Other Typographical Tips

Most systems distinguish between upper-case and lower-case letters. A file named **vuewmrc** is *not* the same file as **Vuewmrc**. Use upper-case letters only where indicated.

Beware that the number "1" looks like a lower-case letter "l" (el), and that "0" (zero) closely resembles the upper-case letter "O".

White space (extra spaces or tabs) at the end of a command line in a text file sometimes alters the meaning of the command. After modifying a file, check for unwanted white space.

Where to Find Additional Information

To order any of the manuals listed here, call:

- HP Direct at 1-800-538-8787.
- Your HP Sales Representative.

and ask for the part numbers listed for the manuals.

The following books provide additional information about HP VUE.

To Learn More About ...	Read ...	Part Number
Preliminary configuration and updating to HP VUE 2.01 on HP-UX 7.* systems.	*HP Visual User Environment Installation Guide*	B1171-90006
Preliminary configuration for HP VUE on HP-UX 8.* systems (including 8.0).	*HP Visual User Environment Configuration Guide*	B1171-90041
Preliminary configuration for HP VUE on other operating systems.	*HP Visual User Environment Configuration Guide* for your operating system.	—
Using HP VUE	*HP Visual User Environment User's Guide*	B1171-90042
Writing applications for HP VUE	*HP Visual User Environment Programmer's Guide*	B1171-90024
Running the X Window System without HP VUE; reference pages for the X Window System	*Using the X Window System*	B1171-90043

These books provide additional information about HP OSF/Motif, the X Window System, widgets, and widget programming:

HP OSF/Motif Style Guide	B1171-90032
HP OSF/Motif Programmer's Reference	B1171-90033
HP OSF/Motif Programmer's Guide	B1171-90034
The X Window System: OSF/Motif Edition	ISBN 0-13-497074-8
Mastering OSF/Motif Widgets	5010-7168

The following books about the X Window System and OSF/Motif might prove useful:

- *Introduction to the X Window System* by Oliver Jones. Prentice Hall, Englewood Cliffs, NJ:1989.
- *The X Window System in a Nutshell*. O'Reilly and Associates, Newton, MA:1990.
- *Xlib Programming Manual for Version 11* by Adrian Nye. O'Reilly and Associates, Newton, MA:1990.
- *Xlib Reference Manual for Version 11* edited by Adrian Nye. O'Reilly and Associates, Newton, MA:1990.
- *X Window System User's Guide OSF/Motif Edition* by Valerie Quercia and Tim O'Reilly. O'Reilly and Associates, Newton, MA:1991.
- *X Protocol Reference Manual*O'Reilly and Associates, Newton, MA:1990.
- *X Window Systems Programming and Applications with Xt* by Douglas A. Young. Prentice Hall, Englewood Cliffs, NJ:1990.
- *OSF/Motif User's Guide*. Prentice-Hall, Englewood Cliffs, NJ:1990.
- *Advanced X Window Applications Programming* by Eric F. Johnson and Kevin Reichard. MIS Press, Portland, OR:1990.

File Locations on OSF/1 and Domain/OS Systems

There are several fundamental differences in the file locations on HP-UX, Domain/OS, and OSF/1 systems that affect HP VUE.

For example, the following X11 system directories have different locations on OSF/1 and Domain/OS systems than on HP-UX systems.

HP-UX Location	OSF/1 and Domain/OS Location
`/usr/lib/X11`	`/usr/X11/lib`
`/usr/bin/X11`	`/usr/X11/bin`

The HP VUE documentation uses the HP-UX paths.

On OSF/1 and Domain/OS systems, HP VUE uses links between the HP-UX directories and their corresponding OSF/1—Domain/OS directories. The actual files reside in their OSF/1—Domain/OS locations.

For example, an OSF/1 or Domain/OS user executing:

```
cd /usr/lib/X11
ls
```

sees the contents of `/usr/X11/lib`.

Note

For Domain/OS Users: The links set up for the X11 system directories use the absolute paths to the local system files. For example, executing:

```
cd //sysb/usr/lib/X11
ls
```

on local system `//sysa` displays the contents of `//sysa/usr/X11/lib`.

2

Birdseye HP VUE

This chapter describes:

- The major layers of software on a typical system running HP VUE.
- A brief description of the HP VUE software.

The Layers of Software on HP VUE Systems

The software that provides the appearance and behavior of HP VUE has three layers.

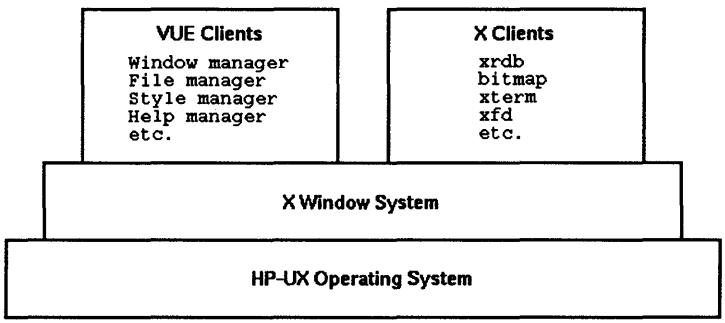


Figure 2-1. Software Layers.

To an end-user, the layers blend together into a single working environment. However, from a system administration point of view, it is important to know how the layers work together.

2

The Operating System

The operating system is the software that controls the operation of the computer system. The operating system provides a multi-user, multi-tasking environment.

HP VUE runs on a variety of operating systems. Most of the appearance and behavior, and many of the configuration issues, are the same for the various operating systems. Major differences are covered in the *HP Visual User Environment Configuration Guide* for your operating system. Smaller differences are noted in the other HP VUE documents.

The X Window System

The central part of the X Window System is the **X server**, also called the **display server**. The X server is the program that controls the screen, keyboard, and mouse, and processes communication requests. The server updates the windows on the screen as a client generates new information or as you enter information through an input device. All client programs communicate through the server.

The X clients are programs designed to run under the X Window System, with or without HP VUE.

There are a number of clients that are included with the X Window System. For example, the `bitmap` client creates custom bitmaps. The `xrdb` client lets you view and modify current resources.

Some clients (for example, `xwininfo` and `xmodmap`) do not create windows. They use an existing terminal emulation window to display their output.

The X clients are described in the manual *Using the X Window System*.

HP VUE

HP VUE is a set of enhancements to the user interface for the X Window System. It consists of a series of clients that provide a visual interface for logging in and out, invoking applications, managing files, and customizing systems.

The HP VUE Clients

To the system administrator, HP VUE is a set of clients and associated configuration and resource files. These are the major components of HP VUE:

Table 2-1. Major Components of HP VUE.

Component	Client	Description
Login manager	**vuelogin**	Performs configuration activities, starts the server, and performs the login activities.
Session manager	**vuesession**	Saves and restore sessions.
Window manager	**vuewm**	Provides window and workspace management.
Workspace manager (front panel)	—	Part of **vuewm**. Provides a panel of controls for managing workspaces and sessions. Also called the front panel.
File manager	**vuefile**	Graphical interface for file management.
Style manager	**vuestyle**	Lets the user customize the appearance and behavior of the system interactively.
Help manager	**vuehelp**	Provides on-line instructions and context-sensitive help.
Terminal emulator	**hpterm** or **xterm**	Provides the ability to emulate terminals. The default terminal emulator for HP-UX is **hpterm**. For other operating systems, the default is **xterm**.
vuepad editor	**vuepad**	Provides text editing.

HP VUE clients are able to communicate with the portions of HP VUE that control certain aspects of the appearance and behavior. For example, viewable HP VUE clients (clients that have their own windows) respond to color changes made using the HP VUE Style Manager.

The Window Manager—A Special Client

If you think of your computer screen as a surface on which things are displayed, you can think of workspaces as a stack of surfaces. At any one time, only one workspace is visible, and it fills the entire screen. A control switch in the workspace manager (also called the front panel) lets you switch from the current workspace to another one in the stack.

Each workspace can contain a number of windows. In functionality, each window is equivalent to a full-sized display terminal and can run a separate process.

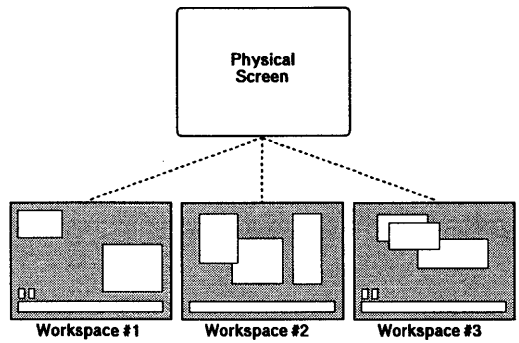


Figure 2-2. Windows in Workspaces.

The window manager lets you establish one or more workspaces and dynamically control the size, shape, state (icon or normal), and location of the windows in those workspaces. It also supplies the frame and window menu for each window, the workspace manager, and the workspace menu.

The window manager is the first client started during a session after the session manager has been started. All other clients with their own windows must be able to interact with the window manager.

The window manager is covered in chapter 11.

Other HP VUE Components

HP VUE includes various other components that contribute to its functionality:

File Management Database	Provides a database of file types and action definitions which allows the user to invoke commands and applications graphically.
Broadcast Message Server	Provides the ability to exchange certain information between HP VUE clients
Command Invoker	Provides the ability to invoke operating system commands on local and remote hosts.

3

Preliminary Configuration

This chapter covers some of the preliminary configuration you must do before starting the X server and HP VUE. It includes:

- Using custom screen configurations.
- Configuring the system for special input devices.
- Using Native Language Support.

Note

Information in this chapter may not apply to all operating systems. See the notes at the beginning of each topic discussion for more information.

There are two other chapters that deal with various aspects of initial configuration:

- Chapter 4 discusses how HP VUE provides for distributed operation.
- Chapter 5 covers the Login Manager, which is responsible for starting the X server, displaying the login screen, and validating the user. Multiple displays are configured by the Login Manager.

Using Custom Screen Configurations

This section covers the default screen configuration and custom configurations for HP VUE.

Note

Information in this section is *not* applicable to Domain/OS systems.

3

Default Configuration

The default screen configuration for HP VUE is specified in `/usr/lib/X11/X0screens`. It assumes:

- There is one display—display 0.
- There is one screen—screen 0.
- The screen uses only one set of pixel planes (the image planes).
- The screen is at the address node specified by `/dev/crt`.

If you use some configuration other than the default, you must edit the default screen file or add additional screen configuration files.

There should be a separate `X*screens` file for each display, where * is a number that matches the display number used when starting the X server.

Creating a Custom 'X*screens' File

There are two ways to create a custom screen configuration for a display:

- You can modify `X0screens` so that it contains device information for all the screen configurations you may want to use. This is generally the preferred way. Only one configuration is used at a time; the others are commented out. To switch from one screen configuration to another, you uncomment some lines and comment others. For multiple displays, you would have a separate file for each display—for example, `X1screens` for display 1.
- You can have a separate `X*screens` file for each screen configuration on a particular display. Switching between them involves modifying the command that starts the X server.

Screen modes and the syntax of the **X*screens** files is covered in the *Using the X Window System* manual.

Running HP VUE in a Multiple Screen Environment

HP VUE supports multiple screens in these ways:

- The HP VUE window manager can be started on multiple screens. Screen-specific resources can be specified for window manager resources.
- The session manager will save screen-related information about clients, and will restore clients to their proper screens at the beginning of the next session.

Multiple screens are not supported by the following portions of HP VUE:

- Only one workspace manager is allowed per X server (display). It will be displayed on the default screen, or on the screen specified in the command line that starts **vuewm** (see "Using the Window Manager with Multiple Screens" in chapter 11).
- The session manager can run on only one screen. (However, it can save clients running on other screens.)
- The Broadcast Message Server and Command Invoker will run only on the screen running the session manager. This affects communication between clients, and may produce unpredictable behavior. For example, clients invoked from the file manager on screen 1 may display on screen 0.

In general, HP VUE should be run on one screen only; use other screens for various non-HP VUE environments such as Starbase.

Making 'X*.hosts' Files for Special Configurations

The `X*.hosts` files, if present, are used by HP VUE the first time HP VUE is run. The `X0.hosts` file contains a list of all remote hosts initially permitted to access your local server. The default screen configuration file `X0screens` uses the default remote host file `X0.hosts`. Each custom `X*screens` file uses a corresponding `X*.hosts` file.

The HP VUE Session Manager maintains a file, `vue.settings`, that saves and restores host information for sessions. Once this file exists (the user has logged in and out), the `X*.hosts` files are no longer used, and the user should use the Hosts dialog box in the style manager to edit the list of hosts.

The `X*.hosts` file is covered in chapter 4.

Using Special Input Devices

HP VUE has an input device file that the X server reads to find out what input devices it should open and attach to the display. The default input device configuration file is `/usr/lib/X11/X0devices`.

Customizing `X*devices` files is covered in the *Using the X Window System* manual.

Running Starbase Applications

Note This section does *not* apply to Domain/OS systems.

Allocating Colors for Starbase

Starbase applications can create their own colormap. However, if a Starbase application uses the default colormap, then colors must be allocated to it before an HP VUE session is run. The colors allocated to Starbase are not available to HP VUE.

To allocate the Starbase colors, you must run the X client **xinitcolormap** before starting an HP VUE session. Do do this:

1. Edit the file **/usr/lib/X11/vue/Vuelogin/Xsession** to include a line that runs **xinitcolormap**. This file is a shell script that is run by the Login Manager right before it invokes the session manager. (The line is commented out in the default **Xsession** file.) Use the **-f** option to specify the file containing the colors.
2. Create the file containing the colors to be allocated to Starbase.

Starbase applications do not communicate with the HP VUE color server.

Environment Variables for Starbase Applications

Most environment variables for Starbase applications can be set in **.vueprofile**. However, two variables must be set before the X server is started—**SB_DISP_ADDR** and **WMSHMSPC**. Meaningful defaults are provided for these variables. If you must change them, use the **vuelogin** resource **environment**. The resource is added to the file **/usr/lib/X11/vue/Vuelogin/Xconfig**.

For example, the following line specifies values for the two variables:

```
Vuelogin*environment:  SB_DISPLAY_ADDR=0xB00000 \
                       WMSHMSPC=0x200000
```

Customizing HP VUE for Native Language Support

This section covers:

- How HP VUE sets the **LANG** environment variable and other environment variables.
- Accessing language-dependent message catalogs and resource files.
- Remote execution in systems on which Native Language Support (NLS) is available.

Setting the LANG Environment Variable

The **LANG** environment variable must be set in order for HP VUE to use native language support. Setting **LANG** causes HP VUE to use the operating system's language-sensitive routines for character handling.

You can set **LANG** to any value supported by the operating system. The NLS directory **/usr/lib/nls** (**/usr/nlslib** on Domain/OS) contains an entry for each supported value. The HP VUE message catalogs are located in subdirectories of the NLS directory.

There are three ways to set **LANG** for HP VUE:

- By a **vuelogin** resource in the **Xresources** file.
- From the Options menu in the login screen.
- In the **.vueprofile** file.

Table 3-1. Effects of Setting LANG Different Ways.

Method	Localized login screen?	Effect on HP VUE Session
vuelogin resource	Yes	**LANG** set for all users.
Login screen Options menu	Yes	**LANG** set for all users.
.vueprofile file	No	**LANG** set for single user.

Setting the Language with a Vuelogin Resource

Setting the language by means of a **vuelogin** resource has these effects:

- The **vuelogin** client reads the appropriate message catalog for that language and brings up the localized login screen.
- The **LANG** environment variable is set to that language for HP VUE sessions for all users. (**LANG** can be set for individual users by an entry in **.vueprofile**.)
- The resource should set the **NLSPATH** environment variable appropriately for the chosen language. If this is not the case, or if you want to set **NLSPATH** yourself, see "Message Catalogs—the NLSPATH Environment Variable" later in this chapter.

The resource is set by placing the following line in **/usr/lib/X11/vue/Vuelogin/Xconfig**:

```
Vuelogin.host_display.language: language
```

For example, the following line sets **LANG** to Swedish on display hpcvxpae:0.

```
Vuelogin.hpcvxpae_0.language: swedish
```

Setting LANG from the Options Menu

Setting the lanugage from the login screen Options menu causes HP VUE to use that value of **LANG** for the next session. **LANG** returns to its default value at the conclusion of the session.

Setting LANG in .vueprofile

User-specific environment variables for HP VUE sessions can be placed in the file **$HOME/.vueprofile**.

- If you use **sh** or **ksh**:

```
LANG=language
export LANG
```

- If you use **csh**:

```
setenv LANG language
```

Other NLS Environment Variables

This section covers other NLS environment variables. These variables are not affected by the `vuelogin` language resource nor by the login screen option menu.

3

- Set them in `/usr/lib/X11/vue/Vuelogin/Xsession` to set system-wide variables.
- Set them in `$HOME/.vueprofile` for user-specific variables.

In the following examples, `%L` is translated by the system into the value of the `LANG` variable; `%N` is translated into the value of the application's class name.

Message Catalogs—The NLSPATH Environment Variable

The `NLSPATH` environment variable determines the paths applications search for NLS message catalogs. HP VUE clients place NLS message catalogs in the NLS directories `/usr/lib/nls/$LANG` (non-Domain/OS systems) or `/usr/nlslib` (Domain/OS). Both `LANG` and `NLSPATH` must be set in order to use those message catalogs.

The proper value of `NLSPATH` depends on whether message catalogs exist for the current value of `LANG`.

- To use the message catalogs for the language to which `LANG` is set, set `NLSPATH` to:

  ```
  /usr/lib/nls/%L/%N.cat:/usr/lib/nls/C/%N.cat:$NLSPATH
  ```

- If no message catalogs exist for the language to which `LANG` is set, set `NLSPATH` to:

  ```
  /usr/lib/nls/C/%N.cat:$NLSPATH
  ```

Setting Language-Depended App-Defaults

The default location for the `app-defaults` for HP VUE and X clients is `/usr/lib/X11/%L/app-defaults`, where `%L` is the the value of `LANG`. For example, if `LANG` is set to `swedish`, then applications will look for their `app-defaults` in

`/usr/lib/X11/swedish/app-defaults`. If `LANG` is not set, `%L`

is ignored, and applications look for their `app-defaults` in `/usr/lib/X11/app-defaults`.

The `XUSERFILESEARCHPATH` environment variables provides the ability to change the location of `app-defaults`. For example, if it were desirable to move `app-defaults` to `/users`, then `XUSERFILESEARCHPATH` could be set to `/users/%L/app-defaults`.

If you set `XUSERFILESEARCHPATH` in `$HOME/.vueprofile`, the value applies to all HP VUE and X clients you run. Non-clients will not find their resource files unless you link or copy them into the directory specified by `XUSERFILESEARCHPATH`.

For additional information about localized resources, refer to the discussions of the resource database in the Xt Intrinsics documentation.

Setting the KBD_LANG Environment Variable

Note

The ability to support native language keyboards depends on the operating system. None of the information in this section applies to to Domain/OS systems. Other operating systems may not support character sets for Asian languages.

HP VUE allows you to override the physical keyboard attached to the HP-HIL. For example, a USASCII keyboard with `KBD_LANG` set to `czech` will behave like a Czech keyboard.

KBD_LANG can be set after the server has started. A convenient place for setting the variable system-wide is in /usr/lib/X11/vue/Vuelogin/Xsession. The following lines set KBD_LANG appropriately for languages that require the keyboard to be remapped. Insert theses lines in Xsessions after the line export EDITOR LOGNAME MAIL TERM:

3

```
if [ ! -z "$LANG" ]
then
     case $LANG in

          bulgarian | czech  | hebrew | hungarian | \
          japanese  | korean | polish | rumanian  | \
          russian   | serbocroatian)
                     KBD_LANG=$LANG
                     export KBD_LANG;;
          chinese-t) KBD_LANG=t_chinese;;
                     export KBD_LANG;;
          chinese-s) KBD_LANG=s_chinese
                     export BKD_LANG;;
          *);;
     esac
fi
```

The NLIO processes for Asian users start only when either the physical keyboard is Asian or KBD_LANG is set to an Asian language.

Language-Dependent Bitmaps—the XBMLANGPATH Variable

The XBMLANGPATH variable specifies the search path for language-dependent bitmaps. It lists the paths for bitmaps in this order:

1. User-specific bitmaps.
2. System bitmaps listed in the XmGetPixmap(3x) man page.
3. Append:

   /usr/lib/X11/bitmaps/%N/%B

   This ensures that you will get the non-localized bitmaps, where necessary.

Other Language-Dependent Resource Files

When `LANG` is set, HP VUE uses the following language-dependent default resource files:

- `/usr/lib/X11/vue/Vuesession/%L/sys.session`.
- `/usr/lib/X11/vue/Vuesession/%L/sys.resources`.
- `/usr/lib/X11/vue/Vuewm/%L/sys.vuewmrc`.

Editing in HP VUE

All characters that can be entered from an HP keyboard into a terminal emulator window can also be entered into HP VUE text entry areas.

Remote Execution and NLS

You can invoke localized HP VUE applications on any remote execution host that has a similarly-configured localized HP VUE installation. The values of the NLS-related environment variables on the host that is invoking the application are passed to the remote host when the application is started. However, the environment variables do not contain any host information. Thus, the message catalogs and application resource files must be in the same locations on both systems unless the `$HOME/.softenv` file on the remote host specifies the location of the files on the remote system.

4

Distributed Operation

This chapter describes how to configure systems so that you can use HP VUE to run applications and access data in a distributed computing environment—that is, and environment where the application and/or data exist elsewhere than on the local workstation.

Note

This chapter does *not* apply to Domain/OS systems. For information about distributed operation in HP VUE running on Domain/OS systems, see the *HP Visual User Environment Configuration Guide for Domain O/S Systems*.

Basic Concepts

HP VUE supports several different types of distributed operations in addition to the standard X windows distributed display capabilities:

- HP VUE can run on one host and invoke applications on a remote host.
- HP VUE can run on one host and access files on a remote host.
- HP VUE can invoke applications on a remote execution host and supply the application with files from a different remote data host.

Thus, when an application is running under HP VUE, there can be up to four different hosts involved:

- The host running the X server. This is the host which displays all the user's activity.
- The host on which HP VUE is running—the local host. This is usually the same as the host running the X server. However, for X-terminal configurations, the two are different.
- The host where the application is running—the execution host.
- The machine where the data files are located—the data host.

4

Accessing Remote Data

This section covers:

- How to specify data files for HP VUE.
- How HP VUE accesses data files.
- How to configure the system for remote data access.

Specifying Data Files

In HP VUE, data files are specified by the syntax:

[*hostname*:]*filename*

The *hostname* is optional when specifying a file located locally, but must be specified when the file is located on a remote host. For example, consider a user running an HP VUE session on host `pronto`. If double-clicking an action icon displays a prompt for a filename, you can specify the file `/tmp/data1` on `pronto` by entering either `/tmp/data1` or `pronto:/tmp/data1`. To specify a file on remote data host `scribe`, you must use the hostname qualifier `scribe:/tmp/data1`.

How HP VUE Accesses Files

If you specify a data file without a hostname qualifier, the file is assumed to reside on the local host, and HP VUE accesses the file using the normal operating system procedure.

Files that are specified with a hostname qualifier are translated into one of two forms:

- If the data host is local, the leading *hostname*: is stripped off. For example `hpcvxba:/tmp/data1` becomes `/tmp/data1`.
- If the data host is remote, *hostname*: is replaced with `/nfs/`*hostname*`/`. For example, `hpcvxba:/tmp/data1` becomes `/nfs/hpcvxba/tmp/data1`.

4

Configuring For Remote Data Access

To allow applications on one host (the execution host) to access files on a remote data host, the execution host must NFS-mount the data host file system in the following manner:

- The mount point for the remote file system must be:

  `/nfs/`*remote_hostname*

  If the data host is in a different domain than the execution host, you must used the fully-qualified domain name of the data host (for example, `pronto.dom.hp.com`). Refer to the networking documentation for your operating system for more information about domain naming.

- The following remote files systems must be mounted:
  - Root (the / directory).
  - The authentication directory for the remote system. For HP-UX, the authentication directory is `/tmp`.

  It is recommended that all file systems on the remote data host be mounted.

In addition, the system requires that files accessed by HP VUE have their file systems mounted. It is recommended that each user have the same user id on all machines accessed via NFS.

Data Access on Diskless Clusters

Note

This section applies only to operating systems that support diskless clusters.

The HP VUE remote file access mechanism works the same on clusters as it does on stand-alone systems, with the following exceptions:

- Only a cluster server can export its file system for NFS mounting.
- Context-dependent files must be given special consideration.

4

Explicit Specification of Context-Dependent Files

If a specific context-dependent file is needed, it can always be accessed using an explicit path (see the `cdf(4)` man page). For example specifying the context:

server`:/etc/inittab`

refers to `inittab` on the server machine. To access `inittab` for a particular node, specify:

server`:/etc/inittab+/`*node*

Accessing Systems in Remote Clusters

When data is needed from remote clusters, the cluster server must be specified as the data host. Under this configuration, context-dependent files default to the version of the server.

Accessing Remote Systems within the Same Cluster

HP VUE accesses files on all remote systems via NFS, including remote systems that are in the same cluster as the local host. In this case, context-dependent files default to the version for the server.

It is also possible to access other systems within the same cluster so that context-dependent files default to the version of the execution host. Both of these approaches produce the same result unless a context-dependent file is present in the path to the specified file.

Defaulting to the Server's Context-Dependent Files

If other machines in the same cluster are used as execution hosts, the cluster server should be a data host, and the server file system should be mounted on to itself.

For example, the root directory of *server* should be NFS mounted on:

server:`/nfs/`*server*

Defaulting to the Execution Host's Context-Dependent Files

Within a cluster, a symbolic link can be used instead of NFS mounting the file system onto itself. In this configuration, context-dependent files are accessed uing the context of the execution host rather than the server.

To create this symbolic link, execute:

`ln -s / /nfs/`*server*

As an example of this configuration, suppose `process1` is executing on `node1`, and `process2` is executing on `node2`. In `process1` and `process2`:

server:`/etc/inittab`

refers to *server*`/etc/inittab+/node1`, and *server*`/etc/inittab+/node2`, respectively.

Remote Execution

This section covers:

- HP VUE requirements for remote execution.
- Configuring HP VUE for remote execution.
- How to specify a remote execution host for an action in the action definition database.
- How the user's access rights are checked on the remote host.
- The environment under which remote applications are run.

Requirement for Access to a Remote Execution Host

HP VUE provides a component called the Sub-Process Control Daemon (SPCD or softspcd) to support remote execution. This component is part of the Broadcast Message Server (BMS). The BMS must be installed on any host that is used as a remote execution host for HP VUE.

When an HP VUE session on one host wishes to invoke an application on a remote host, HP VUE sends a message to the SPCD on the remote host that specifies the execution string for the application.

The remote SPCD is invoked through the standard **inetd** process, which also must be installed and correctly configured on the remote host. Normal installation procedures correctly install and configure **inetd**. (See the **inetd(1m)** man page for more information).

For security reasons, the SPCD does not allow root to perform remote execution; root can't perform the file-based authentication over NFS.

Configuring HP VUE for Remote Execution

In order for an HP VUE session on one host to invoke applications on another host, the following conditions must be met:

- HP VUE must be correctly installed on the local host.
- Either all of HP VUE, or just the BMS fileset, must be installed on the remote execution host.
- Each host must have correctly configured *hostname* and network address specified in their **/etc/hosts** files. Alternatively, the operating system name server mechanism may be used.
- The **/usr/adm/inetd.sec** file on the remote execution host must specify that the local host is allowed to connect to its SPCD service. This is accomplished by listing the local host using the syntax:

  spc allow ... *local_host* ...

  For more information about **inetd.sec**, see "Network Security" later in this chapter.
- The user must have a valid login (username and password) on both hosts. The user should have the same user-id (uid) on both systems.
- The remote execution host must be authorized to connect to the display on the local host. This authorization is implemented by:
  - The **/etc/X*.hosts** file.
  - The style manager Hosts dialog box.

  For more information about display access, see "Limiting Access to a Local Display" later in this chapter.
- If the application running on the remote execution host must access data files from the local host, then the application host must be configured for remote data access to the local host. (See "Configuring for Remote Data Access" in this chapter.)
- The local host must have NFS access to the **/** and **/tmp** directories of the remote host. These file systems must be mounted under **/nfs/***remote_host* and **/nfs/***remote_host***/tmp**.

Access Rights on the Remote Execution Host

When HP VUE runs a process on a remote execution host, the user's access rights on the remote machine are checked, and the user's current environment is transmitted to the remote process.

To check access rights on the remote machine:

- A file is created on the remote system via NFS.
- The user id of the file owner is obtained. The remote process runs as this user id.
- The user id is used as an index into `/etc/passwd` to obtain initial settings for the `HOME` environmental variable for the remote process

Environment for Processes Executing Remotely

When the process is run, the environment of the parent process is modified before it is inherited by the child. For local processes (processes running on the same machine as the parent HP VUE application):

1. The contents of the file `/usr/softbench/config/softenv` are read. Each variable in this file is put into the child environment. The variables in this file overwrite existing variables in the child environment.
2. The environment created by Step 1 is modified by the contents of `$HOME/.softenv`. A variable in `.softenv` overwrites any existing variables.

In order to set environment variables for remote process execution:

- The environment is modified in exactly the same way on the local host as if it were a local process, with the variables specified in `$HOME/.softenv` or `/usr/softbench/config/softenv` overriding current variables.
- The environment is transferred to the remote execution host.
- The environment is further modified according to the `$HOME/.softenv` or `/usr/vue/config/softenv` files on the remote host.
- The `DISPLAY` environment variable is handled specially. If it is set to `local:`*n*, `unix:`*n*, or `:`*n* on the local system, it is translated to the appropriate value of *hostname*`:`*n* before being placed in the remote environment.

Domain Naming Environment Considerations

In a domain naming environment, the host running HP VUE and the remote execution host must all reside in the same subdomain. These remote execution hosts within the local subdomain should be specified using a simple, unqualified name. For example, if VUE is running on "pronto.dom.hp.com" and "indigo.dom.hp.com" is to be used as a remote execution host, it should be specified using the simple name "indigo".

HP VUE is capable of accessing remote data hosts either within the local subdomain or in a separate domain. If the data host is within the local subdomain, it should be specified using the simple, unqualified name (for example, "indigo"). If the data host is in a separate domain, it must be specified using the fully qualified name (for example, "indigo.dom2.hp.com"). This fully-qualified name must be used for both the NFS mount and when specifying a file.

Refer to the networking documentation for your operating system for more information about domain naming.

Controlling Access and Security

This section covers several aspects of system security:

- Network security used by the Broadcast Message Server (BMS).
- X Windows security for X servers and displays.
- NFS security for NFS-mounted files.
- Providing logins on remote hosts.
- Providing network transparency—access without formal login (`.rhosts`).

Network Security

When HP VUE runs on a machine connected to a network, it should be configured to restrict access to only the work-group that requires it.

The following two files provide **mserve** and **spc** security.

Table 4-1. BMS Security Files.

File	Description
`/etc/services`	Lists all network services available through fixed port numbers. The **spc** service must be listed.
`/usr/adm/inetd.sec`	Lists systems for which BMS and Sub-process control services are provided.

4

The **mserve** (Message Server) and **spc** (Sub-Process Control) services provided by HP VUE are automatically restricted in `/usr/adm/inetd.sec` by the customization script to:

- The host name of the system, if it is running stand-alone. (This applies to HP-UX 7.* systems only.).
- The nodes in the cluster, if it is a cluster system. On a diskless cluster, `/usr/adm` is a context-dependent directory. Thus, `/usr/adm/inetd.sec` is not shared automatically within the cluster.

For example, this is the contents of `inetd.sec` on a cluster of three nodes after installation:

```
mserve  allow  hostA hostB hostC
spc     allow  hostA hostB hostC
```

To provide access from other machines, edit `/usr/adm/inetd.sec` manually by appending them to the end of the line. To add a complete network, include the network or subnet component of the address. For example:

```
mserve  allow  hostA hostB hostC 192.6.36.*
spc     allow  hostA hostB hostC 192.6.36.*
```

You should keep the access list as small as possible, since these services allow access to the machine by anyone connecting to the port.

If a machine attempts to connect to the message server or spc without access permissions, an error message occurs. The error message may indicate a broken connection and suggest that you check `inetd.sec`. Since there are other reasons why a connection may break, the system cannot always identify a security problem. If the machine on which HP VUE is started does not have permission to connect to the message server, the session manager displays an error dialog box and exits.

The host on which the environment is first started must always be listed in `inetd.sec`. If `inetd.sec` is shared via an NFS mount (this is not recommended), it would have to include all hosts which need to talk to any message server with access to the shared file. This is not handled automatically by the installation procedure.

If you change the host name or internet address of the machine, make sure that `inetd.sec` contains the new host name.

See the `inetd.sec(4)` man page for additional information.

Limiting Access to a Local Display

There are two basic schemes for limiting remote access to a local display.

- Authorization by host name. This is the default authorization system used by HP VUE. It involves maintaining a list of all the remote hosts that have permission to use the local server to display clients. Authorization by host name is implemented by:
  - The `/etc/X*.hosts` file, where * is the display (for example, `X0.hosts` for display 0).
  - The settings saved by the session manager and changed by the Hosts dialog box in the style manager.
- Authorization by user. This feature is provided by the R4 X server.

The 'X*.hosts' File

When the server starts for the first time (there have been no previously stored sessions), it reads `X*.hosts` to obtain its list hosts. `X*.hosts` has the syntax:

host1
host2
host3

In general, `X0.hosts` should contain only those remote systems where you regularly run clients with windows redirected to the local display. If you use the R4 security mechanism, it is recommended that `X0.hosts` be empty.

4 Once you are using the session manager to begin and end sessions, the `X0.hosts` file is no longer read.

Saving and Editing Hosts with the Style Manager

The first time the user displays the style manager's Hosts dialog box, its list of hosts, obtained from the server, is the same as the contents of `X*.hosts`. Any modifications made there are saved at the end of the session and restored at the beginning of the next session.

Refer to chapter 6 for more information on server settings saved by the session manager.

NFS Security

NFS security is provided by the file `/etc/exports`. It contains a list of the remote systems that are permitted to NFS-mount local disk volumes. For example, if `/etc/exports` on system `hphere` contains the entry:

```
/doc    hpthere
```

then system `hpthere` can mount volume `/doc` located on system `hphere`.

Providing Access Without Login

The **$HOME/.rhosts** file provides access by remote systems to the local system using the local login account. Thus, the remote system does not need to provide a formal log in and password.

The **.rhosts** file has the syntax:

host1 [*user*]
host2 [*user*]
host3 [*user*]

If the user name is not included on the same line as the host name, the **.rhosts** file assumes that the remote user name is the same as the local user name.

HP VUE does not directly use the **.rhosts** mechanism.

Changing Your Hostname

This section describes how to reconfigure the system if you change your host name after HP VUE has been installed. You might do this, for example, to change the host name from **unknown** (the default) to some unique name so that you can connect the system to a network.

In addition to the normal operating system tasks, there are several files you may need to modify:

- You may need to modify the **/usr/adm/inetd.sec** security file.
  - If this file does not exist, or if it contains the following lines:

    ```
    mserve  allow
    spc     allow
    ```

    then all hosts are authorized to connect to your local host. In this case changing your host name will not cause any problems.

4
  - If your system uses **/usr/adm/inetd.sec** to enforce security by listing specific hosts that are allowed, then the entry for your local host must be changed to your new host name. For example, if you change the name of your host from **oldname** to **newname**, then you must alter lines of the form:

    ```
    mserve  allow   oldname host2  ...
    spc     allow   oldname host2  ...
    ```

    by replacing **oldname** with **newname**.

- In cluster systems, you must modify all the context-dependent files in the **/usr/lib/X11/vue** directory. This includes the **/usr/lib/X11/vue/Vuelogin/Xservers** files.

Setting Up Logins on a Remote Host

For a local system to access a remote host:

- The address and hostname of the remote host must be listed in the local system's **/etc/hosts** file.
- The user must have a valid login (username and password) and home directory on the remote host.

To set up a login on a remote host:

- Ensure that the remote host has a valid internet address and hostname in the local**/etc/hosts** file.
- Provide a username, password (if necessary), and a home directory on the remote system.

If your system uses the Name Server, the **/etc/hosts** entry may not be necessary. Refer to the networking documentation for your operating system for more information.

Using HP VUE With Yellow Pages

If you are using Yellow Pages for a hostname server, the Yellow Pages server's database must contain the name of the hosts that are running HP VUE.

5

The Login Manager

The **login manager** is the term used to describe the portion of HP VUE responsible for:

- Reading initial configuration files.
- Starting the X server.
- Displaying the login screen and validating the login and password supplied by the user.
- Invoking the HP VUE session manager.

This chapter covers:

- The files used by the login manager.
- How the login manager obtains environment variables.
- Customizing the appearance and behavior of the login screen.
- How the login manager provides for multiple-display configurations.
- How to configure HP VUE for various special environments:
  - Operation without the X server (non-window operation).
  - Foreign servers (X terminals).
  - HP-UX cluster systems.

Login Manager Files

This section covers the files used by the login manager.

The Login Manager Clients

The functionality of the login manager is provided by three HP VUE clients, located in directory `/usr/lib/X11/vue/etc`.

Table 5-1. The Login Manager Clients.

Client	Description
`vuelogin`	Performs configuration tasks and spawns another `vuelogin` process for each display in the system. Thus, there will be n+1 `vuelogin` clients running, where n is the number of displays managed by the system.
`vuegreet`	Provides the functionality of the login screen.
`vuehello`	Provides a transition effect between successful login and the beginning of the HP VUE session. By default, this is a welcome message and copyright notice.

5

Other Configuration Files

This section describes the configuration files used by the login manager.

Configuration Files That Boot HP VUE

The configuration files that boot HP VUE depend on the operating system. Refer to the *HP Visual User Environment Configuration Guide* for your operating system for additional information.

Files in the Vuelogin Directory

The following login manager files provide system configuration information. They are located in `/usr/lib/X11/vue/Vuelogin`.

Table 5-2. Login Manager Files in /usr/lib/X11/vue/Vuelogin.

File Name	Description
`Xconfig`	Contains the resources for the behavior of **`vuelogin`**. It can also define new locations for the other **`vuelogin`** configuration files.
`Xerrors`	Contains error messages from **`vuelogin`** and other programs run during session initiation.
`Xpid`	Contains the process id of the parent **`vuelogin`** process.
`Xservers`	Contains a list of servers to be run by **`vuelogin`**.
`Xresources`	Contains resources for the appearance of the login screen.
`Xstartup`	Program (usually a shell script) that is run as root after the user login and password is validated.
`Xsession`	Shell script that sets up the user environment variables, runs **`vuehello`**, and invokes the session manager.
`Xreset`	Shell script that is run as root upon termination of an HP VUE session. It can be used to return the system to its pre-**`Xstartup`** configuration.

On an HP-UX diskless cluster, the `Vuelogin` directory is context-dependent.

On Domain/OS systems, the `Vuelogin` directory is a link to `'node_data` (or to `'node_data.`*node_id* for diskless clusters.)

User-Specific Files

The following files provide default configurations that are user-specific.

Table 5-3. Login Manager Files in $HOME.

File Name	Description
`.Xauthority`	Contains authorization information needed by clients that require an authorization mechanism to connect to the server.
`.vueprofile`	Contains environment variables.

How the Login Manager Works

5 The login manager executable (`vuelogin`) is started during the system boot sequence.

1. `vuelogin` reads the configuration file `Xconfig` in `/usr/lib/X11/vue/Vuelogin`. `Xconfig` contains resources for various login manager actions.
2. `vuelogin` reads the following files in `/usr/lib/X11/vue/Vuelogin`:
   - `Xservers`, or the file identified by the resource `Vuelogin.servers` in the `Xconfig` file.
   - `Xresources`, or the file identified by the `Vuelogin*resources` resource in the `Xconfig` file.
3. The parent `vuelogin` starts the X server for each local display.
4. The parent `vuelogin` starts another `vuelogin` process for each managed display.

The following steps happen for each **vuelogin** started by the parent **vuelogin**:

1. **vuelogin** invokes **vuegreet**, which displays the login screen and handles the user's interaction with the login screen.
2. Once the user login and password have been validated, **vuelogin** runs **Xstartup**.
3. **vuelogin** sets certain environment variables to default values. (See "Setting Environment Variables" later in this chapter.)
4. **vuelogin** then runs **Xsession**, which performs several operations:
   - It runs **vuehello**, which displays the message of the day or some other transition effect.
   - It reads **$HOME/.vueprofile**, which may contain additional environment variables or variables that override those set automatically by **vuelogin**.
5. **vuelogin** invokes the session manager, **vuesession**.

When the user terminates a session, the login manager runs **Xreset**.

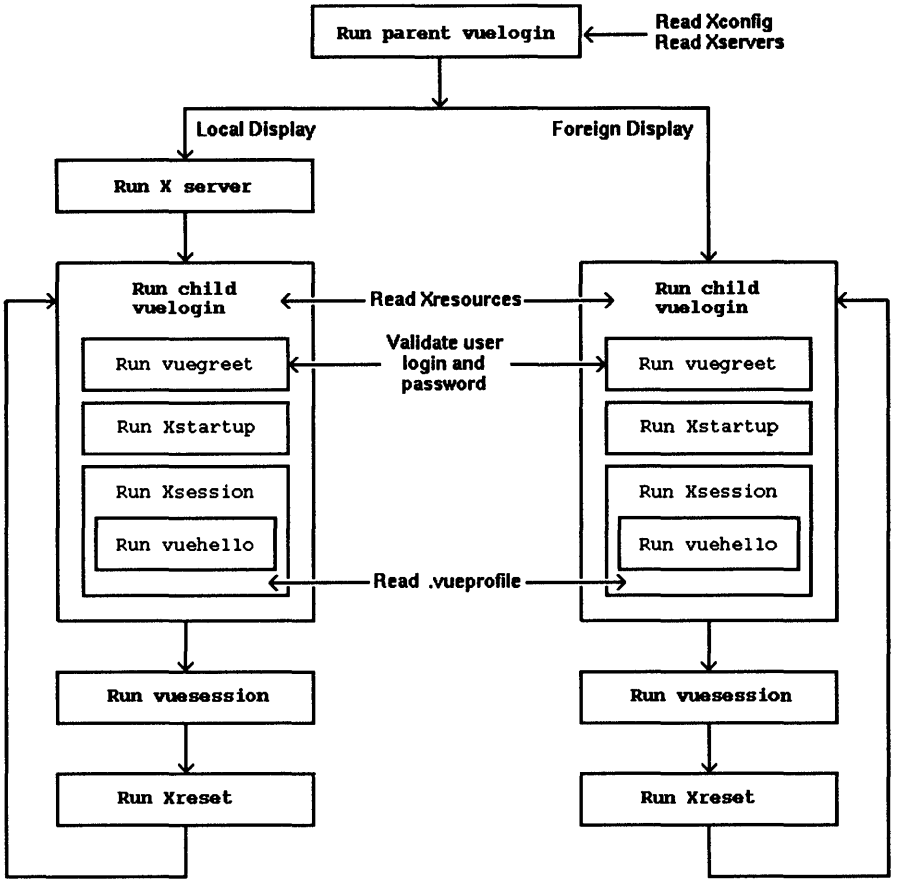


Figure 5-1. Login Manager Operations.

5

Starting HP VUE

The procedure for starting (booting) HP VUE depends on the operating system on which it will be running. Booting HP VUE is covered in the *HP Visual User Environment Configuration Guide*.

Customizing Startup Behavior

It may be necessary for the system to perform certain tasks for the user—for example mounting the user's file system— before invoking the session manager. The `/usr/lib/X11/vue/Vuelogin/Xstartup` file provides this capability. Usually, `Xstartup` will be a shell script. It is run as root after the user has been authenticated, before the session manager is started.

The following environment variables are set to the specified values for the duration of `Xstartup`:

Variable	**Value for `Xstartup`**
`DISPLAY`	Display name.
`USER`	User name.
`HOME`	Home directory of the user.
`PATH`	Value of the `systemPath` resource in `Xconfig`.
`SHELL`	Value of the `systemShell` resource in `Xconfig`.
`XAUTHORITY`	May be set to an authority file.
`TZ`	Set to the current time zone.

You can use `Xreset` to undo tasks done by `Xstartup`.

Setting Environment Variables

This section covers:

- How HP VUE obtains environment variables.
- How to set additional variables for the HP VUE environment.
- How to provide for executing additional shell commands for a non-HP VUE environment.

How HP VUE Obtains Environment Variables

Since HP VUE runs the X server before the user has logged in, the functionality provided in X11 by `bin/login` and `.profile` or `.login` must be provided instead by `vuelogin`.

Caution

When a user logs in to HP VUE, the `.profile` (or `.login`) file is not read.

5

The login manager obtains environment variables as follows:

- Certain environment variables are built into **vuelogin**. These are set separately for each display after the user login and password have been validated. For a list of these variables, see the *HP Visual User Environment Configuration Guide*.
- System-wide variables are set in **/usr/lib/X11/vue/Vuelogin/Xconfig** or **/usr/lib/X11/vue/Vuelogin/Xsession**. See "Setting System-Wide Environment Variables" in this chapter for more information.
- Additional user environment variables can be placed in **$HOME/.vueprofile**. This file can also be used to override system-wide variable settings. This file should contain only variable settings; it cannot contain shell commands that require terminal I/O or that run in the foreground. See "Setting Personal Environment Variables" later in this chapter.

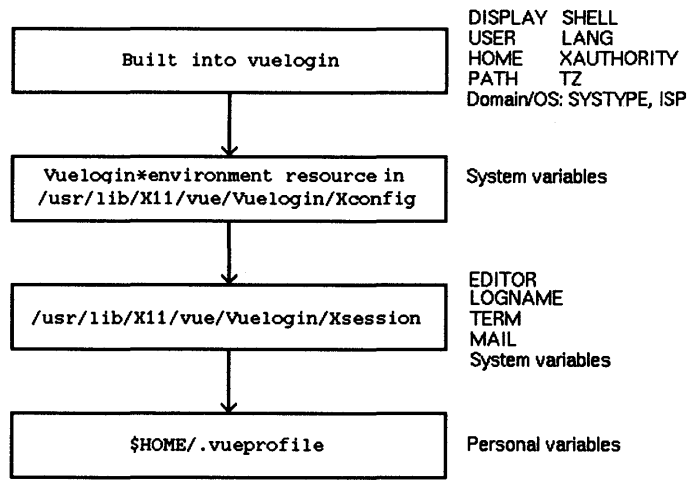


Figure 5-2. How HP VUE Obtains Environment Variables.

Setting System-Wide Environment Variables

System-wide environment variables can be specified in the following locations:

- **/usr/lib/X11/vue/Vuelogin/Xconfig**. Variables set here will be available both to the HP VUE session and to the X server.
- **/usr/lib/X11/vue/Vuelogin/Xsession**. Variables set here are available only to the HP VUE session.

Setting Variables in 'Xconfig'

Variables set in **/usr/X11/lib/vue/Vuelogin/Xconfig** are available both to the HP VUE session and to the X server. If the X server does not use the variable, it will be applied only to the HP VUE session.

There is no shell processing within **Xconfig**. Variables except **TZ** and **LANG** are set using the **environment** resource. Variables can be set for all displays or on a per-display basis.

5

The syntax for specifying environment variables in **Xconfig** is:

Vuelogin*[*host_display.*]**environment:** *variable=value*

To set more than one variable, separate each assignment with a space or tab. The values assigned to the variables must be constants, since the resource does not use a shell to interpret the value.

For example, the following lines set variables. EDITOR will be available for all displays. SB_DISPLAY_ADDR is available only to **hpcvhere:1**.

```
Vuelogin*environment: EDITOR=vi                          All displays.
Vuelogin*hpcvhere_1.environment: EDITOR=vi \             Display hpcvhere:1.
                                 SB_DISPLAY_ADDR=0xB00000
```

The **TZ** and **LANG** variables have dedicated resources. For example, the following line in **Xconfig** sets the time zone for all displays:

```
Vuelogin*timeZone:  PST8PDT
```

See "Setting the Time Zone" later in this chapter for more information about **TZ**.

Setting Variables in 'Xsession'

Variables set in `/usr/X11/lib/vue/Vuelogin/Xsession` are applied to the HP VUE session, but *not* to the Xserver. `Xsession` is a shell script, so shell processing is available. For example, the following line is acceptable syntax for setting MAIL.

```
MAIL=/usr/mail/$USER
```

To apply variables on a per-display basis, use a separate "Xsession" file for each display, and specify the individual files in `Xconfig`. For example, the following line in `Xconfig` specifies the session file for display `hpcvhere:1`:

```
Vuelogin*hpcvhere_1.session: /usr/lib/X11/vue/Vuelogin/Xsession.hpcvhere
```

Setting Personal Environment Variables

User-specific environment variables are set in `$HOME/.vueprofile`.

5

Creating a '.vueprofile' File

A `$HOME/.vueprofile` file is created, if it does not already exist, the first time you log in to HP VUE. However, should you need to create the file yourself, follow this procedure:

1. Copy the template `/usr/lib/X11/vue/sys.vueprofile` to `$HOME/.vueprofile`. Give the new file write permission.
2. If you want the file interpreted according to a shell other than the default shell, specify the shell for the file by placing `#!/bin/sh`, `#!/bin/csh`, or `#!/bin/ksh` at the top of the file.
3. Add all the environment variables you want for your HP VUE environment.

Note

The `.vueprofile` file should contain only lines that set environment variables. It must not contain commands that do terminal I/O, "tset", "stty", etc.

Setting Environment Variables in '.vueprofile'

The `.vueprofile` file should contain only variable settings; it should contain neither shell commands that require terminal I/O nor commands that run in the foreground.

Caution

When a user logs in to HP VUE, the `.profile` or `.login` file is not automatically read.

Set *only* environment variables in `.vueprofile`. Commands like those for terminal input or output will corrupt your session environment.

Using an Existing Shell Environment File with '.vueprofile'

If you have an existing shell environment file (`.profile` or `.login`), you can continue to use that file by doing the following:

5

1. Add lines to the end of `.vueprofile` to source in the shell environment file.

   For ksh:

   ```
   VUE=true; export VUE
   $HOME/.profile
   ```

   For csh:

   ```
   setenv VUE true
   source $HOME/.vueprofile
   ```

2. Edit the shell environment file to provide two sections. One section contains commands that do not apply to HP VUE (for example, commands that require terminal I/O, or variables for which you want to replicate the HP VUE default values). The other section contains variables that apply, whether or not HP VUE is run:

   For example, for ksh:

   ```
   if [ ! "$VUE" ]; then
        stty options
        tset options
        DISPLAY=value
        MAIL=value
        export DISPLAY MAIL
        Additional shell commands that do not apply to HP VUE
        Additional variable assignments that do not apply to HP VUE
   fi
   PATH=value
   Assignments for common variables
   ```

   For csh:

   ```
   if ( ! "$VUE" ) then
        stty options
        tset options
        setenv DISPLAY value
        set mail = value
        Additional shell commands that do not apply to HP VUE
        Additional variable assignments that do not apply to HP VUE
   endif
   setenv PATH value
   Assignments for common variables
   ```

Setting the Time Zone (TZ) Environment Variable

HP VUE obtains the value of the TZ environment variable from the following sources:

- An operating system file may define a default value.
- The **timeZone** resource in **/usr/lib/X11/vue/Vuelogin/Xconfig**. It has the syntax:

  Vuelogin*timeZone: *value*

  You can use the resource to override the value set by the operating system.
- **$HOME/.vueprofile**. Values set here override system-wide values.

The Display Environment Variable

5 The **DISPLAY** environment variable sets the host, display number, and screen number to which a system sends bitmapped output for clients.

The default value of **DISPLAY** is set automatically by **vuelogin** to *hostname*:0, which is display 0, screen 0 of the local display. If you must change it, edit the first field of the appropriate entry in the **Xservers** file. (The HP-UX **shmlink** specification for the **DISPLAY** variable is not supported by HP VUE.)

Most clients have a **-display** option for specifying a different host, display number, and screen number on which the client should display its output.

Setting Additional Shell Commands

If **vuelogin** is not executed (for example, you run a pure X11 environment or you log into an HP VUE system from a remote terminal), you may need to execute shell commands. Shell commands should be placed in **.profile** or **.login**.

Customizing the Login Screen Appearance

To change the appearance of the login screen, you must edit resources in `/usr/lib/X11/vue/Vuelogin/Xresources` or in the file specified by the resource `Vuelogin.`*Display*`.resources` in the `Xconfig` file.

Changing the Login Screen Logo

The syntax for logo resources is:

`Vuelogin*logo*`*resource*: *value*

For example, the following line specifies a different bitmap for the logo:

```
Vuelogin*logo*bitmapFile: /usr/lib/X11/bitmaps/Vuelogin/MYlogo.bm
```

For additional logo resources, refer to the `vuelogin(1x)` man page.

Other Appearance Resources

The `Xresources` file contains additional information about resources for changing the appearance of the login screen.

Login Manager Errors

If the login manger seems to not be working properly, you should check to see whether an error has been generated. Login manager errors are stored in the file `/usr/lib/X11/vue/Vuelogin/Xerrors`. This includes errors produced by:

- `vuelogin`.
- Any standard error output by `Xstartup`, `Xsession`, or `Xreset`.

Message of the Day

The `vuehello` client provides the ability to display a message of the day. It displays a transition window after the user login and password are validated, before the session manager is run.

By default, `vuehello` displays `Starting the HP Visual User Environment` and the copyright message (`/etc/copyright`) in the transition window. To add additional messages, use the `-file` option on the command line in `Xsession` that starts `vuehello`. Up to five files can be added; each uses a separate `-file` option. For example, the following line adds the general message of the day and a user-specified message.

```
/usr/lib/X11/vue/etc/vuehello -file /etc/motd   \
-file $HOME/mymessage
```

5

Running HP VUE on Non-Console and Multiple Displays

Note

This section applies to operating systems in which the X11 Window System supports multiple displays.

The following two `vuelogin` files must be edited if you want to run HP VUE on one or more non-console displays.

- `/usr/lib/X11/vue/Vuelogin/Xservers`. `Xservers` is responsible for starting the server on each display. There must be an entry in `Xservers` for each server to be started.
- `/usr/lib/X11/vue/Vuelogin/Xconfig`. `Xconfig` contains resources necessary for invoking No Windows Mode on a display. The resources specify characteristics of the ITE associated with each display. `Xconfig` can also be used to specify resources for each server started.

In HP-UX systems, you may also need to edit `/etc/inittab`. See the *HP VUE Configuration Guide for HP-UX Systems* for more information.

Editing 'Xservers' for Multiple Displays

The file **/usr/lib/X11/vue/Vuelogin/Xservers** starts the X server on each display running HP VUE. The default **Xservers** file assumes that HP VUE will run on one display—the system console. If HP VUE will be run on displays other than the system console, you must edit **Xservers** to start the server on each of those displays.

Xservers contains a line for each server. The syntax for specifying a server is:

DisplayName DisplayClass DisplayType Command

DisplayName **$DISPLAY**. The display name that is passed to X programs by the **-display** option. It is used in display-specific resources to specify a particular display. The default is *hostname*:0.

DisplayClass Used in setting display-specific resources for a particular class of displays.

DisplayType There are two display types:

local	The local display running the server.
foreign	A remote display. Remote displays usually are X terminals that run sessions from a file server.

Command The command line to start the server (for local servers only). For remote displays, the *Command* entry is ignored.

For example, the following entries in **Xservers** start the server **/usr/bin/X11/X** on three local displays running HP VUE on host **hpcvaaa**.

```
hpcvaaa:0 Local local /usr/bin/X11/X :0
hpcvaaa:1 Local local /usr/bin/X11/X :1
hpcvaaa:2 Local local /usr/bin/X11/X :2
```

The *hostname* can be a simple host name (for example, **hpcvaaa**) or a fully qualified domain name (for example, **hpcvaaa.cv.hp.com**).

Editing 'Xconfig' For Multiple Displays

In a system with multiple displays, the file `/usr/lib/X11/vue/Vuelogin/Xconfig` is used:

- To specify different session initiation resources for different displays.
- To specify a file other that `Xservers`
- To configure No Windows mode for HP-UX systems running HP VUE.

Using Different Resources for Different Servers

The `Xconfig` file can contain resources for customizing session initiation for different displays. The general syntax for these resources is:

$$\texttt{Vuelogin.}\left\{\begin{matrix}\textit{host\_display}\\ \textit{display\_class}\end{matrix}\right\}\textit{.resource: filename}$$

5 The files specified by *filename* are used by the login manager for the specified *host:display* or *display_class*.

The following table lists some of the resources that provide display-dependent session initiation. Refer to the `vuelogin(1x)` man page for additional information.

Table 5-4. Display-Dependent Resources for Session Initiation.

Resource	Description
`startup`	Specifies the file run as root after valid login and password.
`resources`	Specifies the name of the file to be loaded as the resource database for the login screen.
`xrdb`	The program used to load resources.

For example, the following lines specify that separate files are used for two servers.

```
Vuelogin.hpcvxpae_0.resources:              \
     /usr/lib/X11/vue/Vuelogin/Xservers
Vuelogin.hpcvxpae_1.resources:              \
     /usr/lib/X11/vue/Vuelogin/Xservers1
Vuelogin.hpcvxpae_0.startup:                \
     /usr/lib/X11/vue/Vuelogin/Xstartup
Vuelogin.hpcvxpae_1.startup:                \
     /usr/lib/X11/vue/Vuelogin/Xstartup1
```

Specifying a Different File for Starting Servers

You can use the resource `Vuelogin.servers` to specify a different file for starting the server(s). The resource is entered into the file `/usr/lib/X11/vue/Vuelogin/Xconfig`.

`Vuelogin.servers` can take as its value one of the following:

- A single server local to the host.
- The absolute path of a file containing a list of local servers.

The default value for `Vuelogin.servers` is `/usr/lib/X11/vue/vuelogin/Xservers`.

Configuring No Windows Mode for Multiple Displays

Note No Windows mode is not available on Domain/OS systems.

The login manager provides No Windows mode for workstations that are occasionally used for non-X applications that cannot run simultaneously with X. No Windows mode is invoked by selecting No Windows from the option menu in the login screen.

No Windows mode suspends the server and runs a standard login sequence (getty/login). When the user finishes and logs out, the server is restarted and the vuelogin screen is redisplayed.

In order for No Windows mode to run properly on a display:

- The `Xservers` file must specify that the display is type `local`.
- The `Xconfig` file must specify the display's associated ITE device file and speed.

Two resources in `Xconfig` are passed to the `/etc/getty` process to indicate the desired device and speed for the login sequence.

Table 5-5. getty Resources.

Resource	Description
`gettyLine`	The device file for the login sequence. The default is `console`.
`gettySpeed`	An entry from `/etc/gettydefs`. The default is `console`.

5

Each display must have a different value for `gettyLine`.

For example, the following lines configure No Windows mode for non-console displays named `hpcvaaa:1` (associated with device `/dev/ttyi1`) and `hpcvaaa:2` (associated with device `/dev/ttyi2`). The line speed is 9600.

```
Vuelogin*hpcvaaa_1.gettyLine: ttyi1
Vuelogin*hpcvaaa_1.gettySpeed: 9600
Vuelogin*hpcvaaa_2.gettyLine: ttyi2
Vuelogin*hpcvaaa_2.gettySpeed: 9600
```

Note

If there are more displays on the system than the number of available ITE's, not all the displays will be able to operate in No Windows mode.

For operating system releases prior to 8.07 only: Every display on the system must contain a unique `gettyLine` specification in the `Xconfig` file; displays without ITE's must use a "dummy" `gettyLine` value.

X-Terminals

Note

This section applies only to operating systems that support X-terminals.

An X-terminal system consists of a display, keyboard, and mouse that runs only the X server; clients, including HP VUE, are run on one or more "host" systems on the networks. Output from the clients is directed to the X-terminal display.

Terminals Supporting XDMCP

XDMCP (X Display Manager Command Protocol) provides a mechanism by which X-terminals can request login services from a network host. It ensures that the X-terminal is communicating with a valid login manager, and provides the protocol for exchanging authentication information between the X-terminal and the host login manager.

If the terminal supports XDMCP, use the terminal's mechanism to request service from the host system running HP VUE.

Non-XDMCP Terminals

When HP VUE runs as a file server for a collection of non-XDMCP X terminals, edit the `Xservers` file to include an entry for each terminal. The *displayType* of each terminal must be `foreign`. For example, the following lines in `Xservers` directs `vuelogin` to manage sessions on three terminals.

```
ext1:0 HP700X foreign
ext2:0 NCD-19 foreign
ext3:0 NCR-TOWERVIEW3000 foreign
```

vuelogin will respond to two signals for the terminals:

- When it receives a SIGHUP, it rereads **Xconfig** and the **Xservers** file (or the file specified by the **Vuelogin.servers** resource). If it finds a new entry, **vuelogin** starts managing that display. If an entry has been removed, the process associated with that entry is immediately terminated.
- When it receives a SIGTERM, **vuelogin** terminates all sessions in progress.

Cluster Systems

Configuring diskless clusters for HP-UX and Domain/OS systems is covered in the *HP Visual User Environment Configuration Guide* for those systems.

5

Stopping HP VUE

Note

This section applies to HP-UX systems only. For other operating systems, refer to the *HP Visual User Environment Configuration Guide* for your operating system.

To stop HP VUE:

1. Log out.
2. Use the Options menu on the login screen to enter No Windows mode.
3. Log in as root.
4. Switch to a run level that does not run HP VUE by executing:

   ```
   /etc/telinit n; exit
   ```

   where *n* is a run level for which **/etc/inittab** specifies a terminal or console.

You can now log in again.

Shutting Down the Workstation

Note

This section applies to HP-UX systems only. For other operating systems, refer to the *HP Visual User Environment Configuration Guide* for your operating system.

To shut down a workstation running HP VUE:

1. Stop HP VUE. See "Stopping HP VUE," above.
2. Log in as root.
3. Change to the root directory.
4. Execute the appropriate shutdown command for your system. Refer to the owner's manual for your system for additional information.
5. Reboot your workstation.

6

The Session Manager

This chapter covers:

- The concept of an HP VUE session and the session manager.
- The files read and written to during a session.
- Customizing various aspects of the session manager

Introduction to HP VUE Sessions

An HP VUE session starts when the user successfully logs in and ends when the user exits the session and returns to the login screen. The HP VUE Session Manager is responsible for a number of housekeeping tasks associated with managing an HP VUE session:

- It starts the Broadcast Message Server (BMS), window manager and other clients.
- It maintains a database of files that saves session attributes so that the user can return to known states upon login.
- It selects the appropriate database files to use for starting a session, based on whether the user has chosen to return to the current (previous) or home session.

Names and Locations of Session Manager Files

This section covers the files that constitute the session manager and the session manager database.

Session Manager Executable

The session manager is implemented by the client **vuesession**, which is responsible for loading the appropriate resources, restoring server settings, and starting the clients. It is located in directory **/usr/lib/X11/vue/etc**.

Current and Home Session Databases

The session manager database has two major parts:

- The **current session database** reflects the status of the system at the time the user logs out. This allows the session manager to return to the same state upon login if the user has chosen to do so.
- The **home session database** is a snapshot of the system explicitly taken by the user in the style manager. This database allows the user to store away a known state that he can return to later.

6

Each display on which the session manager runs has two directories containing the session management current and home databases:

Current database: **$HOME/.vue/***host*:*display*/**current**.

Home database: **$HOME/.vue/***host*:*display*/**home**.

where *host* is the host on which the session is being displayed.

For example, if a session is running on display 0 of host **hpcvhere**, the directories **$HOME/.vue/hpcvhere:0/current** and **$HOME/.vue/hpcvhere:0/home** contain the session manager files.

Session Database Files

The session manager creates and uses the following files in the session manager database directories.

Table 6-1. Files Comprising the Current and Home Database.

File	Purpose
`vue.session`	Stores the names of active clients, their window geometries, their workspace presence status (which workspaces they are in, their state (normalized or minimized), and startup string.
`vue.resources`	Stores the resources for the active clients (including the window manager) in the session.
`vue.settings`	Stores the server settings and session manager settings.

The following files are used when no `current` or `home` directories exist. They are located in directory `/usr/lib/X11/vue/Vuesession`:

- `sys.session` stores the user default startup script.
- `sys.resources` stores the user default resources.

Session Startup

When **vuesession** executes, it performs the following operations:

1. It starts the Broadcast Message Server, which is responsible for coordinating communication between the various HP VUE components.
2. It searches for the existence of a session manager database directory:
   a. First, it searches for the **$HOME/.vue/***host*:*display*/**current** directory. This directory exists if the user chose, at the end of the previous session, to return to that state at next login.
   b. If the **$HOME/.vue/***host*:*display*/**current** doesn't exist, it looks for the **$HOME/.vue/***host*:*display*/**home** directory. This directory exists if the user has saved a home session in the style manager Startup dialog box.
   c. If neither of these directories is found, the session manager will use **sys.session** and **sys.resources** in directory **/usr/lib/X11/vue/Vuesession**.
3. When it finds the appropriate directory, it uses the **vue.resources** (or **sys.resources**) file in that directory to pass the resource values to the resource database manager (**xrdb**).
4. It reads the accompanying **vue.settings** file.
5. It then starts the window manager (**vuewm**).
6. It reads the **vue.session** file and starts the clients listed in that file.
7. It runs the optional **sessionetc** script.

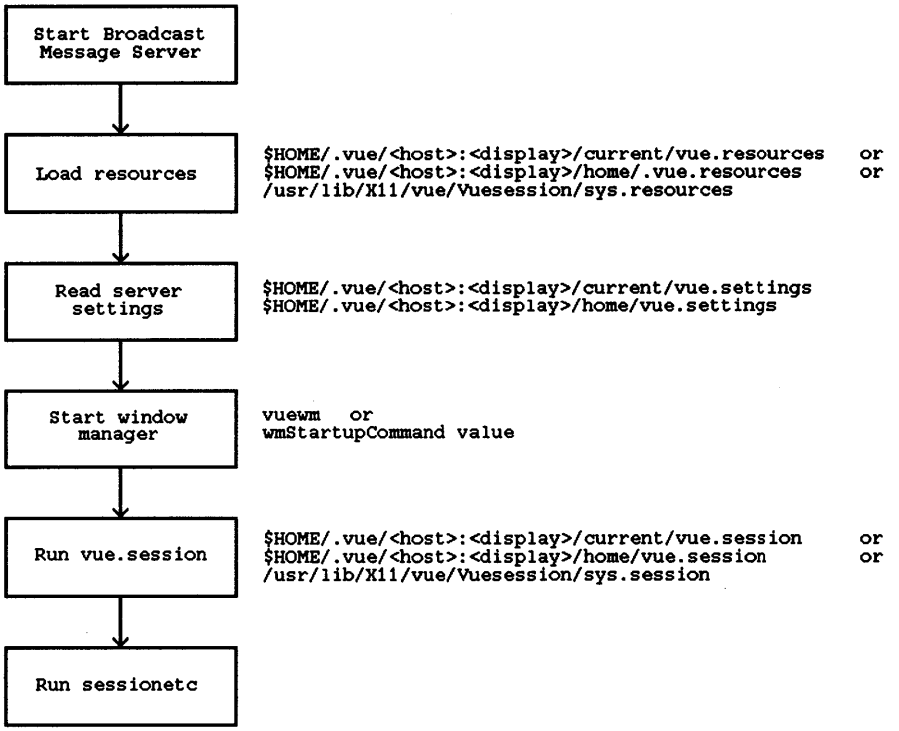


Figure 6-1. How the Session Manager Starts a Session.

6

Changing Resource Settings During a Session

Since the session manager maintains its own resource files for current and home sessions, there is no single file that can be used to add, delete, and modify resources. In general, it is unwise to write to the `vue.resource` files, since these are overwritten by user actions (logout or storing a home session).

The best way to modify resources uses `xrdb` to merge new resources into the resource database. Managing resources is covered in chapter 8.

Starting a Session With a Different Window Manager

The window manager (**vuewm**) is automatically started by **vuesession**. However, there may be configurations that require you to start a different window manager.

The **wmStartupcommand** resource allows you to specify a different window manager. It takes as its value the executable command for starting the alternate window manager. For example, if the resource file contains the line:

```
vuesession*wmStartupCommand:     /usr/bin/X11/mwm
```

then **vuesession** starts the **mwm** window manager instead of **vuewm**.

Customizing Session Startup

In general, the session manager will take care of starting clients at the beginning of each session.

If desired, however, there are several ways to customize session startup:

6

- To customize the first session, edit the files **sys.session** and **sys.resources** in **/usr/lib/X11/vue/Vuesession/sys.session**.
- If you need to execute X commands for processes or settings that are not saved by the session manager, create the shell script file **$HOME/.vue/***host*:*display*/**sessionetc**. When this file exists, the session manager runs it after it starts the clients listed in **vue.session**.

For example, suppose you want to use **xsetroot** to customize the root (workspace) cursor at startup. You can do this by adding the following line to **sessionetc**:

```
xsetroot -cursor /usr/include/X11/bitmaps/pointer \
  /usr/include/X11/bitmaps/pointer.mask -fg black -bg red
```

Processes that run indefinitely must be run in the background.

Customizing the Display Lock

The default workspace manager includes the lock control for locking the display. When the display is locked, it ignores all keyboard and mouse actions until the user enters the password to unlock the display.

The session manager provides three resources for customizing the display lock.

Table 6-2. Display Lock Resources.

Resource	Description
`coverScreen`	When "True,", covers the locked display.
`alarmTime`	Sets the number of seconds the lock message dialog box is displayed.
`keys`	Specifies the users who can unlock the display with their passwords. The default is the login user and root.

For example, the following lines specify that a locked screen is covered and that the login message is displayed for 30 seconds whenever the system detects keyboard or mouse activity. Users kreta, charlie, dex, and anna can unlock the screen with their passwords. (Root can also unlock the display).


```
vuesession*coverScreen:     True
vuesession*alarmTime:       30
vuesession*keys:            kreta,charlie,dex,anna
```

Error Logging

Errors that occur during a session are written to the file **$HOME/.vue/errorlog**. This includes errors generated by **vuesession**, **vuewm**, and any other clients.

When the Messaging Service Fails to Start

If the session manager cannot start the BMS, it displays an error dialog box and exits back to the login screen. You must then use "No Windows" mode or a fail-safe session to correct the problem. Refer to "What to Do if the Session Manager Won't Start" at the end of this chapter for more information.

Session Termination

The action taken by the session manager when a user terminates a session depends on:

6

- Whether or not the user has requested that they want the current session restored. This determines how the session manager manages its database.
- The value of the **queryServerSettings** resource, which determines which settings are saved.

Managing the Database at Session Termination

If the current session will be restored, the session manager stores the current resource database, server settings, and client information in files in the **$HOME/.vue/***host*:*display*/**current** directory.

If the user has specified that the home session be used at next login, the session manager removes the **$HOME/.vue/***host*:*display*/**current** directory. When the user logs in again, the session manager will use the files in the **$HOME/.vue/***host*:*display*/**home** directory.

Specifying Which Settings are Saved

By default, the session manager saves only those server settings that are set in the style manager:

- Bell volume, pitch, and duration.
- Keyboard click volume and autorepeat.
- Mouse acceleration and threshold.
- Screen saver time.
- Host information.
- Session startup information.

However, the session manager resource **querryServerSettings** allows you to specify that additional server settings be saved.

To save additional settings, set the resource to True by adding the following line to the resource file:

```
vuesession*queryServerSettings:    True
```

Canceling a Session

6

It may be necessary for you to cancel a session without using the normal HP VUE logout procedure. For example, this could happen if you made a mistake while trying to customize the window manager and workspace manager that deactivated both the logout button in the workspace manager and the **Log out ...** button in the workspace menu.

To cancel a session:

- HP-UX systems: Hold down Shift and CTRL and press Reset (Shift CTRL Pause for HP C1429A Enhanced Vectra keyboards). The system will return to the login screen without saving your current session. Use the **ps -fu** *login* to make sure all **hpterm** processes are stopped. If any are still running, kill them using **kill -9**.
- Other systems: Use the **ps** command with the appropriate options to obtain the X server's process id. The process is associated with the line **/usr/X11/bin/***server_name* **:***display*. Then, use the apropriate kill command to halt the server.

Running Non-Standard Clients

Certain non-standard clients may not be saved by the session manager. These clients are not restored when the user logs out and back in.

The session manager file `$HOME/.vue/`*host*`:`*display*`/sessionetc` provides a way to execute these clients at the beginning of each session. Clients contained in this file will be started at the beginning of every session—both current and home sessions.

Note

Do not use `sessionetc` to start clients that are restored by the session manager. Doing so causes the session manager to start a growing number of processes, since a new process is started each time the user logs in. You may not immediately detect this because the windows are superimposed.

Multiple Screens

In a multiple screen configuration, the session manager is able to restore clients to the screens on which they were running.

6

What To Do if the Session Manager Won't Start

This section provides trouble-shooting procedures to use if the session manager fails to start because it cannot start the messaging service (the Broadcast Message Server or BMS). The symptom of this is that the system displays an error dialog box and exits back to the login screen.

If the BMS fails to start, log into HP VUE using a fail-safe session and check your system:

- Make sure that the line that starts the server in `/usr/lib/X11/vue/Vuelogin/Xservers` uses the actual host name. (The * in the first field of the command is also allowed.)

- If you have previously used Softbench, and have used your own `$HOME/.softinit` file, you may need to remove it or edit it to include the HP VUE tools. Refer to chapter 15 for more information.
- The BMS requires the following files and cannot start if any of them are missing or lacks read permission:
  - In the `/usr/softbench/bin` directory: `softmsgsrv softspcd`, and `softspcd.o`. Series 800 systems running HP-UX 7.* also require `softmsgsrv.o`. Domain/OS systems also require `softinit` and `softmerge`.
  - A valid `softinit` file: either `/usr/softbench/config/softinit` or `$HOME/.softinit`. Domain/OS systems also require `softboot`.

Additional Considerations for HP-UX 7.* Systems

These requirements apply only to HP-UX 7.* systems:

- If the workstation is configured as a solitary system (it is not networked to other systems), it must still be properly configured for networking in order for the BMS to start. This can be done by configuring your system to use the loopback address (127.0.0.1). Refer to chapter 1 of the *HP Visual User Environment Installation Guide* for procedures for configuring a solitary system.

- Verify that networking is properly configured:
  1. Execute the command:

     ```
     /etc/ping 'hostname'
     ```

     Use the literal string `'hostname'`—do not substitute your actual host name.
  2. Interrupt this command after approximately 10 seconds (use CTRL c, DEL, or Break).
  3. You can proceed if the output shows a 0% packet loss under the line:

     ```
     ----hostname PING Statistics----
     ```

     If this is not the case, see the configuration information in chapter 1 of the *HP Visual User Environment Installation Guide* and the troubleshooting sections in the *Installing and Administering LAN/9000* manual.

- The security file **/usr/adm/inetd.sec** must be properly configured to allow your system to connect to the BMS. Since the **/usr/adm** directory is a context-dependent file, you must check this file from the system experiencing the difficulties.

  Look for lines of the form:

  ```
  mserve    allow  host1  host2
  spc       allow  host1  host2
  ```

  These lines must specify the official name of your system (not an alias). It is best to use the short form of the host name (for example, **hpcvlx**) rather than the long form (**hpcvlx.cv.hp.com**).

  This file must be readable by all users.

- The file **/usr/softbench/config/softtypes/***language* must exist. If necessary, see "Localization Issues" in the *HP Visual User Environment Installation Guide* for more information.

7

Running Programs Using Command Lines

The terminal emulator (**hpterm** or **xterm**) gives you access to a command line and shell. You can use the command lines to start applications and to execute operating system commands.

Syntax for Starting Local Clients

The general syntax for the command line that starts a client is:

client [*-options*] [**&**]

An **&** at the end of the command line causes the client to start as a background process.

Command-Line Options

Each client has its own set of options; certain options are available for many clients. For example, most viewable clients have a **geometry** option for specifying the size and position of the client window. Command-line options override all default files. If no options are specified, the client is started using resource values from the resource database, the client's app-defaults, or from defaults built into the client.

Options have the syntax:

-option argument

For example, the following command line starts an **xterm** window with a black background and white foreground:

```
xterm  -bg  Black  -fg  White &
```

The next command line starts an **xterm** window in the workspaces named `project1` and `project2`.

```
xterm  -xrm '*workspaceList: project1 project2' &
```

Specifying the Display and Screen

The default display on which a client is displayed is obtained from the `DISPLAY` environment variable of the system on which the client starts. It sets the host, display number, and screen number to which the server sends output. This is typically display 0, screen 0.

Most clients have a **-display** option that lets you set the host, display number, and screen on which the client will display its output. The **-display** option has the syntax:

-display [*host:display.screen*]

host	The hostname of a valid system on the network.
display	The number of the display on the system on which you want the output to appear. A display can include more than one screen.
screen	The number of the screen where the output is to appear. The default is 0.

For example, executing the command:

```
xterm  -display  hpthere:0.1 &
```

starts an **xterm** process on the local system and displays the window on display 0, screen 1 of the **hpthere** system (assuming the local system has permission to display on **hpthere**). The window has the default size, location, and color.

Specifying the Workspace

By default, clients are started in the current workspace. Some clients have a command-line option to start the client elsewhere.

- If the client has the **-workspaceList** option, use it to specify the workspaces in which the client is displayed.
- If the client does not have the **-workspaceList** option, but it does have the **-xrm** option, you can use it. The syntax for using the **-xrm** option is:

  -xrm '*workspaceList: { *workspace_name* [*workspace name* ...] | **all** }**'**

  The character string following **-xrm** must be quoted with right-handed single quotes.

For example, the following command line starts an **xterm** window that will be displayed in two workspaces—**project1** and **extra space**.

```
xterm  -xrm '*workspaceList: project1 "extra space"' &
```

The command:

```
xterm  -display  hpthere:0.1 -xrm '*workspaceList:  project2' &
```

starts an **xterm** window in workspace **project2** on display 0, screen 1 of system **hpthere**.

Stopping Clients

If a client has data you want to save, you must save the data *before* you stop the client. For example, to save **bitmap** data, use the "Write Output" selection on the sidebar menu to save the bitmap before you stop **bitmap**.

If a terminal window is running a non-client containing data, you must stop the non-client in the approved manner before you stop the window.

After you have saved any data and exited any non-clients (in the case of terminal windows), stop the client by choosing the "Close" selection from the client's window menu.

Note that if you started a non-client as an option of creating a window, when you stop the non-client, the window will stop.

7

Running and Stopping Non-Client Programs

A non-client normally relies on a terminal instead of a window for displaying its output. To start a non-client program in a window environment, create a terminal emulation window, and then run the non-client from the command prompt in that window.

Non-clients are placed in the current workspace.

Stopping Non-Clients

Stop all non-clients in the manner approved in the instructions for that non-client. Generally, a non-client program stops automatically when it finishes executing or has a "stop" provision.

Killing Programs That Won't Stop

If you are unable to stop a program in the normal manner, you should "kill" the program before you log out. To kill a program, first try these keystrokes:

- Press CTRL c.
- Press CTRL d.
- Press q.
- Press ESC, then :, then q.

If these don't work, use the operating system "kill" command to stop the program's process:

1. Save any data that needs saving.
2. Find the PID (process ID) by executing the appropriate command for your operating system. (For example, in HP-UX execute **ps -fu** *login_name*.)
3. To kill the program, execute the kill command using the PID for the process. (For example, in HP-UX execute **kill** *-n pid*. Use *n* values of 2, then 3 if necesary, then 9 until the process is killed.)

7

Killing Cached Clients

Certain HP VUE clients are cached during a session; that is, once they are started, closing them unmaps the window but does not stop the process. If you need to halt one of these processes during a session, use the **kill** command.

For example, once the file manager is running during a session, it remains running during the current session, even if you exit all the file manager windows. If you need to halt the process, you must use the appropriate kill command for your operating system.

8

Application Resources

Resources are data used by applications to set their appearance and behavior.

This chapter covers:

- The files used by HP VUE to create the resource database.
- The various ways to change resource settings.
- The scope of resources— how specifically or generally a resource is applied.
- The syntax for specifying color and geometry resources.

How Applications Obtain Attributes

The resources for an application are obtained and managed by the **resource manager**. An application can get attributes from several different places:

- Resources directly loaded into an application's resource database:
  - Command-line options.
  - Resources loaded into the RESOURCE_MANAGER property.
  - Certain standard resource files (for example, `app-defaults` files).
- Other sources:
  - Defaults built into the client.
  - Environment variables.
  - Inter-client communications.

The following list and illustration show how applications obtain resources by means of the resource manager:

- Command-line options. These options are good for only that one instance of the application. A command-line option is the equivalent of a *client.resource* statement in a resource file.
- A host environment:
  - If an `XENVIRONMENT` variable exists, it may contain the name of a file that specifies application attributes.
  - A `$HOME/.Xdefaults-`*host* file may contain resources to be used for a specific remote host. It is read only if no `XENVIRONMENT` variable exists.
- General user resources loaded into the RESOURCE_MANAGER property. See "The RESOURCE_MANAGER Property" in the next section for more information.
- User-specific files for particular classes of applications:
  - If an `XUSERFILESEARCHPATH` variable exists, it may specify a directory of files containing application class defaults for the specific user.
  - If `XUSERFILESEARCHPATH` variable does not exist, and if an `XAPPLRESDIR` variable exists, it may specify a directory of files containing user-specific application class defaults.
  - `$HOME/`*app-class* files may contain application resources. These files are read only if `XUSERFILESEARCHPATH` and `XAPPLRESDIR` do not exist.

  For information about these variables, refer to *Programming with the Xt Intrinsics*.
- Application-specific configuration files in the `/usr/lib/X11/app-defaults` directory. Each file specifies attributes for a particular class of application. An `app-defaults` file is the equivalent of a *Class*resource* statement in a resource file. (The environment variable `XFILESEARCHPATH` may define a language-dependent location of `app-defaults`.)
- Internal defaults built into the application.

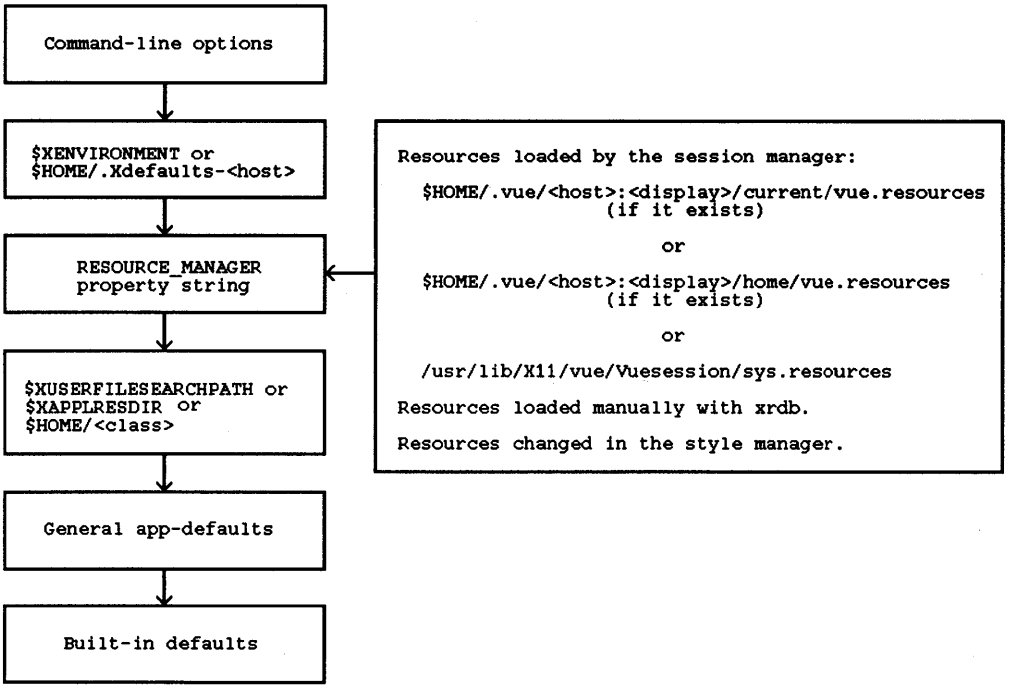


Figure 8-1. How Applications Obtain Their Resources.

A resource specified towards the top of the illustration overrides the same resource in a lower source. For instance, a resource specified in **$HOME/.vue/***host*:*display*/**current/vue.resources** overrides a resource in the **app-defaults** file; but a resource specified in the command line overrides the **vue.resources** specification.

Note

HP VUE does not automatically load **.Xdefaults** or some other personal resource file. Instead, resources are loaded by the session manager from its own database.

For additional information about how applications obtain resources, refer to the discussions of the resource database in the Xt Intrinsics documentation.

The RESOURCE_MANAGER Property

The RESOURCE_MANAGER property is a property on the root window that is treated the same way as a resource file by the resource manager.

There are several ways that attributes are loaded into the RESOURCE_MANAGER property:

- The session manager loads resources from one of these resource files. A file is loaded if the file above it in the list doesn't exist.
  - `$HOME/.vue/`*host*`:`*display*`/current/vue.resources`.
  - `$HOME/.vue/`*host*`:`*display*`/home/vue.resources`.
  - `/usr/lib/X11/vue/vuesession/sys.resources`.

  The existence of the `vue.resources` files depends on whether the user has chosen to restore the current session, and whether the user has saved a home session. (See "Session Startup" in chapter 6 for additional information.)
- During a session, the RESOURCE_MANAGER property may be modified by the the style manager and the `xrdb` client.
- Certain style manager customizations write to the RESOURCE_MANAGER property.

Ways to Change Resources

There are several ways to change a resource. The way you choose depends on:

- The nature of the resource.
- When you want the change to take effect—immediately, at the beginning of the next session, or when you choose to return to a home session.

Resources can be changed by:

- Using the style manager to change the resource interactively.
- By loading the new resources into the server's RESOURCE_MANAGER property using the X client `xrdb`.
- By hand editing a resource file.
- Using command-line options.

Changing Resources Using the Style Manager

When the system is customized using the style manager, a new resource may be written to the RESOURCE_MANAGER property. This depends on:

- The nature of the customization. Certain customizations involve resources (for example, changing workspace backdrops) and others do not (for example, adding or deleting hosts).
- The value of the resource `Vuestyle*writeXrdbImmediate`. When this resource is "False" (the default), the session manager is notified of font and mouse double-click changes, but the new resources are not written to the RESOURCE_MANAGER property until the user logs out.

Normally, changes to the RESOURCE_MANAGER property are not processed by an application until the application is restarted. However, the style manager provides the ability to communicate certain resource changes immediately to cooperating clients. For example, the workspace backdrops can be changed dynamically without restarting `vuewm`.

The session manager maintains two files for storing the RESOURCE_MANAGER property at the end of a session:

- If the user has chosen to restore the current session at next login, the resource data is written to `$HOME/.vue/`*host*`:`*display*`/current/vue.resources` when the user logs out.
- Any time during a session, the user can choose to store away the current state of the resources into the file `$HOME/.vue/`*host*`:`*display*`/home/vue.resources`.

Refer to chapter 6 for additional information about the resource files used by the session manager.

Adding and Changing Resources With 'xrdb'

The `xrdb` client allows you to manipulate the resources in the server's RESOURCE_MANAGER property. This allows you to alter the resource database during a session. The changes take effect the next time the application is run.

For information about the syntax of `xrdb`, refer to the *Using the X Window System* manual.

Adding Resources with 'xrdb'

To add resources, you can:

- Use `xrdb` interactively. This is useful for adding one or two resources.
- Create a file and merge it into the resource database. This technique is preferable when you want to add a number of resources.

Using 'xrdb' Interactively to Add Resources. To add resources interactively:

1. Execute:

   ```
   xrdb -merge -nocpp
   ```

   in a local terminal emulation window.

2. Type in the resource specifications. Each resource must be on a separate line.

3. When you've typed all the resources, press CTRL d to merge the resources and restore the shell prompt.

Adding Resources by Merging a File. This technique allows you to type the resources into a file that is then merged into the database:

1. Create a file containing the resources you want to add.

2. Execute:

   `xrdb -merge -nocpp` *filename*

Editing Resources Using the Resource Editor Action (RES_EDIT)

The resource editor action RES_EDIT lets you edit the RESOURCE_MANAGER property without manually executing the `xrdb` client.

The default RES_EDIT action uses the editor specified by the `EDITOR` environment variable. The action is written for an editor that runs in a terminal window (for example, `vi`). If the editor you want to use has its own windowing support, you will have to modify the action.

To use the default RES_EDIT action:

1. Double-click the RES_EDIT action icon in **$HOME/.vue/apps/system_apps/X11_apps**.

   The action opens a terminal window, executes **xrdb -query**, and directs the output to a temporary file. It then runs the editor defined by the **EDITOR** environment variable using the temporary file as an argument.

2. Edit the file. You can add, edit, and remove resources from the file.

3. Store the edited file using the editor's save command.

4. Close the terminal window.

   The action loads the temporary file into the RESOURCE_MANAGER properly using the **xrdb -nocpp** command. The temporary file is then removed.

There are two ways to modify the editor used by **RES_EDIT**:

- Set the **EDITOR** environment variable to the editor you want to use.
- Modify the action itself. You may have to do this if:
  - You want to use an editor other than the one specified by the **EDITOR** environment variable.
  - The editor you want to use has its own windowing support.

To modify the RES_EDIT action:

1. Copy the action, located in **/usr/lib/X11/vue/types/vue.ad**, to your personal action definition database (**$HOME/.vue/types/*.ad**).
2. Edit the copy of RES_EDIT:
   - To specify an editor other than $EDITOR, edit the line that invokes the editor. For example, if $EDITOR is some editor other than **vi**, but you want to use **vi** to edit your resources, change:

     ```
     \$EDITOR /tmp/\$\$.xrdb ;      \
     ```

     to

     ```
     /path/vi/tmp/\$\$.xrdb ;    \
     ```

   - If your editor provides its own windowing support, change the line **TERMINAL** to **NO-STDIO**. Also change the exec-string field of the action definition to remove commands for the terminal window. For example, the following exec-string invokes an $EDITOR editor that has its own windowing support:

     ```
     /bin/ksh -c '/path/xrdb -q > /tmp/\$\$.xrdb;       \
                  \$EDITOR /tmp/\$\$.xrdb ;                    \
                  /path/xrdb -nocpp  /tmp/\$\$.xrdb ;  \
                  rm -rf /tmp/\$\$.xrdb'
     ```

3. To activate the new action definition, reread the action definition database by double-clicking the REREAD_DB action icon in the **$HOME/.vue/apps/system_apps/sys_admin** directory.

Changing and Deleting Resources Using a Resource File

If you are familiar with using a resource file such as `.Xdefaults` to manage resources, you may prefer to have a single resource file that you can view and edit. If you choose to manage resources this way, you must adapt the procedure for maintaining this file, keeping in mind that:

- The session manager automatically saves and restores resources.
- The style manager writes out resources.

Use the following procedure to edit and remove resources using a defaults file:

1. Create the file by downloading the RESOURCE_MANAGER property string into a file during an active session. To do this, execute:

   `xrdb -edit` *filename*

   where *filename* is the name of your personal defaults file.
2. Edit the file to contain the resources you want.
3. Load the edited file into the resource database by executing:

   `xrdb -nocpp` *filename*

Editing Session Manager Resource Files Directly

The session manager resource files can be edited directly. However, this should not be done within an active session, since the session manager may overwrite these files during the session or at logout.

Table 8-1.
Files to Edit For Changing Resources Between Sessions.

File $HOME/.vue/host:display/ ...	When the Change Takes Effect
`current/vue.resources`	At the beginning of the next session if the user chose to restore the current session.
`home/vue.resources`	At the beginning of the next session if the user chose not to restore the current session.

Changing Resources for Cached Clients

Resource changes made to clients do not take effect until the client is restarted.

Certain HP VUE clients started from the workspace manager are cached—that is, they may be running but unmapped to the display. For example, when the file manager or help manager is closed (either from the File menu or window menu), the process is not halted. Closing these clients simply unmaps them from the display. Once these clients are started during a session, they are not killed until the user logs out, or until the client is closed manually using the operating system "'kill" command.

HP-UX systems only: `hpterm` windows started from the workspace manager (front panel) on HP-UX systems are also cached. At login, several unmapped terminal clients are started. Therefore, changes made to terminal resources may not take effect during the current session for new terminal windows mapped from the workspace manager.

Syntax of Resource Specifications

Resource files are text files. They must obey the following syntax rules:

- Each resource specification must be on a separate line. If the last character on a line is a backslash (\), the new-line following the backslash is ignored and the resource specification is assumed to continue on the next line.
- To add comments to resource files:
  - Use the exclamation (!) character. Anything to the right of the ! is interpreted as a comment. This is the preferred way of commenting all or portions of lines.
  - You can place a pound (#) character in column 1. This makes the entire line a comment. Keep in mind that you must use the **xrdb** option **-nocpp** when loading a commented resource to avoid it being interpreted as a preprocessor directive.
- The resource name is separated from the value by a colon (:) and optional spaces or tabs.

The general syntax for specifying a resource for a client is:

$$\left[\left\{\begin{matrix} client\_name \\ client\_class \end{matrix}\right\}\right]* \; resource: \; value$$

For example:

```
xterm*cursorColor: skyblue
```

sets the color of the **xterm** cursor to skyblue.

Certain clients allow you to set resources for particular parts of the client. For example,

```
xterm*scrollBar*background: mediumblue
```

sets the scrollbar on **xterm** windows to mediumblue.

Color Resources

Colors can be specified several ways:

- By the color set Id resources. The color set Id is a number representing the color set in the current palette that will be used to color display elements. There are four color set Id resources:
  - `primaryColorSetId`
  - `secondaryColorSetId`
  - `activeColorSetId`
  - `inactiveColorSetId`

  For example, the following line specifies that the background of the client `xclock` will be colored based on color set 3.

  ```
  xclock*primaryColorSetId:  3
  ```

  Color sets are covered in more detail in chapter 9.

- Using resources that take color names or rgb numbers as their values.
  - The file `/usr/lib/X11/rgb.txt` lists all the named colors.
  - The rgb numbers have the syntax:

    #*RedGreenBlue*

    where *Red*, *Green*, and *Blue* are hexadecimal numbers containing 1, 2, 3, or 4 digits that indicate the amount of that primary color used. There must be the same number of digits *for each* of the primary colors. Thus, valid color values consist of 3, 6, 9, or 12 hexadecimal digits. For example, black can be specified by any of these rgb values: `#fff`, `#ffffff`, `#fffffffff`, or `#ffffffffffff`. The style manager uses rgb values when writing out color information to its palette files.

  For example, the following line specifies the background color of `xterm` icons:

  ```
  Vuewm*xterm*iconImageBackground:   DarkSlateGrey
  ```

Use these guidelines when deciding how to color a display component:

- Use a color set Id when you want the color to reflect the current palette scheme. For example, specifying:

  ```
  xterm*primaryColorSetId:    6
  ```

  causes the background of **xterm** windows to be based on color set 6. The color of the **xterm** windows will change when the user switches to a different palette.

- Use actual color resources when you want the component to be a particular color, regardless of the current palette. For example,

  ```
  Vuewm*menu*background:    wheat
  ```

  specifies the color for window manager menus. You should be aware that the style manager will not automatically create a pleasing 3-D effect for colors specified explicitly.

The style manager's Modify Color dialog box provides controls for adjusting and viewing colors. The R, G, and B sliders are labeled with decimal values 0 through 255. To create an rgb value, convert each slider value to a two-digit hexadecimal equivalent and then combine them.

Geometry Resources

The geometry of a window is its size and location. The syntax for geometry resources is:

$$[=]\left\{\begin{array}{l} Width \times Height \\ \pm column \pm row \\ Width \times Height \pm column \pm row \end{array}\right\}$$

Use a lower-case **x** for the times sign.

Width — The width in characters (for terminal windows) or pixels (for other clients). For widths in characters, the window size depends on the font size.

Height — The height of the window in lines (for terminal windows) or pixels (for other clients). The height of a terminal window depends on the font.

column — The column location of the window in pixels.

Plus (+) values	The location of the left side of the window relative to the left side of the workspace.
Minus (−) values	The location of the right side of the window relative to the right side of the workspace.

row — The row location of the window given in pixels:

Plus (+) values	The location of the top of the window relative to the top of the workspace.
Minus (−) values	The location of the bottom of the window relative to the bottom of the workspace.

Table 8-2. Example Locations for an 80×24 Terminal Window.

To position a window here ...	Use this location ...
The upper left corner of the workspace.	+0+0
The lower left corner of the workspace.	+0−0
The upper right corner of the workspace.	−0+0
The lower right corner of the workspace.	−0−0

For example, the following line specifies that all **xterm** windows be created 80 characters wide and 24 characters high, and that they are initially placed in the upper right corner of the display.

```
xterm*geometry:    80x24-1+1
```

Scope of Customization

Scope of customization determines how generally or specifically a resource is applied. For example, you can specify that all clients have a background color of black (very general). At the other extreme, you can say that you want the softkeys of one particular **xterm** window to be red.

Scope of customization is determined by:

- Using names or classes of clients.
- Using names or classes of resources.
- Specifying particular areas of clients (for example, softkeys and scrollbars).
- Using wildcards in the resource string.

Names and Classes of Clients

Every client has both a name and a class. The name defines the specific client, while the class categorizes the client. Thus, the class is more general than the name.

Frequently, the two identifiers are very similar, and often differ only in capitalization. For example, the client named **vuestyle** belongs to class **Vuestyle**.

Resources specified by client name take precedence over resources specified by client class.

Naming a Client

You can assign a name to a particular instance of a client. This allows you to allocate resources to that client by class, by client, *and* by name.

For example, the following command line starts an instance of **xterm** named **localTerminal**.

```
xterm -name localTerminal
```

If the following resource exists in the resource database:

```
localTerminal*scrollBar:     True
localTerminal*background:    white
```

then the **localTerminal** window will have a scroll bar and be white, overriding the colors used by the current palette.

Names and Classes of Resources

Like clients, resources have both a name and a class.

An individual resource begins with a lowercase letter. For example, **foreground** refers to the foreground resource. A class resource, however, begins with an upper-case letter. For example, **Foreground** refers to the the entire class of foreground resources.

Thus, if no other specifications overruled, the line ***foreground: blue** in your resource file would make all foregrounds blue. However, the line ***Foreground: blue** would make all resources that belonged to the Foreground

class blue. This would include such resources as `foreground`, `cursorColor`, `pointerColor`, `bottomShadowColor` for softkeys, frames, icons, and mattes.

Name/Class Precedence

Specific resource specifications always have precedence over general specifications. For example, suppose a resource file contains:

```
*Foreground:            red
Xterm*Foreground:       DarkSlateGray
Xterm*foreground:       coral
Xterm*cursorColor:      green
```

The first line makes all resources of the class Foreground red. The second line overrules the first line, but *only* in the case of clients of class Xterm (of which there is only one–the `xterm` client itself). Line two makes the Foreground class resources of all `xterm` clients DarkSlateGray. Lines three and four give `xterm` clients coral foregrounds and green cursors, while the other resources of class Foreground (pointerColor, cursorColor, softkey foreground and bottomShadowColor, and scrollbar foreground and bottomShadowColor) remain DarkSlateGray for `xterm` clients.

Similarly, if a resource file contains:

```
xterm.name:                 local
Xterm*softkey*background:   wheat
Xterm*background:           pink
local*background:           white
```

then all softkey backgrounds will be wheat. For the rest of the `xterm` window, the backgrounds will vary. Windows named `local` will be white, other windows will be pink.

Wildcards and Exact Paths

The * character in a resource string is a wildcard that provides resource generality. For example, the following list of resources shows increasing specificity.

```
*foreground:                 white
xterm*foreground:           yellow
xterm*softkey*foreground:   red
```

The resource `*foreground` refers to *all* foregrounds. The more specific resources override it. All the `xterm` foregrounds will be yellow except for the foreground of the softkeys.

9

The Viewable HP VUE Clients

This chapter covers customizing the viewable HP VUE clients—that is, the clients that display windows. It covers these topics:

- How to change resources for the HP VUE clients.
- Using the **xterm** or **hpterm** terminal emulator.
- How the style manager manages colors and various system settings.
- How the style manager changes other settings.
- Client caching.
- Customizing the help manager, file manager, and xload.

Aspects of several HP VUE clients are covered in other chapters:

- Setting resources using the **xrdb** client is covered in chapter 8.
- The window manager (**vuewm**) is covered in chapter 11.
- The databases used by the file manager are covered in chapter 12.
- The workspace manager (front panel) is covered in chapter 13.

Setting Resources for the HP VUE Clients

The app-defaults (system-wide application defaults) files for the various clients contain a number of resources for customizing the appearance of the clients. The files are located in directory **/usr/lib/X11/app-defaults**, and are named according to the class name of the client. For example, app-defaults for the **vuestyle** client are located in **/usr/lib/X11/app-defaults/Vuestyle**.

Resources for a client can be modified on a system-wide or per-user basis.

- To modify system-wide resources, edit the app-defaults file for the client.
- To modify resources on a per-user basis:
  1. Copy the app-defaults file to another file. Give that file write permission.
  2. In the new copy, remove lines for resources you will not be changing.
  3. Edit the resources you want to change. Add the client class name to the beginning of the resources, so that they have the syntax:

     *Classname*resource*: *value*

     For example, to set the maximum value for the tone duration scale to 30, use the resource:

     ```
     Vuestyle*durationScale.maximum: 30
     ```

  4. Merge the file into the resource database by executing:

     xrdb -merge -nocpp *file_name*

Terminal Emulation

Terminal Characteristics

The login manager does not perform the normal console login. Consequently, your terminal characteristics may not be what you expected; you may need to set the special control characters and process group control characters with a resource.

By default, the session manager sets the following special control characters and process control characters:

Table 9-1. Control Characters for Terminals.

Control Name	Control Character
erase	^H
intr	^C
kill	^U
start	^Q
stop	^S

You can set any of the following: quit, eof, elo, swtch, brk, susp, dsusp, rprnt, flush, weras and lnext.

To set the resource for your control characters, add the following resource to your resources file.

```
*ttyModes:    name ^C
```

where *name* is the name of a control character and *C* is the character.

You can configure a single terminal window by sourcing in the appropriate files.

Process Caching for the 'hpterm' Terminal Emulator

Note

This section applies only to systems that support the **hpterm** terminal emulator. HP VUE does not use process caching for **xterm**.

Each time a session is started, several cached terminal processes are started—that is, their processes are running but their windows are not displayed. Clicking the terminal emulator button on the workspace manager accesses one of these processes and displays the window. The terminal processes started from command lines do not use these cached processes.

The system maintains two cached terminal processes; as their windows are mapped, additional unmapped processes are started to replenish the cache. When a window is closed, a process is returned to the cache.

The following table lists some of the **vuecommand** resources for customizing caching behavior and the appearance of cached windows.

Table 9-2. Terminal Caching Resources.

Resource	Description	Default
minCache	The minimum number of terminal windows to pre-start at the beginning of a session	2
maxCache	The maximum number of terminal windows to keep in the cache.	4
terminalName	The window name for the terminal window	Terminal Window
terminalIcon	The icon label	Terminal

For example, the following line specifies that the terminal name for terminal windows displayed from the workspace manager is `Local`:

```
Vuecommand*terminalName:    Local
```

The Style Manager

This section covers:

- Opening and closing the style manager.
- Resources controlling the appearance of the style manager.
- Customizations by the Style Manager.
- How the style manager sets colors and manages the color map.

Opening and Closing the Style Manager

The style manager is started by clicking the style manager control in the workspace manager. The executable file is `/usr/bin/X11/vuestyle`.

There can be only one style manager running. Attempting to start another `vuestyle` process from a command line (for example, from a menu or a terminal emulator) will fail.

Style Manager Resources

The app-defaults file (`/usr/lib/X11/app-defaults/Vuestyle`) contains a number of resources for changing the appearance of the style manager.

Style Manager Customizations

The following table lists the major categories of customizations done by the style manager.

Table 9-3. Style Manager Customizations.

Customization	Description	Covered in chapter ...
Color	Client, window manager, and workspace colors.	9
Fonts	Fonts used for system and user areas in applications.	8
Backdrop	Bitmaps used as workspace backdrops.	11
Keyboard	Click volume and autorepeat.	9
Mouse	Double-click time, acceleration, and threshold.	9
Audio	Beeper volume, tone, and duration.	9
Screen	Screen saver behavior.	9
Hosts	Specifies which remote hosts can access the local server.	4
Startup	Session startup behavior.	6

Using the 'xset' Client

The `xset` client allows you to change certain user preference options of the display. All of these setting are saved by the session manager when the session manager `queryserverSettings` resource is True. Note that hardware limitations and implementation differences may affect the results of the `xset` client.

The following functionality provided by the style manager is also provided by the **xset** client. These settings are always saved by the session manager, regardless of the value of **queryServerSettings**:

- Bell volume, pitch, and duration.
- Keyboard click volume and autorepeat.
- Mouse acceleration and threshold.
- Screen saver time.

Using the 'xhost' Client

The **xhost** client provides an alternative to the style manager Hosts dialog box for dynamically controlling access to your local system. Using **xhost**, you can add or delete a remote host's permission to access the local display server.

For example, the following command allows the remote host **hpcvfgg** to access your local display.

```
xhost  +hpcvfgg
```

The change is saved by the session manager when you log out.

How the Style Manager Sets Colors

The style manager provides the ability to select a color scheme for display components from a variety of built-in and user-created palettes.

This section covers:

- How the style manager interacts with HP VUE and X clients.
- The files used to store palette information.
- Setting the palette configuration, which determines how many color sets each palette has.
- The color sets contained within color palettes, and how those colors relate to display components.
- How to minimize color usage.

Changing Colors Dynamically

The style manager provides the ability to change the client colors.

In order for a client to respond to the style manager palettes, the following must be true:

- The client must be an HP VUE client or another client written using the OSF/Motif 1.1 or later toolkit. X clients written with other toolkits do not have the ability to adopt new color resources dynamically. For those clients, a change made in the style manager does not take effect until the client is restarted.
- There must be no other specific color resources specified for the client. (This includes user-specified resources, app-defaults, and resources built into the client.)

Color Palette Files and Their Locations

The style manager provides a set of color palettes for selecting color schemes for window frames and HP VUE applications.

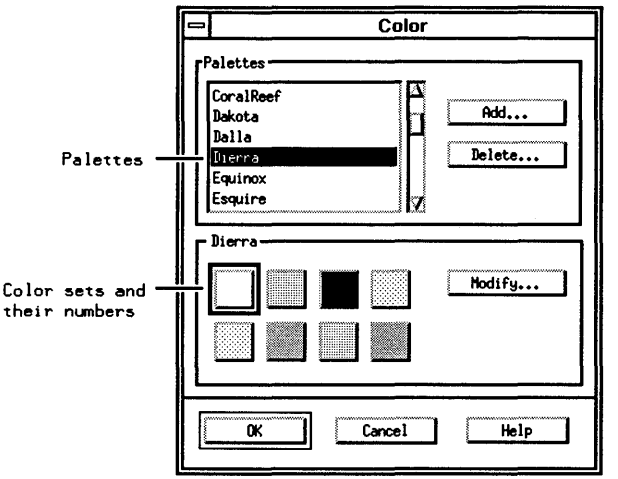


Figure 9-1. The Color Palette Dialog Box.

9

A file exists for each color palette. The resource **paletteDirectories** specifies the directories containing palette files. Its default value specifies these directories:

- The **usr/lib/X11/vue/palettes** directory, which contains files for the palettes shipped with HP VUE.
- The **$HOME/.vue/palettes** directory, which contains new palettes and palettes that were created by modifying built-in palettes.

When the user selects a palette, it is stored into the file **$HOME/.vue/***host*:*display*/**palette.vue**. This allows the session manager to restore the current palette at the beginning of the next session.

Specifying Palette Configuration

A palette consists of a group of color sets. Each set is represented by a color palette button in the Color Palette dialog box.

The number of color sets in a palette is determined by the display type and the **colorUse** resource, which has the syntax:

```
*colorUse: { B_W
           { LOW_COLOR
           { MEDIUM_COLOR
           { HIGH_COLOR
```

The default value is set according to the number of color planes being used.

Table 9-4. Palette Configuration.

Configuration	Number of Color Sets	Maximum Number of Colors	Default Number of Colors Used	Configuration is Default for this many planes ...
B_W	2	2	2	1–3
LOW_COLOR	2	12	8	4–5
MEDIUM_COLOR	4	22	22	6
HIGH_COLOR	8	42	42	7 or more

There are four palettes for B_W.

Color Sets

Each color palette button in the Color dialog box represents one color set.

There are up to five colors in a color set. The colors affect these display components:

- `foreground`. This is the foreground of the application window or window frame, and is always black or white.
- `background`. This is the background of the application window or window frame.
- `topShadowColor`. This is the color of the top and left bevels of application controls (push buttons, text boxes etc) and window frames.
- `bottomShadowColor`. This is the color of the bottom and right bevels of application controls and window frames.
- `selectColor`. This is the color of certain controls—for example, depressed buttons, editable areas, and slider bars.

The number of colors in a color set depends on two resources—`shadowPixmaps` and `foregroundColor`.

9

How Top and Bottom Shadows are Colored. The resource `shadowPixmaps` determines how the 3-dimensional effects are created on the display.

- When "True," the system creates a `topShadowPixmap` and `bottomShadowPixmap` for the bevels instead of a `topShadowColor` and `bottomShadowColor`. This reduces the number of colors in a color set by 2.
- When False, the system uses solid colors for the bevels.

Table 9-5. Default Values for the shadowPixmaps Resource.

Display	Value
B_W	ignored
LOW_COLOR	True
MEDIUM_COLOR	False
HIGH_COLOR	False

Setting the Foreground Color. The syntax for specifying the foreground color is:

```
*foregroundColor: { White   }
                  { Black   }
                  { Dynamic }
```

If the value is set to `White` or `Black`, all foregrounds are set to that color, and the number of colors in a color set is reduced by 1. The foreground will not change in response to the background color.

If the value is set to `Dynamic`, the foreground changes in response to the background color.

Table 9-6. Default Values for foregroundColor Resource.

Display	Value
B_W	ignored
LOW_COLOR	`Dynamic`
MEDIUM_COLOR	`Dynamic`
HIGH_COLOR	`Dynamic`

Computing How Many Colors a Palette Uses

The minimum number of colors used by the style manager is the number of colors used by a palette. This number depends on the values for these style manager resources:

- `colorUse`.
- `shadowPixmaps`.
- `foregroundColor`.

The number of colors in a palette is calculated by multiplying the number of color sets in the palette by the number of colors in each color set, then adding black and white (these are always allocated by the server).

9

For example, consider a configuration with these settings:

```
*colorUse:          MEDIUM_COLOR
*shadowPixmaps:     True
*foregroundColor:   White
```

MEDIUM_COLOR configurations have 4 color sets. Each color set includes a `background` color and `selectColor`. (There is no `topShadowColor`, `bottomShadowColor`, or `foreground`. The bevels are colored using pixmaps and the foreground is always white.) Therefore, the number of colors used by a palette is 4 × 2 + black + white = 10.

Here is another example:

```
*colorUse:          HIGH_COLOR
*shadowPixmaps:     True
*foregroundColor:   Dynamic
```

In this configuration, there are 8 color sets. Each color set includes a `background`, `foreground` (since `foregroundColor` is set to `Dynamic`), and select color. Therefore, the number of colors used by a palette is 8 × 3 + black + white = 26.

Mapping Display Components to Color Sets

Color sets in a palette are mapped to display elements by the following resources. The value of each resource is an integer representing a color set.

Table 9-7.
Resources for Assigning Color Sets to Display Elements.

Resource	Display Element	Default Colorset
`activeColorSetId`	Active frame color.	1
`inactiveColorSetId`	Inactive frame color.	2
`primaryColorSetId`	Application's primary colors (main background area).	3
`secondaryColorSetId`	Application's secondary colors (menubar, menus, dialog boxes).	4

By default, all HP VUE clients use the same `primaryColorSetId` to group them visually. However, the app-defaults include client-specific `secondaryColorSetId` resources:

Table 9-8. Default Secondary Color Sets.

Client	Client-specific secondaryColorsetId
Style manager	7
Help manager	6
File manager	5

This provides a degree of color coding—for example, the user can visually associate dialog boxes with their parent application windows by matching the colors of the menubars and dialog box backgrounds. Note that this color differentiation is only visible on HIGH_COLOR displays. For

MEDIUM_COLOR and LOW_COLOR displays, color set Id's of 7, 6, and 5 are mapped to available color sets.

The following example shows how you would group all **xterm** windows by specifying that they use color set 8 for their primary color.

```
xterm*primaryColorSetId:  8
```

Color Resources Affected by the Style Manager

When the user changes palettes, color values in the newly chosen palette are written to the following resources in the server's RESOURCE_MANAGER property string:

- `*background.`
- `*foreground.`

These resources will determine the background and foreground colors of clients other than those created using the OSF/Motif 1.1 toolkit or toolkits following OFS/Motif 1.1. The resources take effect the next time the clients are started.

If the user has a more specific color specification in the resource database, that resource overrides the style manager. For example, if the resource database contains:

```
xterm*background: green
vuewm*topShadowColor: pink
```

then those colors will persist regardless of changes made using the style manager.

Note

Specific color resources affect only the display element controlled by the resource. Thus, it may require multiple resources to create a pleasant 3-D effect. For example, specifying `topShadowColor` for window frames does not provide appropriate colors for other frame elements.

Turning Off Dynamic Colors

The `dynamicColor` resource provides a way to reduce color usage. It controls whether clients change colors dynamically and has a default value of "True"—clients change color when the user switches palettes. When the resource is set to "False," clients will not respond dynamically to palette changes.

This is how setting `dynamicColor` to "False" reduces color usage. When `dynamicColor` is "True," clients that cannot change colors dynamically allocate different cells in the color map than those used by clients that change colors dynamically. When `dynamicColor` is set to "False," all clients share the same color cells.

Reducing Color Usage

Several factors contribute to color usage by HP VUE:

- The style manager must use some color cells for its active color palette.
- HP VUE clients may require additional colors. For example, the style manager uses additional color cells for its red, green, and blue scales; `vuelogin` requires colors for the login screen.
- Applications may specify client-specific color resources (for example, `bitmap*foreground: yellow`). Colors specified this way use additional color cells.
- Client that do not change colors dynamically (clients built using OSF/Motif prior to version 1.1, or non-OSF/Motif clients) use different color cells than clients that change color dynamically.

If the system runs out of color cells, you may want to reduce the number of colors in use. There are several ways to do this:

- Let the style manager handle color resources as much as possible. In other words, minimize color resource specifications in resource files.
- Use pixmaps instead of colors for the shadow colors. (Set the **shadowPixmaps** resource to True.
- Use White or Black for the foreground color, rather than letting the system determine the color dynamically (use the **foregroundColor** resource).
- After the user has settled on a color scheme, set the ***dynamicColor** resource to False.
- Set the **colorUse** resource to a lower value. For example, if you have a MEDIUM_COLOR display, setting **colorUse** to **LOW_COLOR** will reduce the number of colors used by the current palette.

Changing Mouse Double-Click Time

Changing the mouse double-click time in the style manager changes the resource **multiClickTime**. By default, the style manager notifies the session manager of the change, and the new resource specification is written to **vue.resources** when the user logs out. Therefore, the change does not take effect until the next session.

When the resource **Vuestyle*writeXrdbImmediate** is "True," the change is written immediately to the RESOURCE_MANAGER property string, and applications use the new double-click time the next time they are started.

Other Mouse, Keyboard, and Screen Customizations

Changes to mouse acceleration and threshold, beeper settings, keyboard click and autorepeat, and screen saver parameters are implemented by changing preferences for display options. (These changes were previously done using the **xset** client). HP VUE does not support changes to beeper settings for Domain/OS systems.

The changes take effect immediately and are saved by the session manager in its **vue.settings** file.

When Customizations Take Effect

These factors affect when customizations take effect:

- Changes that are implemented by changing preferences for display options take place immediately.
- Some resource changes take effect dynamically—that is, they take effect without having to restart the application. An example is changes to the `Vuewm*backdrop*image` resource for changing backdrop bitmaps.
- In general, the state of an attribute is undefined when the dialog box for that resource is open. Users should close all style manager dialog boxes before logging out in order for changes made in them to be written out by the session manager.
- Font and mouse double-click changes take effect the next time the user logs in (by default), or the next time the application is restarted.

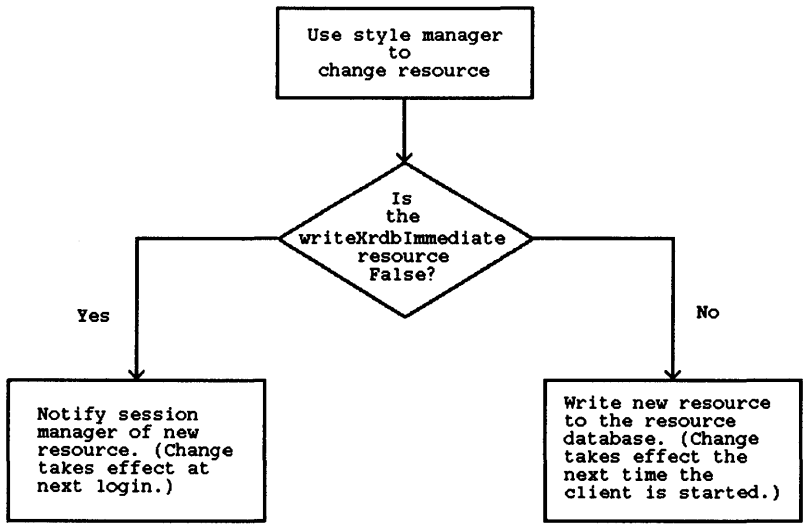


Figure 9-2.
The writeXrdbImmediate Resource Affects Font and Mouse Double-Click Changes.

9

Customizing the Help Manager

This section covers:

- How opening and closing the help manager windows is related to the actual **vuehelp** process.
- Customizing the appearance of the help manager windows.
- Configuring systems to access the HP LaserROM/UX from the Help Viewer's Browse menu.

Opening and Closing the Help Manager

The help manager is started:

- By clicking its control in the workspace manager.
- By selecting an entry in the Help menu of an application window.
- When the user clicks [Help] in a dialog box.

There can be only one **vuehelp** process running during a session. Its process has three main windows—the Help Viewer, Help Index, and Help Snapshot. All applications using the help manager share the Viewer and Index windows.

You should not attempt to start a second **vuehelp** process.

The help manager is a cached client—that is, once it is started during a session, the process remains running until the session is ended. Closing all help manager windows unmaps the windows from the display but keeps the **vuehelp** process running.

Help Manager Resources

The app-defaults file (**/usr/lib/X11/app-defaults/Vuehelp**) contains a number of resources for customizing the appearance of the help manager windows.

In addition to basic font resources, there are resource specifications for the three help manager windows. These have the syntax:

```
Vuehelp*[{index | topic | viewer}].resource: value
```

`index`	The Help Index window.
`topic`	The Help Viewer window.
`viewer`	The Help Snapshot window.

For example, to expand the width of the Help Viewer to 80 columns, use the resource:

```
Vuehelp*topic.columns: 80
```

Accessing the HP LaserROM/UX From the Help Manager

Note

This section applies only to systems that support HP LaserROM/UX.

The Browse menu in the Help Viewer window contains an entry for accessing the HP LaserROM/UX. In the default configuration, the menu item is greyed out.

To activate the menu item:

- Copy the file **/usr/lib/X11/vue/types/laserrom.ad** to **$HOME/.vue/types**.
- Uncomment the action definition in the file by removing the **#** at the beginning of the line:

  ```
  LASERROM * ${ACTION_L} ${ACTION_S} COMMAND NO-STDIO %LocalHost% laserrom
  ```

  The action assumes that the HP LASERROM/UX application and database are located locally. If necessary, modify the action to access the appropriate remote system.

- Log out and then back in. The HP LaserROM/UX item will be active the next time the Help Manager is started.

The File Manager

The executable file is **/usr/bin/X11/vuefile**.

Opening and Closing the File Manager

The file manager is started by clicking the file manager control in the workspace manager.

The user can open multiple file manager windows; however, there will be only one **vuefile** process running that supports all the opened file manager windows. You should not attempt to start another **vuefile** process from a command line (for example, from a menu or terminal emulator).

The file manager is a cached client—that is, once it is started during a session, the process remains running until the session is ended. Closing all file manager windows unmaps the windows from the display but keeps the **vuefile** process running.

Viewing Large Files

By default, the VIEW action in the file manager's Action menu (for data files) returns an error for files larger than 512,000 characters. The maximum number of characters can be changed using the **vuehelp** resource **fileSize**. It has the syntax:

Vuehelp*fileSize: *number_of_characters*

For performance reasons, you should not specify a value larger than the default unless necessary.

The 'xload' Client

The default workspace manager contains an **xload** client running in it. It is started with the **nolabel** command-line option. **xload** displays a histogram of the current system load of the local system.

By default, **xload** measures the average load on the system using a scale of 0 (no load) to 1 (a single division). The **-scale** *division* option specifies the minimum number of divisions to be used.

If the load exceeds the number of divisions, extra divisions are drawn automatically.

You can customize the workspace manager **xload** by:

- Restarting it with new command-line options.
- Specifying resources for it using its resource name **vueload**. For example, the following resource specifies that **xload** will use color set 8 for its background color:

```
vueload*primaryColorSetId:  8
```

To kill and restart **xload**:

1. Determine the pid (process-ID) of the **xload** process by executing the appropriate operating system command. (For example, the HP-UX command **ps -fu** *login_name* **| grep xload**).
2. Kill the process by executing the appropriate kill command (for example, in HP-UX: **kill -9** *pid*
3. Restart **xload** by executing:

   ```
   xload -name vueload [-option]... &
   ```

   Make sure you include the **-nolabel** option if you do not want a label in the histogram.

   For example, the following command line starts **xload** with a scale factor of 3.

   ```
   xload -name vueload -nolabel -scale 3 &
   ```

9

10

Using Fonts

A **font** is a type style in which text characters are printed. HP VUE and the X Window System include a variety of fonts:

Bitmapped fonts are made from a matrix of dots. The font is completely contained in one file. Many files are needed to have a complete range of sizes, slants, and weights.

Scalable typefaces are each defined by a file containing a mathematical outline used by the system to create a bitmapped font for a particular size, slant, or weight. Scalable fonts are not available on systems running HP-UX releases prior to HP-UX 8.0.

This chapter covers:

- Using the HP VUE Style Manager to set fonts.
- Setting font resources for HP VUE.
- Displaying samples of fonts.

Note

For additional information about fonts, refer to the manual *Using the X Window System.*

Specifying Fonts Using the Style Manager

This section covers:

- The resources used to specify the font choices displayed by the style manager Fonts dialog box.
- How the style manager writes out the user's choice to the general font resources that control the actual fonts used by the clients.

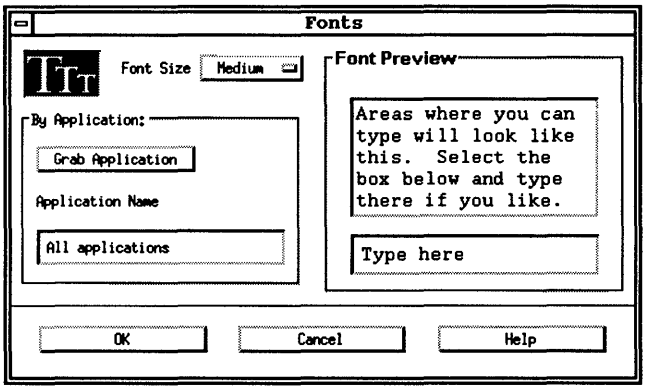


Figure 10-1. Fonts Dialog Box

Font Choices Displayed by the Style Manager

Selecting a font size from the Fonts dialog box option menu chooses two fonts:

- The system font. It will be used for system areas such as menu bars, menus, push buttons, toggle buttons, and labels.
- The user font. It will be used for text entered into windows.

The following style manager resources specify the fonts available to the user in the Fonts dialog box. They use the syntax:

`Vuestyle*`*resource*`:` *value*

where *value* can be an X logical font description (xlfd) or a font alias.

Table 10-1. Resources for Setting Style Manager Fonts.

Resource	Display Resolution	Description

`smallSysFontMR`
`mediumSysFontMR`
`largeSysFontMR`
	Medium	System fonts used for menu bars, menus, push buttons, labels, and other system areas.

`smallSysFontHR`
`mediumSysFontHR`
`largeSysFontHR`
	High	System fonts used for menu bars, menus, push buttons, labels, and other system areas.

`smallUserFontMR`
`mediumUserFontMR`
`largeUserFontMR`
	Medium	Fonts used for text the user types.

`smallUserFontHR`
`mediumUserFontHR`
`largeUserFontHR`
	High	Fonts used for text the user types.

The default values for these resources are in `/usr/lib/X11/app-defaults/Vuestyle`.

General Font Resources Accessed by the Style Manager

When the user selects a font size from the Fonts dialog box, the change affects the following resources:

Table 10-2. General Font Resources.

Resource Written	Description	Font Written
`Font`	General user font for non-OSF/Motif and OSF/Motif 1.0 clients.	User
`FontList`	Displayed in system areas of HP VUE clients and other clients created using the OSF/Motif toolkit.	System
`XmText*FontList` `XmTextField*FontList`	Displayed in the text entry boxes of HP VUE clients and other clients created using the OSF/Motif toolkit.	User
`systemFont` `userFont`	Used by application developers to specify the current system and user fonts for their applications.	

For example, if the user chooses the "Large" option from the Fonts dialog box, the following resources will be in effect on a high resolution display either at the next session (the default) or when applications are started (see the next section for more information).

`*Font:`	*Value of* `largeUserFontHR`
`*FontList:`	*Value of* `largeSysFontHR`
`*XmText*FontList:`	*Value of* `largeUserFontHR`
`*XmTextField*FontList:`	*Value of* `largeUserFontHR`
`*systemFont:`	*Value of* `largeSysFontHR`
`*userFont:`	*Value of* `largeUserFontHR`

Choosing When Font Changes Take Effect

By default, changes made in the Fonts dialog box are reported to the session manager, which writes them out to `$HOME/.vue/`*host*`:`*display*`/current/vue.resources` at the end of the current session. Thus, changes do not take effect until the beginning of the next session.

When the resource `writeXrdbImmediate` is set to "True," font changes are written to the resource database when the user clicks [OK] in the Fonts dialog box. The new font will be used for clients started after the change is made.

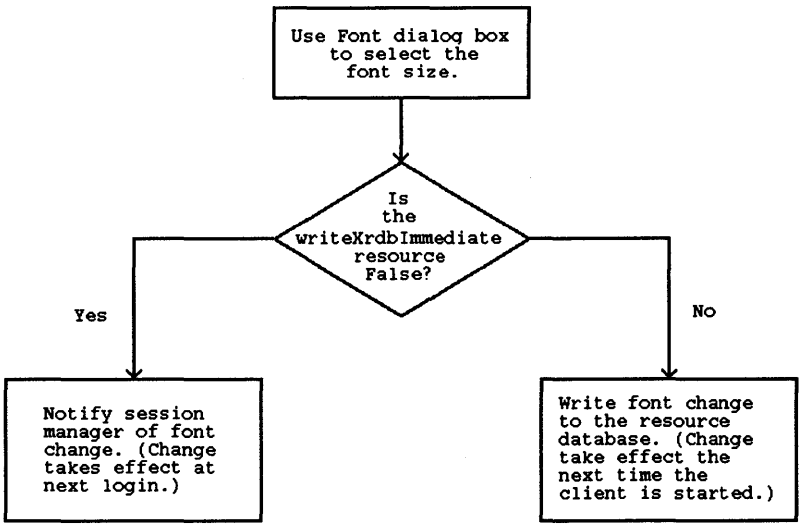


Figure 10-2. Specifying When Font Changes Take Effect.

Due to caching of some HP VUE clients, closing a client window (from the File or window menu) and starting another one from the workspace manager may not start a new instance of the client. Therefore, font changes written immediately to the resource database may not be incorporated into clients started from the workspace manager until the user logs out and back in.

Specifying Fonts For Particular Clients

You can specify fonts for particular clients by:

- Using the [Grab Application] button in the Fonts dialog box to specify the client name.
- Manually specify font resources for a particular name or class of clients.

Syntax for Client-Specific Fonts

Fonts for a particular client can be specified by the syntax:

$$\left\{ \begin{matrix} client\_class \\ client\_name \end{matrix} \right\} * \ fontresource: \ fontname$$

fontresource The name of the resource for the client.

fontname The name, alias, or xlfd name for the font.

Setting Client-Specific Fonts with the Style Manager

The [Grab Application] button in the Fonts dialog box grabs the application name. For example, if you grab an `xterm` window, the following font resources will be written to the resource database.

```
xterm*Font:                    fontname
xterm*FontList:                fontname
xterm*XmText*FontList:         fontname
xterm*XmTextField*FontList:    fontname
xterm*systemFont:              fontname
xterm*userFont:                fontname
```

If the **xterm** that was grabbed had been started by the command line:

```
xterm -name remote_terminal
```

then these resources would be written out:

```
remote_terminal*Font:                    fontname
remote_terminal*FontList:                fontname
remote_terminal*XmText*FontList:         fontname
remote_terminal*XmTextField*FontList:    fontname
remote_terminal*systemFont:              fontname
remote_terminal*userFont:                fontname
```

Specifying Resources Manually

You can add client-specific font resources manually using **xrdb**. For example, the following line in your resource file changes the font of **xterm** windows to the monospace font **-*-courier-bold-r-*-8-***:

```
Xterm*font: -*-courier-bold-r-*-8-*
```

Displaying a Font (xfd)

You can display the complete character set of any valid HP VUE font using the **xfd** client.

The **xfd** client is described in the *Using the X Window System* manual.

11

The Window Manager

The window manager is a client that:

- Provides multiple workspaces and the ability to switch from one to another.
- Manages the general appearance of window frames and icons.
- Creates the window menu and the workspace menu.
- Manages mouse button bindings and keyboard key bindings.
- Sets the focus policy. The focus policy determines how the user directs the system to change the input focus from one window to another.

The window manager also creates the workspace manager (also called the front panel). Customizing the workspace manager is covered in chapter 13.

Running the Window Manager

The window manager client **vuewm** is started automatically by the session manager. By default, it is started on one screen.

For configurations not running the session manager, the window manager is started using the syntax:

vuewm [*-option ...*]

11 Window Manager Resource Files

This section covers:

- How the window manager obtains client-specific resources for its appearance and behavior.
- The resource configuration files used specifically by the window manager.

How the Window Manager Obtains Resources

Like other clients, the window manager obtains its resources from a variety of sources. The primary sources are:

- The file **/usr/lib/X11/app-defaults/Vuewm**.
- The resources loaded into the RESOURCE_MANAGER property at the beginning of an HP VUE session.
- Resources manually loaded into the RESOURCE_MANAGER property using the **xrdb** client.

Changes to **vuewm** resources take effect when the window manager is restarted.

How clients obtain their resources is covered in more detail in chapter 8.

Window Manager Configuration Files

The window manager uses one of the following files to set up the workspace manager, window and workspace menus, and key and button bindings:

- The file specified by the resource `Vuewm*configFile`.
- `$HOME/.vue/vuewmrc`.
- `/usr/lib/X11/vue/Vuewm/sys.vuewmrc`. This is the default file shipped with HP VUE. It is used if `$HOME/.vue/vuewmrc` doesn't exist.

To create `$HOME/.vue/vuewmrc`, copy the default file, give it write permission, and then modify it.

Caution

The update process for HP VUE replaces any previous version of `/usr/lib/X11/vue/Vuewm/sys.vuewmrc`. If you modify this file and want to preserve your changes, you should move the file before updating.

If the `LANG` environment variable has been set, the system searches for language-dependent files:

- The file specified by the resource `Vuewm*configFile`.
- `$HOME/.vue/%L/vuewmrc`.
- `/usr/lib/X11/vue/Vuewm/%L/sys.vuewmrc`.

Changes to the configuration file take effect when the window manager is restarted.

11 Syntax For Window Manager Resources

The syntax for specifying window manager resources is:

`Vuewm`[**scr*][**wksp*][**comp*][**subcomp*]**resource*: *value*

scr Name of the screen for which the resource is applied.

wksp Name of the workspace for which the resource is applied.

comp A `vuewm` component:

`backdrop`	Workspace backdrops.
`client`	Client window frames.
`icon`	Icons.
`feedback`	Confirmation and resize boxes.
`menu`	Menus managed by `vuewm` (window menu, workspace menu).
`frontPanel`	The workspace manager.
`workspacePresence`	The Workspace Presence dialog box.
`workspaceController`	The Rename Workspace dialog box.

subcomp A subcomponent of a `vuewm` component:

`title`	Title bar; subcomponent of `client`.
`Switch`	The workspace switch; subcomponent of `frontPanel`.

The syntax allows you to specify resources that will be applied with varying degrees of specificity. Specifically-applied resources override general specifications.

The following examples illustrate applying one particular resource, **background**, to various **vuewm** components:

`Vuewm*background: blue`	Colors all **vuewm** background elements blue.
`Vuewm*client*background: maroon`	Colors client window frames maroon.
`Vuewm*client*title*background: pink`	Colors the title bar in client window frames pink.

The result of these resources is that the client window frames are maroon and title bars in the window frames are pink. All other **vuewm** backgrounds (for example, menus, feedback boxes, icons) are blue.

Default Screen Names

When the style manager writes out resources, it uses the following default screen names, obtained from the type of display.

Table 11-1. Default Screen Names.

	High Resolution	Medium Resolution
High Color	`HighResHighColor`	`MedResHighColor`
Medium Color	`HighResMedColor`	`MedResMedColor`
Low Color	`HighResLowColor`	`MedResLowColor`
Monochrome	`HighResMono`	`MedResMono`

11 Adding or Deleting Workspaces

The default configuration provides six workspaces. The workspace manager provides the ability to rename these workspaces. However, to create or delete workspaces, you must use the **workspaceList** resource. The value of the resource is a list containing one or more workspace names separated by spaces. Workspace names containing spaces must be quoted.

For example, the entry:

```
Vuewm*workspaceList:  Main Remote "project #1" "project #2"
```

creates an environment with four workspaces with the specified names.

Workspace names must be unique—that is, they cannot be the same as any client name or class, or any screen name.

Appearance of Window Frames

This section describes the resources for changing the following aspects of window frame appearance:

Color	The color of foreground, background, top and bottom shadows, menus, and dialog boxes.
Tile	The mixture of foreground and background color that composes the pattern of the frame surface.
Font	The type style and size used in the title bar, menus, and icon labels.
Components	Whether or not to include titlebars, control buttons, resize handles, frame mattes, etc.

Dynamic Control of Frame Color

The window manager obtains the colors for the window frames from the current color palette. The display components are mapped to palette color sets by the following resources:

Table 11-2.
Color Set Id Resources Affecting Window Manager Colors.

Resource	Display Component	Default Color Set
`activeColorSetId`	active window frames	1
`inactiveColorSetId`	inactive window frames, menus, dialog boxes, icons	2

The syntax for changing theses resources is:

`Vuewm*`{`activeColorSetId` | `inactiveColorSetId`}`:` *colorset*

For example, the resource **background**, which determines the background color of inactive window frames, defaults to the inactive color set. If the inactive color set has been assigned to palette color set 5 by the resource:

```
Vuewm*inactiveColorSetId:  5
```

then the background of inactive window frames is obtained from color set 5 of the current color palette.

Refer to "Mapping Display Components to Color Sets" in chapter 9 for additional information.

Window Frame Color Resources

The general syntax for explicitly coloring window frames is:

`Vuewm*client*`*resource*`:` *color*

11

When you use resources to explicitly color a part of a window frame, that component no longer changes color dynamically; the resource specification overrides any changes attempted by the style manager.

Table 11-3. Window Frame Color Resources.

Resource	Definition	Default Color Set
`background`	Background of inactive frames.	inactive
`topShadowColor`	Left and upper bevel of inactive frames.	inactive
`bottomShadowColor`	Right and lower bevel of inactive frames.	inactive
`foreground`	Foreground (title bar text) of inactive frames.	inactive
`activeBackground`	Background of the active frame.	active
`activeTopShadowColor`	Left and upper bevel of the active frame.	active
`activeBottomShadowColor`	Right and lower bevel of the active frame.	active
`activeForeground`	Foreground (title bar text) of the active frame.	active

For example, the following lines specify frame colors that override the default color sets.

```
Vuewm*client*activeBackground:    MediumBlue
Vuewm*client*background:          SlateBlue
```

Window Frame Pixmap Resources

Pixmaps create shades of colors without adding additional colors to the color map. Pixmaps can be used as **tiles** that provides a visual texture by "mixing" the foreground and background colors into a color pattern.

Table 11-4. Window Frame Tiling Resources.

Resource	Frame Element Tiled
`backgroundPixmap`	Background of inactive frames.
`bottomShadowPixmap`	Right and lower bevels of inactive frames.
`topShadowPixmap`	Left and upper bevels of inactive frames.
`activeBackgroundPixmap`	Background of the active frame.
`activeBottomShadowPixmap`	Right and lower bevels of the active frame.
`activeTopShadowPixmap`	Left and upper bevels of the active frame.

The following illustration shows the acceptable values for pixmap resources. The pixmap is created from a mixture of foreground and background colors.

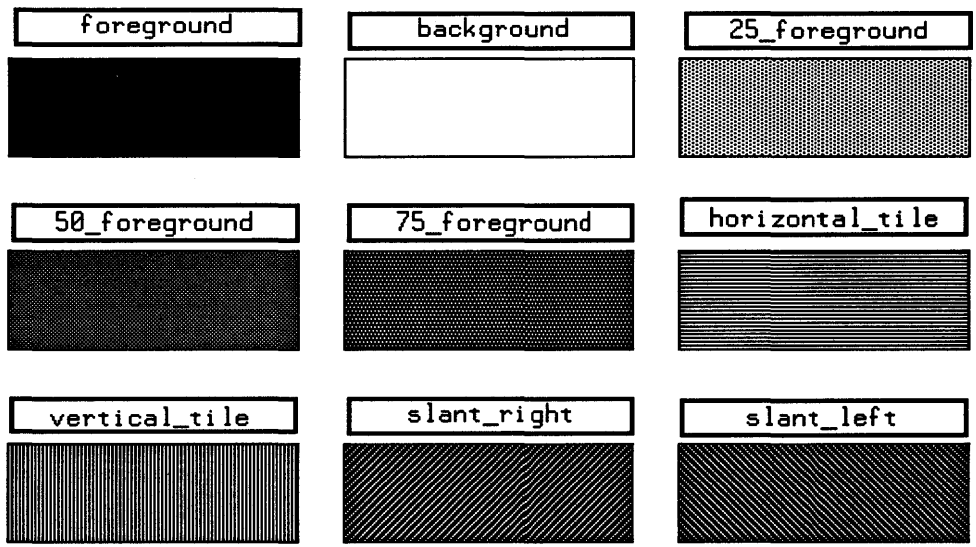


Figure 11-1. Valid Tile Values.

For example, the following lines provide tiling for the inactive windows.

```
Vuewm*client*background:          White
Vuewm*client*foreground:          Black
Vuewm*client*backgroundPixmap:    horizontal_tile
```

Font Resources

The default font resources for the text of the window manager are set by the style manager. You can view the current value by executing:

```
xrdb -q | grep fontList
```

Additional default font (`fontList`) resouces are provided in `/usr/lib/X11/app-defaults/Vuewm`.

Use the `fontList` resources to override the default fonts. The resources can use any valid font name as its value.

For example, the resources:

```
Vuewm*icon*fontList:                                    courB12
Vuewm*HighResHighColor*frontPanel*vuewmdate*fontList1: courB18
```

causes the window manager to use the font with font alias `courB12` for icons, and `courB18` for the first line of the date in the workspace manager.

Refer to the vuewm(1X) man page for additional information about font resources.

Specifying the Components of the Window Frame

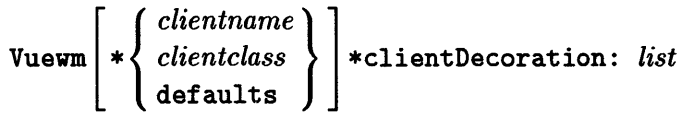
The window manager has two resources for customizing the contents (decoration) of the window frame.

Table 11-5. Window Frame Decoration Resources.

Resource	Description
`clientDecoration`	The decoration for client window frames.
`transientDecoration`	The decoration for frames around transient windows (secondary windows—for example, dialog boxes.)

Functionality provided by decoration that has been removed can still be accessed by binding its functions to mouse buttons or key presses.

The syntax for the `clientDecoration` resource is:



The syntax allows you to specify decoration for particular instances or classes of clients. The `defaults` reserved word is used to specify decoration for clients of unknown class.

The syntax for the `transientDecoration` resource is:

```
Vuewm*transientDecoration: list
```

The value of the resources, *list*, is a list of frame elements. It has the syntax:

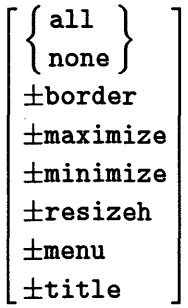


Table 11-6. Window Frame Elements.

Frame Element	Description
`all`	Include all frame elements (default value).
`border`	Window border.
`maximize`	Maximize button (includes title bar).
`minimize`	Minimize button (includes title bar).
`none`	Include no window frame elements.
`resizeh`	Resize border handles (includes border).
`menu`	Window menu button (includes title bar).
`title`	Title bar.

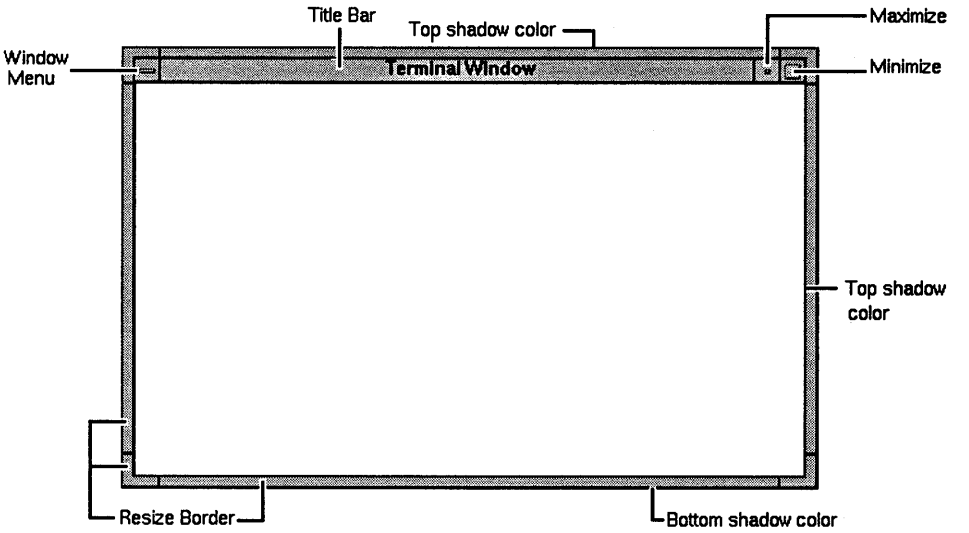


Figure 11-2. Frame Elements.

If the first element in the list is preceded by a plus (+) sign or no sign, the workspace manager starts with no frame elements and adds the elements in the

list. If the list begins with a minus (−) sign, the window manager starts with a complete frame and subtracts the elements in the list.

For example, to remove the maximize button and resize handles from all client windows, use:

```
Vuewm*clientDecoration: -maximize-resizeh
```

To specify a border with only a title bar and window menu button around a particular **xterm** window started as **xterm -name pronto**, use:

```
Vuewm*pronto*clientDecoration:    +menu
```

Specifying Resources for the Frame Title Bar

Resources can be specified separately for the title bar using the syntax:

```
Vuewm*client*title*resource: value
```

For example, the following resource specifies a pink title bar for the active background of all client windows.

```
Vuewm*client*title*activeBackground:    pink
```

Matting Clients

A matte is a 3-D border just inside the window between client area and window frame. The syntax for specifying mattes is:

$$\texttt{Vuewm}\left[*\left\{\begin{array}{l}\textit{clientname}\\ \textit{clientclass}\\ \texttt{defaults}\end{array}\right\}\right]*\textit{MatteResource}:\ \textit{value}$$

Table 11-7. Matte Resources.

Resource	Description	Default Value
matteWidth	Width of the matte.	0 (no matte)
matteBackground	background color.	**vuewm** background
matteTopShadowColor	Left and upper bevel colors.	Lightened **matteBackground** color.
matteBottomShadowColor	Right and lower bevel colors.	Darkened **matteBackground** color.
matteForeground	Matte foreground color.	Darkened **matteBottomShadowColor**.
matteBottomShadowPixmap	Right and lower bevel tiling.	Client bottom shadow color
matteTopShadowPixmap	Left and upper bevel tiling.	Client top shadow color

For example, the following line creates a 10-pixel-wide LightBlue matte for a particular **xterm** window started as **xterm -name scribe**.

```
xterm*scribe*matteWidth:        10
xterm*scribe*matteBackground:   LightBlue
```

Appearance of Particular Components

11

You can specify the appearance of particular objects managed by the window manager using the syntax:

```
                 menu
                 icon
                 client
                 backdrop
Vuewm*           feedback          *resource: value
                 Switch
                 workspaceController
                 workspacePresence
```

For example, the following lines specify background colors for the window manager menus and for the workspace presence dialog box.

```
Vuewm*menu*background:                SkyBlue
Vuewm*workspacePresence*background:   MediumBlue
```

The backdrop resources apply to a specific workspace. See "Workspace Backdrops" later in this chapter for additional information.

Customizing Icons

This section covers:

- The graphical parts of an icon.
- Changing the appearance and placement of icons.
- Using an iconbox to hold icons.

11

Parts of an Icon

Icons have two parts (decorations)—a text label and a graphics image.

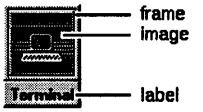


Figure 11-3. Graphical Components of an Icon.

Icon Label

The label is usually supplied by the client (via the **WM_ICON_NAME** window property). Some clients provide a command-line option for changing the label. Icon labels are truncated on the right to the width of the icon image.

Icon Image

An icon image is a pixmap. It is supplied by one of several sources, listed in the order of precedence:

- A user-supplied supplied image by means of the `iconImage` resource.
- A client-supplied icon image—an icon window or a bitmap— supplied by the client through the 'WM_HINTS" window property.)
- A built-in default icon image supplied by **vuewm**. This image is used if neither the user nor the client specifies an icon image.

The resource `useClientIcon` interchanges the precedence of user-supplied icon images and client-supplied icon images. The default value is "False." When the resource is set to "True," client-specified icon images have precedence over user-supplied icon images.

Specifying the Icon Decoration

Use the `iconDecoration` resource to select the parts of an icon to be displayed:

Table 11-8. The Values That Control Icon Decoration.

Appearance	Resource Value
Label only.	`label`
Image only.	`image`
Both label and image.	`label image`
An untruncated active label.	`label activelabel`

Dynamic Colors for Icons

The default color for icon backgrounds comes from the color set specified by the `inactiveColorSetId` resource. Thus, icons are the same color as inactive window frames.

11 Coloring and Tiling Icons

The resources that control icon color and tile have the syntax:

$$\texttt{Vuewm}\ \left[\ *\left\{\begin{array}{l}\textit{clientclass}\\ \textit{clientname}\\ \texttt{default}\end{array}\right\}\ \right]\ *\textit{resource}\texttt{:}\ \textit{value}$$

Table 11-9. Coloring Icons with Window Manager Resources.

Resource	Description
`iconImageBackground`	Icon image background.
`iconImageTopShadowColor`	Left and upper bevel of icon image.
`iconImageBottomShadowColor`	Right and lower bevel of icon image.
`iconImageForeground`	Icon image foreground color.
`iconImageBottomShadowPixmap`	Tiling of right and lower bevels of an icon image.
`iconImageTopShadowPixmap`	Tiling of left and upper bevels of an icon image.

Icon Frames

The syntax for changing the appearance of the frame elements of icons is:

`Vuewm*icon*`*resource*`:` *value*

For example, the following line changes the font used in the icon label:

```
Vuewm*icon*fontList: courB012
```

Organization of Icons

There are two ways to organize icons:

- Stand-alone icons. These icons do not use an icon box. This is the default configuration.
- In an icon box.

The choice of organization is persistent across workspaces. Thus, if you choose to use the icon box, it will be present in all workspaces.

Stand-Alone Icons

Resources control where icons are placed on the screen and their size.

Icon Placement. The resources that place icons have the syntax:

`Vuewm*`*resource*`:` *value*

You can have the window manager place the icons according to a scheme you specify, or you can place the icons yourself.

Table 11-10. Icon Placement Resources.

Resource	Description	Default
`iconAutoPlace`	Determines whether the window manager automatically places icons.	True
`iconPlacement`	A placement scheme for icons when `iconAutoPlace` is "True."	left top
`iconPlacementMargin`	The distance between screen edge and icons when `iconAutoPlace` is "True."	The default space between icons

The `iconPlacement` resource can have these values:

11

Table 11-11. Schemes for Automatic Placement of Icons.

Placement Scheme	Resource Value
Left to right, top of the screen.	`left top`
Right to left, top of the screen.	`right top`
Left to right, bottom of the screen.	`left bottom`
Right to left, bottom of the screen.	`right bottom`
Bottom to top, left side of the screen.	`bottom left`
Bottom to top, right side of the screen.	`bottom right`
Top to bottom, left side of the screen.	`top left`
Top to bottom, right side of the screen.	`top right`

For example, to specify automatic placement of icons starting at the top of the screen and proceeding down the right side, use:

```
Vuewm*iconAutoPlace:    True
Vuewm*iconPlacement:    top right
```

Icon Size. Icon images have a default maximum and minimum size. The size limits can be changed to other values within the allowable bounds.

Table 11-12. The Maximum and Minimum Sizes for Icon Images.

	Maximum Size	Minimum Size
Default	50×50 pixels	32×32 pixels
Allowable	128×128 pixels	16×16 pixels

The maximum and minimum sizes are changed with these resources:

Table 11-13.
Controlling Icon Image Size with Window Manager Resources.

Resource	Description
`iconImageMaximum`	Maximum size of an icon image.
`iconImageMinimum`	Minimum size of an icon image.

Using the Icon Box

The icon box is a scrollable window containing an icon for every client window in the current workspace that can be iconified. The icons of normalized windows are visually distinct from icons of minimized windows.

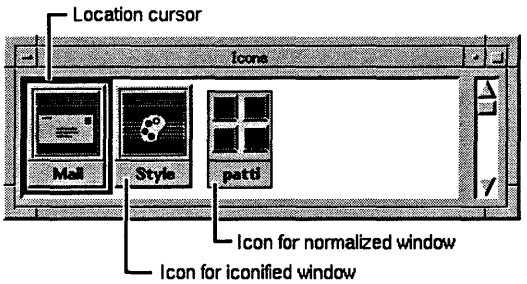


Figure 11-4. A Horizontal Icon Box.

Specifying Use of the Icon Box. There are two ways to specify the use of an icon box:

- Set the `useIconBox` resource is set to "True."
- Edit the `FrontPanel` section in `$HOME/.vue/vuewmrc`:

  1. Uncomment the line for the **`vuewmbox`** control.
  2. Comment the second **`vuewmrib`** control.

  The lines in `vuewmrc` will look like this:

```
# vuewmrib        [N] =65
vuewmbox          [P] @iconbox.1.bm =65
```

11

Orientation and Size of the Icon Box. By default, the icon box is oriented vertically. Several resources specify the geometry and general appearance of the icon box.

Table 11-14. Icon Box Appearance Resources.

Resource	Description
`iconBoxGeometry`	Size and placement of the icon box.
`iconBoxSBDisplayPolicy`	Whether the icon box has only horizontal or vertical scrollbars, or both.
`iconBoxTitle`	Title appearing in the titlebar.

The value of the `iconBoxGeometry` resource is a standard window geometry string with the syntax:

$$= Width \times Height [\pm x \pm y]$$

where *width* and *height* are in units of icons. The actual size of the icon box window depends on the `iconImageMaximum` (size) and `iconDecoration` resources.

Table 11-15. Values for the iconBoxSBDisplayPolicy Resource.

Value	Effect
`vertical`	Displays vertical scrollbar and sets the icon box orientation to horizontal.
`horizontal`	Displays horizontal scrollbar and sets the icon box orientation to vertical.
`all`	Displays horizontal and vertical scrollbars. The largest dimension of the `iconBoxGeometry` resource determines whether the icons are placed in a row or a column.

Appearance and Behavior of Icons in the Icon Box. Two resources determine the appearance of icons in the icon box:

Table 11-16. Icon Behavior in the Icon Box.

Resource	Description
`iconDecoration`	Determines icon components—label, image, or both (see "Specifying the Icon Decoration" in this chapter).
`fadeNormalIcon`	Icons for normal windows are grayed-out. The default is "False."

Minimizing the Icon Box. There are two ways to temporarily remove the icon box from the display:

- If the workspace manager contains the **vuewmbox** control, use the system menu button to "close" the icon box. Click the **vuewmbox** control to redisplay it.
- If there is no **vuewmbox** control in the workspace manager, the icon box will contain a minimize button to iconify the icon box.

Other Icon Box Resources. The `iconBoxName` resource specifies the name that is used to look up client-specific icon box resources. The default name is "iconbox."

The syntax for icon box resources is:

```
Vuewm*iconboxname*resource: value
```

You can use **vuewm** resources dealing with frame decoration and icon appearance. The icon appearance resources affect the icon displayed when the icon box is minimized.

For example, the following line specifies that a bitmap file named **icons** be used when the icon box is iconified.

```
Vuewm*iconbox*iconImage: $HOME/bitmaps/icons
```

11 Using Custom Pixmaps For Icons

When you iconify a client, either the client supplies its own icon image, the window manager supplies a default image, or you supply an image of your own—one you obtain "ready-made" or one you create using the 'bitmap' client.

There are two resources that tell the window manager where custom icons are located:

- The `iconImage` resource specifies the bitmap for a particular icon image. Its value is the path to the file containing the bitmap. Note that, if specified, this resource overrides any client-specified image.
- The `bitmapDirectory` resource causes the window manager to search the specified directory whenever a bitmap is named with no complete path. The default value for `bitmapDirectory` is `/usr/include/X11/bitmaps`.

The syntax for specifying these resources is:

`Vuewm*bitmapDirectory:` *directory*

`Vuewm*` [{ *clientclass* / *clientname* / `defaults` }] `*iconImage:` *path/bitmap*

For example, the following lines specify a custom bitmap for all `emacs` client windows, and two different bitmaps for two `xterm` windows named `term1` and `term2`. All the custom files are in directory `$HOME/bitmaps`.

```
Vuewm*bitmapDirectory: $HOME/bitmaps
Vuewm*emacs*iconImage: quill
Vuewm*term1*iconImage: star
Vuewm*term2*iconImage: square
```

Workspace Backdrops

This section covers:

- How the window manager obtains backdrop images (bitmaps).
- How to change backdrop colors from their default values.

Backdrop Images

The workspace backdrops shipped with HP VUE are a collection of bitmap files located in the directory `/usr/lib/X11/bitmaps/Vuebackdrops`. Files in this directory are listed in the style manager's Backdrop dialog box.

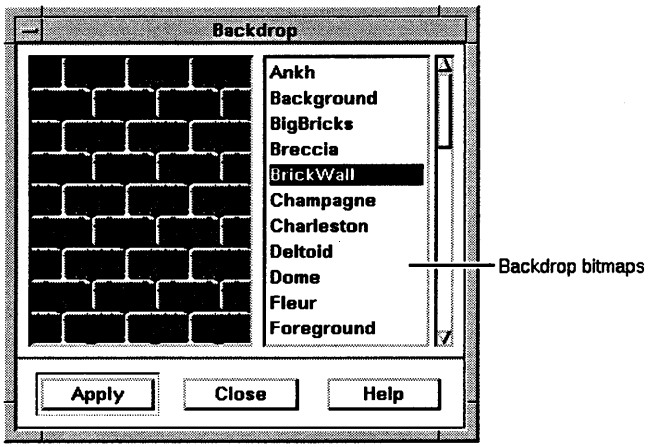


Figure 11-5. The Backdrop Dialog Box.

Changing the Backdrop Directory

Use the style manager resource `backdropDirectory` to change the directory the style manager uses to locate the backdrop files. Use the syntax:

`Vuestyle*backdropDirectory:` *path/directory*

The value of the resource must be an absolute path.

Associating Backdrops With Workspaces

The following resource associates a backdrop with a workspace:

```
Vuewm[*screen]*workspace*backdrop*image: @bitmap_file
```

where *bitmap_file* is the absolute path of the bitmap.

For example, the following line specifies the backdrop for a workspace named "Mail":

```
Vuewm*Mail*backdrop*image:    \
   @/usr/lib/X11/bitmaps/Vuebackdrops/Squids
```

Displaying the Root Window in a Workspace

The workspace backdrops overlay the root window. To turn off the backdrop in a particular workspace, use the **image** resource as follows:

```
Vuewm[*screen]*workspace*backdrop*image: none
```

Adding Custom Backdrops

If you have root permission, you can add a backdrop by simply adding a bitmap file to **/usr/lib/X11/bitmaps/Vuebackdrops**.

The following procedure allows users who lack permission to write to the default backdrop directory to create and use their own bitmaps:

1. Create a directory for the backdrops bitmaps.
2. Use the **backdropDirectory** resource to specify this directory. For example, if the directory is **$HOME/.vue/backdrops**, then add:

   ```
   Vuestyle*backdropDirectory:  $HOME/.vue/backdrops
   ```

   to the resource database.
3. Create a symbolic link between the new directory and the backdrops shipped with HP VUE by executing:

   ```
   ln -f -s /usr/lib/X11/bitmaps/Vuebackdrops/* $HOME/.vue/backdrops
   ```

   This links all the files in the default directory with the new directory.
4. Create the bitmap files for the new backdrops. Put them in the new directory.

The new backdrops appear in the style manager Backdrop dialog box at the next session.

Combining a Bitmap Image With the "Deep" Backdrop

The "deep" backdrops are the backdrops that appear as stacked rectangles with different gradations of the same color.

You can create a deep backdrop containing a bitmap image in the center by using the naming convention `deep`*bitmap_file* for the bitmap file name. For example, if you've added a bitmap named `Yogurt` to the backdrop directory, the resource:

```
Vuewm*Printing*backdrop*image:  @deepYogurt
```

creates a deep backdrop for the workspace named `Printing` containing the `Yogurt` bitmap in the center.

Backdrop Colors

This section describes:

- How HP VUE generates the default backdrop colors.
- How to change the color sets used for the backdrops.
- How to specify specific colors for backdrops.

Default Colors

The colors of the backdrops are obtained from the color sets in the current palette:

- For high color displays, the workspaces are assigned color sets in reverse order—that is, workspace #1 uses color set 8, workspace #2 uses color set 7, etc. The active and inactive color sets (by default, color sets 1 and 2) are not used as backdrop colors (a seventh workspace reuses color set 8).
- For medium color displays, all backdrops use the secondary color set (by default, color set 4).
- For low-color and monochrome displays, all backdrops use the inactive color set (by default, color set 2.

Two colors are used from each color set—the `topShadowColor` is used for the image background and the `bottomShadowColor` is used for the image foreground.

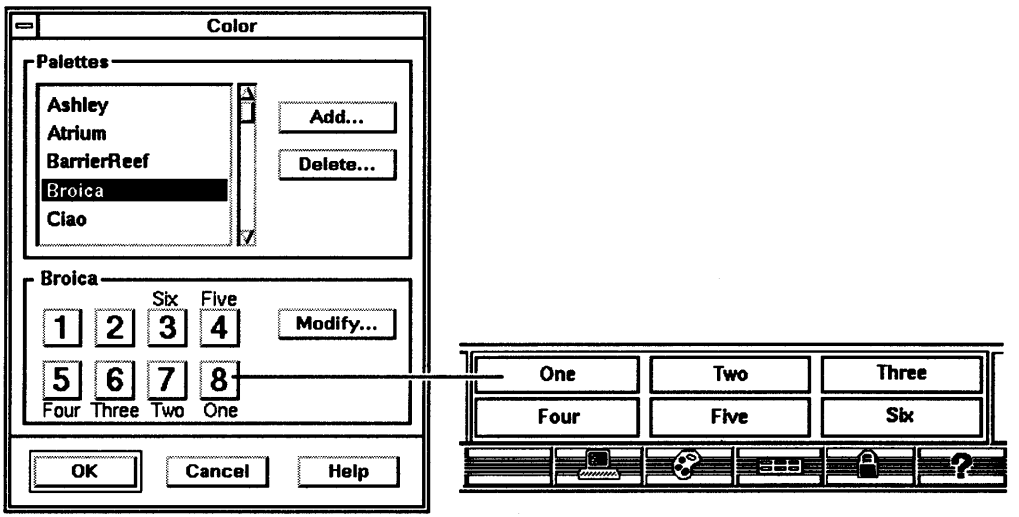


Figure 11-6. Color Sets Used for Backdrops on a High Color Display.

Changing the Color Sets Used for Backdrops

The `colorsetId` resource specifies which color set is used for a particular workspace. It has the syntax:

`Vuewm[`*`*screen`*`]*`*`workspace`*`*backdrop*colorsetId:` *color_set*

For example, the following line causes the workspace named "Graphics" to use color set 1.

```
Vuewm*Graphics*backdrop*colorsetId:  1
```

Explicitly Setting the Backdrop Colors

You can use resources to explicitly specify the foreground and background color of backdrops. Colors specified this way cannot be changed dynamically.

`Vuewm[`*`*screen`*`]*`*`workspace`*`*backdrop*`*`resource`*`:` *color*

Table 11-17. Backdrop Color Resources.

Resource	Description
`imageBackground`	Sets the backdrop background color.
`imageForeground`	Sets the backdrop foreground color.

For example, the following lines create a violet and black backdrop for the workspace named "Printing", regardless of the current palette.

```
Vuewm*Printing*backdrop*imageBackground:   violet
Vuewm*Printing*backdrop*imageForeground:   black
```

Customizing the Root Window with 'xsetroot'

In HP VUE, the root window is the window behind the workspace backdrops. Ordinarily, the root window is completely obscured by the backdrops. However, the **Vuewm*backdrop*image** resource allows you to specify that no backdrop be used for one or more workspaces, thus allowing the root window to show through.

For example, the following resource specifies no backdrops for the workspace named **Testing**.

```
Vuewm*Testing*backdrop*image:  none
```

Changing the Workspace Cursor

The workspace cursor appears when the pointer is elsewhere than a client window. The **xsetroot** client lets you change the bitmap used for the root window cursor. For example, the following command changes the workspace cursor using two custom bitmaps located in directory **$HOME/bits**.

```
xsetroot -cursor ~/bits/shuttle.bm ~/bits/mask.bm
```

11 Window Manager Functions

The window manager functions can be accessed with button bindings, key bindings, or by window manager menus.

Functions may be constrained by:

- The contexts in which the function can operate:

Root	Root window is selected.
Icon	Icon is selected.
Window	Client window is selected. Some functions operate only when the window is in its normalized state (f.maximize), or its maximized or iconified state (f.normalize).

- Which devices (resources) can specify and invoke the function:

Button	Button binding (mouse).
Key	Key binding.
Menu	window manager menu.

Any selection that uses an invalid context, an invalid function, or a function that doesn't apply to the current context is grayed out. This is the case with the "Restore" selection of a terminal window's window menu or the "Minimize" selection of an icon's window menu.

Table 11-18. Window Manager Functions.

Functions		Contexts			Devices		
Name	**Description**	**Root**	**Icon**	**Window**	**Button**	**Key**	**Menu**
Name	Description	Root	Icon	Window	Button	Key	Menu
f.action	Invokes the specified action.	√	√	√	√	√	√
f.beep	Sounds a tone.	√	√	√	√	√	√
f.circle_down	Puts window on bottom of stack.	√	√	√	√	√	√
f.circle_up	Puts window on top of stack.	√	√	√	√	√	√
f.exec	Uses **/bin/sh** to execute a command.	√	√	√	√	√	√
f.focus_color	Sets colormap focus when colormap focus policy is explicit.	√	√	√	√	√	√
f.focus_key	Sets keyboard input focus when keyboard focus policy is explicit.	√	√	√	√	√	√
f.kill	Terminates a client's connection to server.		√	√	√	√	√
f.lower [*−client*]	Lowers a window to bottom of stack.		√	√	√	√	√
f.maximize	Enlarges a window to its maximum size.		√	√	√	√	√
f.menu [*menuname*]	Associates a menu with a selection or binding.	√	√	√	√	√	√
f.minimize	Changes a window into an icon.			√	√	√	√
f.move	Enables the interactive moving of a window.		√	√	√	√	√

Table 6-17a. Valid Window Manager Functions (continued).

Functions		Contexts			Devices		
Name	**Description**	**Root**	**Icon**	**Window**	**Button**	**Key**	**Menu**
f.next_cmap	Installs the next colormap in the window with the colormap focus.	√	√	√	√	√	√
f.next_key [icon | window | transient]	Sets keyboard focus policy to the next window/icon in the stack.	√	√	√	√	√	√
f.next_work-space	Switches to the next workspace.	√	√	√	√	√	√
f.nop	Does no function.	√	√	√	√	√	√
f.normalize	Normalizes a window.		√	√	√	√	√
f.normalize_ and_raise	Normalizes and raises a window (if `raiseKeyFocus` is "True").		√	√	√	√	√
f.occupy_all	Makes a window persistent across all workspaces.		√	√	√	√	√
f.pack_icons	Packs icons rows in the workspace window or icon box.	√	√	√	√	√	√
f.pass_keys	Toggles between enabling and disabling processing of key bindings.	√	√	√	√	√	√
f.post_wmenu	Posts the window menu	√	√	√	√	√	

Table 6-17b. Valid Window Manager Functions (continued).

Functions		Contexts			Devices		
Name	**Description**	**Root**	**Icon**	**Window**	**Button**	**Key**	**Menu**
f.prev_cmap	Installs the previous color map in the window with the colormap focus.	√	√	√	√	√	√
f.prev_key [icon | window | transient]	Sets the keyboard input focus to the next window/icon in the stack.	√	√	√	√	√	√
f.previous_workspace	Switches to the previous workspace.	√	√	√	√	√	√
f.quit_mwm	Terminates the window manager without exiting the session.	√			√	√	√
f.raise [*–client*]	Lifts a window to the top of the window stack.		√	√	√	√	√
f.raise_lower	Raises a partially concealed window; lowers an unconcealed window.		√	√	√	√	√
f.refresh	Redraws all windows.	√	√	√	√	√	√
f.refresh_win	Redraws a client window.			√	√	√	√
f.remove	Removes client window from current workspace		√	√	√	√	√
f.resize	Enables you to interactively resize a window.			√	√	√	√

11

Table 6-17c. Valid Window Manager Functions (continued).

Functions		Contexts			Devices		
Name	**Description**	**Root**	**Icon**	**Window**	**Button**	**Key**	**Menu**
f.restart	Restarts the window manager.	√			√	√	√
f.send_msg *messagenumber*	Sends a client message.		√	√	√	√	√
f.separator	Draws a line between menu selections.	√	√	√			√
f.set_behavior	Restarts **vuewm** with CXI or custom behavior.	√	√	√	√	√	√
f.title	Inserts a title into a menu at the specified position.	√	√	√			√
f.toggle_frontpanel	Minimizes the front panel.	√	√	√	√	√	√
f.workspace_presence	Displays the Workspace Presence dialog box.			√	√	√	√
f.workspace_rename	Displays the Rename Workspace dialog box.	√	√	√	√	√	√

Window Manager Menus

The window manager menus are defined in the window manager configuration file (typically, `$HOME/.vue/vuewmrc` or `/usr/lib/X11/vue/Vuewm/sys.vuewmrc`).

Menu Syntax

Window manager menus have the syntax:

```
Menu MenuName
{
    selection1 [mnemonic]  [accelerator]  function [argument]
    selection2 [mnemonic]  [accelerator]  function [argument]
    selection3 [mnemonic]  [accelerator]  function [argument]
                                  ⋮
    selection* [mnemonic]  [accelerator]  function [argument]
}
```

The order of the selections is the order of their appearance in the menu. Any character string containing a space must be enclosed in double quotes.

A selection name can be a character string or a bitmap. The syntax for using a bitmap is:

`@`*/path/bitmapfile function* [*argument*]

A mnemonic is specified using the syntax:

`mnemonic =` `_`*character*

An accelerator is specified using keyboard binding syntax described later in this chapter (see "Keyboard Binding Syntax").

11 Default Menus

The window manager has two default menus:

- The workspace menu. The HP VUE default button bindings display it by pressing buttons 1 or 3 on the workspace window. This menu is also called the "root" menu.
- The window menu. The default window menu is built into **vuewm**. For reference, a copy of its contents are placed in **sys.vuewmrc**. A built-in button binding displays it by pressing button 1 on a frame's window menu button. An additional default HP VUE button binding displays it by pressing button 3 anywhere on a window frame.

You can modify either menu. However, for consistent window operation, it's usually better to modify only the workspace menu.

Changing the Menu Associated with the Window Menu Button

The **windowMenu** resource specifies the menu displayed when you press button 1 on the window menu button. It has the syntax:

Vuewm* [{ *clientname* | *clientclass* | **defaults** }] ***windowMenu:** *MenuName*

For example, the following line associates a menu named **EditMenu** with an **xterm** window started as **xterm -name editor**.

```
Vuewm*editor*windowMenu: EditMenu
```

Changing the Workspace Menu

The default workspace (root) menu is bound to mouse buttons 1 and 3 by the button bindings:

```
<Btn1Down>          root        f.menu VueRootMenu
<Btn3Down>          root        f.menu VueRootMenu
```

To use a different menu, replace **VueRootMenu** with the name of another menu in the **vuewmrc** file.

Making Other New Menus

You can create completely new menus that are displayed by:

- Key presses. You must provide a key binding for the menu.
- Button presses. You must provide a button binding for the menu.
- Cascading the menu from another menu. For example, the following workspace menu entry, when selected, displays the menu named `Remote_Hosts`.

```
"Remote Hosts"   f.menu Remote_Hosts
```

Button Bindings

A **button binding** associates a button operation with a window manager function. Button bindings persist across all workspaces.

The window manager recognizes these button operations:

Press	Holding down a mouse button.
Click	Pressing and releasing a mouse button.
Double-click	Pressing and releasing a mouse button twice in rapid succession.
Drag	Pressing a mouse button and moving the pointer (and mouse device).

You should not bind the same button to different functions for the click and press operations. For example, you should not bind `<Btn1Click>` and `<Btn1Down>` to different functions.

11 Button Binding Syntax

The syntax for button bindings is:

```
Buttons ButtonBindingSetName
{
    button   context|context   function   [argument]
    button   context|context   function   [argument]
    button   context|context   function   [argument]
}
```

The context indicates where the pointer must be for the binding to be effective. The following are valid contexts:

`root`	Workspace window.
`window`	Client window or window frame.
`frame`	Window frame.
`icon`	Icon.
`title`	Title bar.
`app`	Client window (excludes the window frames).

You should avoid binding more than one function that changes window or window manager state, or that posts a dialog box or menu, to the same button and context.

Default Button Bindings

The window manager has the following built-in button bindings.

Table 11-19. Built-In Button Bindings.

Location of Pointer	Behavior
Window menu button	Pressing button 1 displays the window menu. This behavior can be modified by the **wMenuButtonClick** resource.
Window menu button	Double-clicking button 1 closes the window. This behavior can be modified by the **wMenuButtonClick2** resource.
Minimize button	Clicking button 1 minimizes the window.
Maximize button	Clicking button 1 maximizes the window.
Title bar	Dragging button 1 moves the window.
Window or icon	Pressing button 1 gives it keyboard focus.
Resize border	Dragging button 1 resizes the window.
Icon	Clicking button 1 displays the icon window menu. This behavior can be modified by the **iconClick** resource.
Icon	Double-clicking button 1 normalizes the window.
Icon	Dragging button 1 moves the icon.

These bindings are fixed—they cannot be deleted or replaced by other bindings. However, you can add to some of them (see "Modifying Button Bindings and Their Functions.") For example, you can specify an additional function for double-clicking button 1 in an icon, but the double click will also normalize the window.

11

The **sys.vuewmrc** file contains these additional HP VUE bindings that can be deleted or replaced. They are located in **vuewmrc** in the button binding set named **VueButtonBindings**.

Table 11-20. Additional HP VUE Button Bindings.

Button Binding	Description
`<Btn1Down> root f.menu VueRootMenu`	Pressing button 1 on the workspace window displays the workspace (root) menu.
`<Btn3Down> root f.menu VueRootMenu`	Pressing button 3 in the workspace window displays the workspace (root) menu.
`<Btn1Down> frame|icon f.raise`	Pressing button 1 on a frame or icon raises it to the top of the stack.
`<Btn3Down> frame|icon f.post_wmenu`	Pressing button 3 on a frame or icon displays the window menu.
`Meta<Btn1Down> icon|window f.move`	Pressing Alt (Extend char) and dragging button 1 on an icon or window moves the window.
`Meta<Btn3Down> window f.minimize`	Pressing Alt (Extend char) and button 3 on a window minimizes the window.

Modifying Button Bindings

Button bindings can be modified by:

- Editing the `VueButtonBindings` in `vuewmrc`.
- Making a new button binding set.

To create a new button binding set:

1. Edit `vuewmrc` to include a new key binding set with a unique *ButtonBindingSetName*.
2. Set the `buttonBindings` resource to the new *ButtonBindingSetName*.

For example, here is a new button binding set:

```
Buttons GraphicsButtonBindings
{
 <Btn2Down>  root  f.menu  "Graphics Project"
}
```

The following resource makes the new set active.

```
Vuewm*buttonBindings: GraphicsButtonBindings
```

Key Bindings

You can bind window manager functions to "special" keys using **keyboard bindings**. The window manager recognizes the following special keys:

- Shift.
- Escape.
- Alt (Meta or Extend Char).
- Tab.
- Ctrl.
- Lock.

11 Keyboard Binding Syntax

The syntax for keyboard bindings is:

```
Keys KeyBindingSetName
{
    key   context|context   function    [argument]
    key   context|context   function    [argument]
    key   context|context   function    [argument]
}
```

The first line specifies the function type (**Keys**) and the name of the keyboard binding set.

The remaining lines specify the actual keyboard bindings. Each line identifies a unique key press sequence. The context indicates what element(s) must have keyboard focus for the binding to be active:

root	Workspace window.
window	Client window.
icon	Icon.

Default Key Bindings

The window manager has the following HP VUE default key bindings.

Table 11-21. HP VUE Window Manager Keyboard Bindings.

When the keyboard focus is:	Press:	Effect:
Icon, window, or none	Alt Menu[1]	Toggles the workspace manager between iconified and restored state.
Window or icon	Shift Escape	Displays window menu.
Window or icon	Alt space	Displays window menu.
Window, icon, or none	Alt Tab	Switches keyboard focus to the next window or icon.
Window, icon, or none	Alt Shift Tab	Switches keyboard focus to the previous window or icon.
Window, icon, or none	Alt Escape	Switches keyboard focus to the next window or icon.
Window, icon, or none	Alt Shift Escape	Switches keyboard focus to the previous window or icon.
Icon, window, or none	Alt ↓	Puts the top window on the bottom of the stack.
Icon, window, or none	Alt ↑	Puts the bottom window on the top of the stack.
Window	Alt F6	Switches keyboard focus to the next transient window in an application.
Window, icon, or none	Alt Ctrl Shift !	Restarts **vuewm** with default HP/OSF Motif or custom behavior.

1 For the HP C1429A Enhanced Vectra keyboard, use Alt Print Screen. For keyboards with a Pop key, use Alt Pop.

11

The HP VUE default bindings are specified in **sys.vuewmrc**. To see them, search the file for this portion of the file:

```
Keys VueKeyBindings
{
    list of key bindings
}
```

Modifying Key Bindings

Key bindings can be modified by:

- Editing the **VueKeyBindings** in **vuewmrc**.
- Making a new key binding set.

To create a new key binding set:

1. Edit **vuewmrc** to include a new key binding set with a unique *KeyBindingSetName*.
2. Set the **keyBindings** resource to the new *KeyBindingSetName*.

New key bindings take effect when the window manager is restarted.

Controlling Window Size and Placement

The window manager has several resources that allow you to refine your control of the size and placement of windows.

Table 11-22. Refining Control with Window Manager Resources.

Resource	Description	Default
`interactivePlacement`	Lets the user position new windows interactively.	False
`limitResize`	Lets the user enlarge windows beyond the `maximumClientSize`.	False
`maximumMaximumSize`	Maximum size of a client window set by user or client.	2xscreen
`moveThreshold`	Number of pixels the pointer must move before a move operation is started.	4 pixels
`positionIsFrame`	Position information refers to the window frame rather than the window.	True
`positionOnScreen`	When "True," positions new windows so that they are not clipped by the screen edges.	True
`resizeBorderWidth`	Width of the frame's resize border.	10 pixels
`resizeCursors`	Whether or not a resize cursor is displayed in a resize border.	True
`maximumClientSize`	Maximum size of a maximized client.	screen size

All the resources that refine control over the size and placement of windows, except **maximumClientSize**, have the following syntax:

Vuewm*resource*: *value*

The **maximumClientSize** resource has the syntax:

Vuewm* { *clientname* / *clientclass* / **defaults** } **.maximumClientSize:** *width*x*height*

Setting Focus Policies

The window manager focus policies control the arbitration of resources among clients by determining what happens when a window becomes the active window.

Table 11-23. Focus Policy Resources.

Resource	Description	Default Value
colormapFocusPolicy	Specifies which window has the colormap focus.	keyboard
keyboardFocusPolicy	Specifies which window has the keyboard and mouse focus.	explicit

These values are valid for the **colormapFocusPolicy** resource:

keyboard The window manager tracks keyboard input and installs a client's colormap when the client window gets the keyboard input focus.

pointer The window manager tracks the pointer and installs a client's colormap when the pointer moves into the client window or the window frame around the client.

explicit The window manager tracks a specific focus-selection operation and installs a client's color map when the focus-selection operation is done in the client window.

These values are valid for the **keyboardFocusPolicy** resource:

pointer The window manager tracks the pointer and sets the keyboard focus to a client window when the pointer moves into that window or the window frame.

explicit The window manager tracks a specific focus-selection operation and sets the keyboard focus to a client window when the focus-selection operation is done in that client window.

For example, the following line changes the keyboard focus policy:

```
Vuewm*keyboardFocusPolicy: pointer
```

When the keyboard focus policy is explicit, the **passSelectButton** resource specifies the consequence of the focus-selection operation. When **passSelectButton** is "True" (the default value), the focus-selection operation is passed to the client or is used by the window manager to perform some action; when **passSelectButton** is "False," the focus-selection operation is used only to select the focus and is not passed.

11 Switching Between Default and Custom Behavior

The window manager has a built-in key binding that allows you to switch back and forth between customized **vuewm** behavior and default OSF/Motif behavior. The key presses for doing this are Alt Shift Ctrl !.

Switching to the default behavior has these effects:

- The workspace manager is removed.
- Custom key and button bindings are removed.

Using the Window Manager with Multiple Screens

By default, the window manager manages one screen. Managing multiple screens can be specified in two ways:

- Using resources.
- By editing the startup command for **vuewm**.

When **vuewm** manages more than one screen, it tries to create at least one workspace on each screen.

Using Resources to Manage Multiple Screens

The following resources configure the window manager to manage multiple screens:

- To specify that **vuewm** manage multiple screens, use the resource:

  ```
  Vuewm*multiScreen:  True
  ```

  This tells the window manager to try to manage all screens that the server manages.

- Use the **vuewm** resource **screenList** to define the screen names. For example, the following resource names two screens **zero** and **one**.

  ```
  Vuewm*screenList:  zero one
  ```

Specifying Multiple Screens from the Command Line

The the session manager resource **wmStartupcommand** lets you specify a different command line for starting the window manager. You can use command-line options to start **vuewm** so that it manages multiple screens:

- The **-display** option specifies the display. It has the syntax:

  -display *hostname*:*display*.*screen*

  The workspace manager (front panel) will be displayed on the screen specified by the *screen* parameter.

- The **-multiscreen** option causes **vuewm** to manage all the screens on the specified display.

- The **-screens** option specifies the screen names used to obtain screen-specific resources.

For example, the resource:

```
Vuesession*wmStartupcommand: \
   vuewm -display local:0.1 -multiscreen -screens zero one
```

causes **vuewm** to manager all the screens on display 0. Screens 0 and 1 are named **zero** and **one**. The workspace manager is displayed on screen 1 (named **one**).

Workspaces in Multiple Screens

When you manage multiple screens with **vuewm**, the workspace manager is displayed on only one screen. However, the other screens can have multiple workspaces. In order to switch workspaces on those screens, you must either add **f.prev_workspace** and **f.next_workspace** functions to the workspace menu, or create button or key bindings for them.

11 Clients That Help You Manage Windows

Three additional clients are directly related to window management:

- `resize`
- `xrefresh`
- `xwininfo`

Resetting Environment Variables with 'resize'

The `resize` client resets three environment variables: `TERM`, `LINES`, and `COLUMNS`. This enables a shell to reflect the current size of its window. Resetting the environment variables enables non-client programs to adjust their output to the window's new size. Use `resize` whenever you resize a terminal emulator window and want a non-client program running in that window to reflect the window's new size.

The `resize` client is typically used as an argument to the HP-UX `eval` command.

For example, after you have resized a window, reset the `LINES` , and `COLUMN` environment variables to reflect the new window size by executing:

```
eval 'resize'
```

Repainting the Screen with 'refresh'

The `xrefresh` client "repaints" the screen or a specified portion of the screen. This removes the "graphics litter" that occasionally disfigures a screen.

The `xrefresh` client performs a similar task to the `f.refresh` window manager function. However, the `xrefresh` client, because of its options, is more versatile.

For example, to repaint the upper left quarter of the screen, execute:

```
xrefresh -geometry 800x400+1+1
```

Getting Window Information with 'xwininfo'

The **xwininfo** client is a utility program that displays useful information about a window. The window is specified by selecting it (clicking mouse button 1) after executing the command to start **xwininfo**.

For example, executing:

```
xwininfo -stats
```

displays this window information:

```
xwininfo ==> Window id: 0x5001e0 (has no name)

         ==> Upper left X: 33
         ==> Upper left Y: 13
         ==> Width: 824
         ==> Height: 846
         ==> Depth: 8
         ==> Border width: 0
         ==> Window class: InputOutput
         ==> Colormap: 0x80065
         ==> Window Bit Gravity State: ForgetGravity
         ==> Window Window Gravity State: NorthWestGravity
         ==> Window Backing Store State: NotUseful
         ==> Window Save Under State: no
         ==> Window Map State: IsViewable
         ==> Window Override Redirect State: no
         ==> Corners:  +33+13  -423+13  -423-165  +33-165
         -geometry =824x846+33+13
```

11 Using Other Window Managers

The session manager resource **wmStartupCommand** allows you to use a different window manager during the session. For example, the entry:

```
Vuesession*wmStartupcommand:  twm
```

causes the session manager to start the **twm** window manager rather than **vuewm**.

12

Filetypes and Actions

The HP VUE File Manager can manipulate:

- Directories. These are standard directories and are represented by a folder icon.
- Files. These are standard files. However, HP VUE provides the ability to classify files into different types. Each file type can be assigned its own icon and behavior.
- Actions. Actions are unique to HP VUE. They allow operating system commands and applications to be represented, manipulated, and invoked graphically.

File types and actions are defined by the HP VUE file management database. It provides the ability to:

- Define different file types and the recognition rules that let the system distinguish one file type from another. For example, the system can define the file types MEMO and BM (bitmap), and be configured to recognize files with extensions `.memo` and `.bm` as files of those types.
- Associate icons with file types and actions for visual recognition. The icons are used by the file manager to provide a graphical representation for directory listings.
- Define actions that operate on files. For example, you can define an action entitled EDIT for editing operations. The behavior of an action can depend on the file type. For example, editing a MEMO file can invoke a text editor, while editing a BM file can invoke a bitmap editor.
- Associate icons with actions. For example, the EDIT action can be represented by an icon. The action can then be invoked by dragging a file icon onto the action icon or by selecting the action from the Actions menu.

- Associate default actions with file types. The default action is the action that occurs when the user double-clicks a file in the file manager.

12 Structure of the File Management Database

This section covers:

- The naming convention for files in the file management database.
- The locations of database files, and the resources that control where the system looks for the files.
- How the database is created from system-wide, local, and user-specific portions. This allows you to add local and personal customizations to the database.

Naming Conventions

The file management database has two general parts:

- The **file type database**. This database consists of one or more files that describe the file types recognized by the system. Files in this database must follow the naming convention *filename*`.ft`.
- The **action definition database**. This database consists of one or more files that define the actions the system recognizes. Files in this database must follow the naming convention *filename*`.ad`.

Location of Database Files

When an HP VUE session is started, the file management database is loaded by searching multiple directories for database files. There are several default directories; these defaults can be modified by adding, removing, or substituting directories.

This section covers:

- The default search path for database files.
- How to modify the search path.

Default Directories

By default, database files are located in three directories:

- **`$HOME/.vue/types`**. Files in this directory affect only the user. If you are an end user and intend to customize your database, you should do so only in this directory.
- **`/usr/local/lib/X11/vue/types`**. Use this directory for database files that will be available to all users on the system. If this directory does not exist on your system, you can create it (this requires superuser permission). Once the directory is created, it will be searched the next time a user logs in. In general, this is the directory system administrators modify.
- **`/usr/lib/X11/vue/types`**. This directory contains the default database files shipped with HP VUE. Its files create the system-wide file type and action definition databases. You should not modify the files in this directory. If you want to add or modify a file type or action, you should do so in one of the other database directories.

Because the file management database is searched in the order listed above, personal customizations override system customizations, and system customizations override the default definitions.

Caution

The update process for HP VUE removes the current files in **`/usr/lib/X11/vue/types`** and replaces them with new ones. If you have modified those files and want to preserve the changes, move your modifications to **`/usr/local/lib/X11/vue/types`** before updating.

The Default File Type and Action Definition Database

The following figure shows the directory structure of the system-wide database shipped with HP VUE.

12

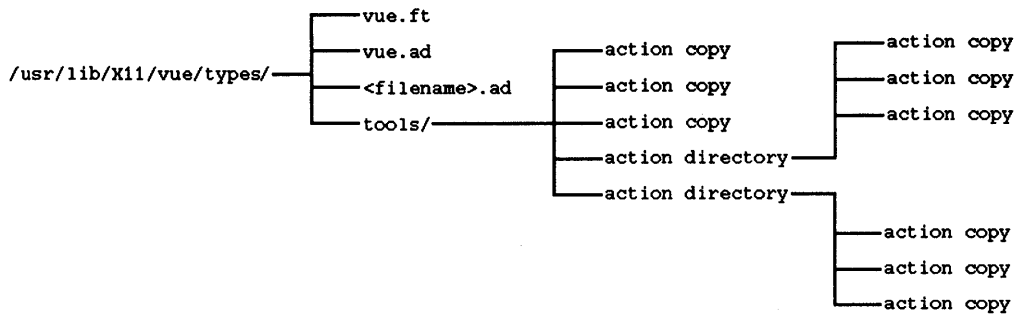


Figure 12-1. The System-Wide Default Database.

Note these aspects of its organization:

- The file type definitions are contained in files with the naming convention *filename*`.ft`.
- Action definitions are contained in files with the naming convention *filename*`.ad`.
- The `types` directory contains a `tools` subdirectory. The `tools` directory and its subdirectories contain files representing actions defined in the action definition files. For example, `/usr/lib/X11/vue/types/vue.ad` contains definitions for actions `VI` and `RES_EDIT`. The action files associated with these definitions are `/usr/lib/X11/vue/types/tools/edit/VI` and `/usr/lib/X11/vue/types/tools/X11_apps/RES_EDIT`. These action files create the icons for the actions in the file manager applications directory. (See "Organization of the Applications Directory and Action Database" later in this chapter for more information.)

Some actions do not have corresponding files in the `tools` directory, and therefore do not have action icons in the applications directory. For example, the EXIT_SESSION action defined in `/usr/lib/X11/vue/types/vue.ad` is used by the workspace manager's logout button. This action does not appear in the applications directory.

Creating Local and Personal Databases

To construct a local or personal database:

1. Create one or more *filename*`.ft` or *filename*`.ad` files in the appropriate database directory (see "Default Directories," above).
2. Edit these files to include the file types and actions you want:
   - You can create an entirely new file type or action.
   - You can modify an existing action by copying it from the system-wide database and modifying it.
3. If you've added action definitions, you can create action files in the `tools` directory, or in a subdirectory you've created under `tools`.

The following illustration shows the structure of a personal database created by user **anna**. It contains a file for file type definitions, two files for action definitions, and three subdirectories of `tools` containing the actions defined in the two action definition files.

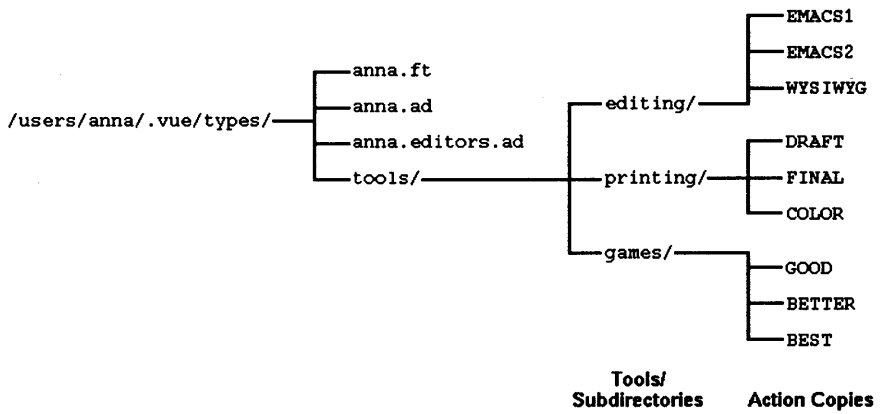


Figure 12-2. A Personal File Type and Action Definition Database.

Adding Directories to the Database Search Path

The resource `Vue.TypesDirs` identifies the directories containing database files. The value of the resource is a list of directories separated by commas.

The syntax for specifying each directory is:

12

[*hostname*:][*path/*]*directory*@*symbolic_name*

If the *hostname* is omitted, the system assumes that the directory is located on the local system. If you specify a relative path, it is assumed to be relative to the user's home directory.

Clicking the applications directory button in the workspace manager displays a file manager view of the applications directory. Each top-level folder in this window corresponds to one of the directories in the `Vue.TypesDirs` search path. The folders are labeled with their *symbolic_name*. If a directory on the search path does not exist, or does not have a `tools` subdirectory, it does not have a folder in the application directory window.

The default value for the `Vue.TypesDir` resource is:

```
Vue.TypesDirs:  .vue/types@personal-apps,\
/usr/local/lib/X11/vue/types@local-apps,\
/usr/lib/X11/vue/types@system-apps
```

The directories are separated by commas; there can be no embedded spaces.

There are two ways to change the database search path:

- For system-wide changes, edit the value of the `Vue.TypesDir` resource in `/usr/lib/X11/app-defaults/Vue`.
- To make changes for a single user, load the new resource value into the resource database (using the RES_EDIT action or `xrdb`). The user must then log out and login in. (For more information on loading resources, see "Adding and Changing Resources with xrdb" in chapter 8.

General Syntax Rules for Database Files

The following general syntax rules apply to both the file types and action definition databases:

- The database files are standard text files.
- There is one entry per line. Each entry consists of a number of fields.
- The backslash character (\) is used as an escape character, removing the special meaning of the character following it. When a backslash character is the last character on a line (a new-line character immediately follows the backslash; there can be no other characters between the backslash and the new-line character), the new-line character is not interpreted as the end of the entry. This allows you to put a single entry on more than one line.
- Fields are separated from one another by spaces or tabs. There can be no embedded spaces within a field, except for last field of each entry.
- Blank lines are ignored.
- If the first non-blank character in a line is the pound sign (#), the entire line is interpreted as a comment line.
- A commented line should never immediately follow a line ending with a backslash.
- You can use string variables in database files (see "Using String Variables in Database Files" later in this chapter).

The File Types Database

The file types database describes the file types known to the system and the criteria the system uses to differentiate one file type from another.

Default File Types

The default file types are defined in `/usr/lib/X11/vue/types/vue.ft`.

Table 12-1. Default File Types

File Type	Naming Convention	Description
BM	`*.bm`	Bitmaps
BROKEN_LINK	—	Broken link. This file was symbolically linked to a file that has been removed.
DIRECTORY	*	Directories (folders).
EXECUTABLE	*	Executable files.
DATA	*	All files that are not directories or executables and do not match another file type.
NODE	*	On Domain/OS systems only. A network node.

Adding a File Type to the Database

To add a new file type to a database:

1. Choose which database file will contain the new file type. (See "Location of Database Files," above). The file name must end with `.ft`.
2. If you are creating a database file in a directory other than one of the default directories, be sure to add its path to the list of directories specified by the `Vue.TypesDirs` resource.
3. Add the file type to the selected file. The syntax of the file type database is described below.
4. To activate the file type, reread the action definition database by double-clicking the REREAD_DB action icon in the `system_apps/sys_admin` applications directory.

File Type Fields

Below is a portion of a typical file types database file. It defines file types for directories and executable files.

```
DIRECTORY    *  d     dir.l.bm    dir.s.bm
EXECUTABLE   *  !d&x  exec.l.bm   exec.s.bm   EXECUTE
DATA         *  !d!x  data.l.bm   data.s.bm   EDIT,VIEW,PRINT
```


Each definition is a separate line. (Use \ to indicate that the definition is continued on the next line.) A file type definition consists of six fields:

Filetype Filename Mode LargeIcon SmallIcon Actions

Fields are separated from one another by spaces or tabs. The last field can be omitted.

Filetype Field

This is the name of the file type. File types should have unique names.

Filename Field

This field contains the naming convention for the file type. The following table describes the special characters in this field.

Table 12-2. Special Characters in the Filename Field.

Character	Meaning
?	Matches any single character.
*	Matches any sequence of characters (including a null string).
[*cc ...*]	Matches any of the characters (*c*) enclosed in brackets.
[*c*–*c*]	Matches any of the characters in the range *c* through *c*.

The pattern is compared to the full pathname of the file, including the host name (*hostname*:*directory*/*filename*). Thus, to specify a pattern that matches all files named **xyz**, you must specify a filename of ***/xyz**. If you want to

include only files named **xyz** located on host **hpcvxlp**, you must specify a filename of **hpcvxlp:*/xyz**.

The database is searched in order, and the first match is used. Thus, if the filename patterns for two different file types overlap, you must specify the more specific one first. For example, three filetypes with filename patterns ***/ch??.c**, ***.c**, and ***** must be specified in that order.

12

Since personal database files in **$HOME/.vue/types** are searched before the system files in **/usr/lib/X11/vue/types**, you must not specify general name patterns in **$HOME/.vue/types** that obscure other more specific entries in the system files.

Here are some filename examples:

File Name Entry	Interpretation
`*.c`	All files ending with `.c`.
`chapter?.text`	All files named `chapter`*x*`.text`, where *x* is any character.
`chapter[1-9].text`	All files named `chapter`*x*`.text`, where *x* is a character in the range 1 through 9.
`hpcvxlp:/doc/sysadm/*.b`	All files ending in `.b` in the `/doc/sysadm` directory or any of its subdirectories on host `hpcvxlp`.
`*/Mail/*`	All files in any `Mail` directories or their subdirectories.

Mode Field

The mode field contains a boolean expression specifying the mode requirements for the file. The expression is built from the following operators and values:

Table 12-3. Valid Operators and Values for the Mode Field.

Character	Meaning
!	Logical operator NOT.
&	Logical operator AND.
d	The file is a directory.
r	The file is readable by the current user.
w	The file is writable by the current user.
x	The file is executable by the current user.

You can simulate a logical OR operator by creating several adjacent entries.

Here are some mode examples:

Mode Entry	Interpretation
`!d`	Not a directory.
`!d&!x`	Not a directory and not a file executable by the user.
`!d&r`	Not a directory, and readable by the user.

Large and Small Icon Fields

These two fields contain the location of the file containing the bitmap for this file type's large and small icons. The locations can be:

- The absolute pathname.
- A relative pathname. The system will use the bitmap search path specified by the environment variable `XBMLANGPATH` to look for the file. (Refer to the XmGetPixmap(3X) man page for additional information.)

The large icon is used when the file manager view preference is set to "By Name and Icon." The small icon is used when the file manager view preference is set to "By Name and Small Icon."

Table 12-4. Size of Default HP VUE Large Icons.

File Type	Width	Height
Data files	22 pixels	30 pixels
Executable files	32 pixels	32 pixels
Directories	30 pixels	23 pixels

Table 12-5. Size of Default HP VUE Small Icons.

File Type	Width	Height
Data files	14 pixels	17 pixels
Executable files	16 pixels	16 pixels
Directories	20 pixels	16 pixels

Actions Field

This field contains a list of actions displayed in the file manager Actions menu when a file of this file type is selected. Items in the list are separated by commas. The first action in the list is the default action taken when the user double-clicks the file icon.

Each action must have a corresponding entry in the action definition database.

Example File Types

Example file types are provided in `/usr/lib/X11/vue/examples/types/examples.ft`. See the `README` file in the same directory for additional information.

Using String Variables in Database Files

You can define a string variable in a database file that remains in effect for the rest of that file. The syntax for defining a string variable is:

set *variable_name=value*

Variable names can contain any alphanumeric characters and underscore (_). A new-line character must immediately follow *value*.

To refer to the contents of a variable, use the syntax:

$ [**{**]*variable_name*[**}**]

For example, if a file type file contains:

```
set  TEXT_ACT=EDIT,PRINT
BUDGET *.b  !d  budg.l.bm  budg.s.bm  ${TEXT_ACT},GRAPH
```

then the standard actions for file type BUDGET are EDIT, PRINT, and GRAPH.

The Action Definitions Database

There are several ways that the user may invoke an HP VUE action:

- By dragging a file icon onto an action icon.
- By double-clicking on a file icon. This initiates the default action for the file type.
- By dropping a file icon onto the workspace background. This also initiates the default action for the file type.
- By selecting a file icon, then selecting an action from the file manager Actions menu.
- By double-clicking the action icon.

The action definitions database describes the actions known to the system, the operation done when the action is invoked, and the file type(s) for which the action is valid.

Organization of the Applications Directory and Action Database

When the user clicks the applications directory control in the workspace manager, a file manager window opens with the current directory set to `$HOME/.vue/apps`. This directory acts as the top level directory for browsing all of the actions available to the user. The contents of `$HOME/.vue/apps` is
12 a set of symbolic links, one for each `tools` subdirectory along the user's file management database search path.

Table 12-6. Symbolic Links for Application Directories.

Symbolic Name	Linked Action Database Directory	Contents
`system-apps`	`/usr/lib/X11/vue/types/tools`	System-wide default actions.
`personal-apps`	`$HOME/.vue/types/tools`	Personal user actions.
`local-apps`	`/usr/local/lib/X11/vue/types/tools`	Local actions.

The name `local-apps` appears only when the directory `/usr/local/lib/X11/vue/types/tools` exists.

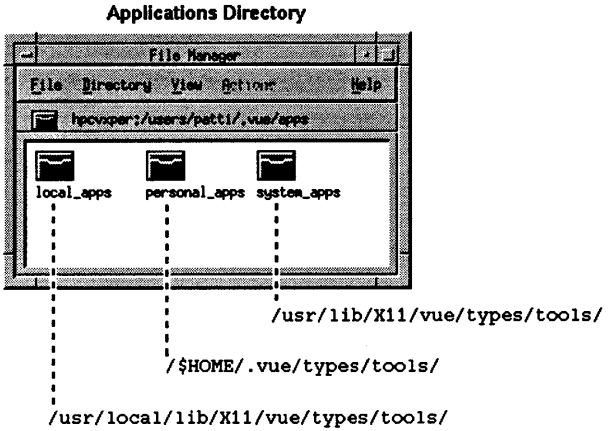


Figure 12-3.
The Applications Directory is Linked to the Action Definitions Database.

The linkage provides user access to the actions in each database directory. Thus, for example, when the current file manager directory is **personal-apps** the user sees the actions contained in the directory **$HOME/.vue/types/tools**.

The set of symbolic links in **$HOME/.vue/apps** are rebuilt by the file manager each time an HP VUE session is started. Therefore, changes to your **Vue.TypesDirs** resource appear in **$HOME/.vue/apps** the next time you start an HP VUE session. For example, suppose the **Vue.TypesDir** resource is modified by adding an additional database directory, as follows:

```
Vue.TypesDir:   $HOME/.vue/types@personal-apps,\
/usr/local/lib/X11/vue/types@local-apps,\
/usr/lib/X11/vue/types@system-apps,\
hpcvlp1:/usr/local/lib/X11/vue/types@remote-apps
```

In this case, the symbolic name **remote-apps** appears in the applications directory the next time the user logs in; it will be linked to **hpcvlp1:/usr/local/lib/X11/vue/types/tools**.

Providing a Graphical Representation of an Action

An action is defined by its entry in the action definition database. Once an action is defined, you can add it to the "Actions" menu for a specific file type, and you can create a control in the workspace manager that accesses that action. However, if you want the action to appear in the file manager applications directory window so that it can be manipulated graphically, you must place a copy of the action in a directory or subdirectory on the file management database search path.

To make a copy of an action, you create an executable file with the same name as the action. The contents of the file doesn't matter (it can be zero length), but it must be executable. Action copies should be placed in the `tools` subdirectory, or a subdirectory beneath `tools`. The action copy is what users see when they press the applications directory control in the workspace manager and browse through the folders. They can invoke the action using this copy, or make additional personal copies of the action.

The following illustration shows the relationship between the user's interaction with the applications directory and the underlying action definition database with its copies of actions. It shows a configuration with system-wide, local, and personal databases. Opening the directory `personal-apps` shows the contents of `tools`—in this case, three subdirectories. The print actions are accessed by opening `printing`, which contains three printing action copies.

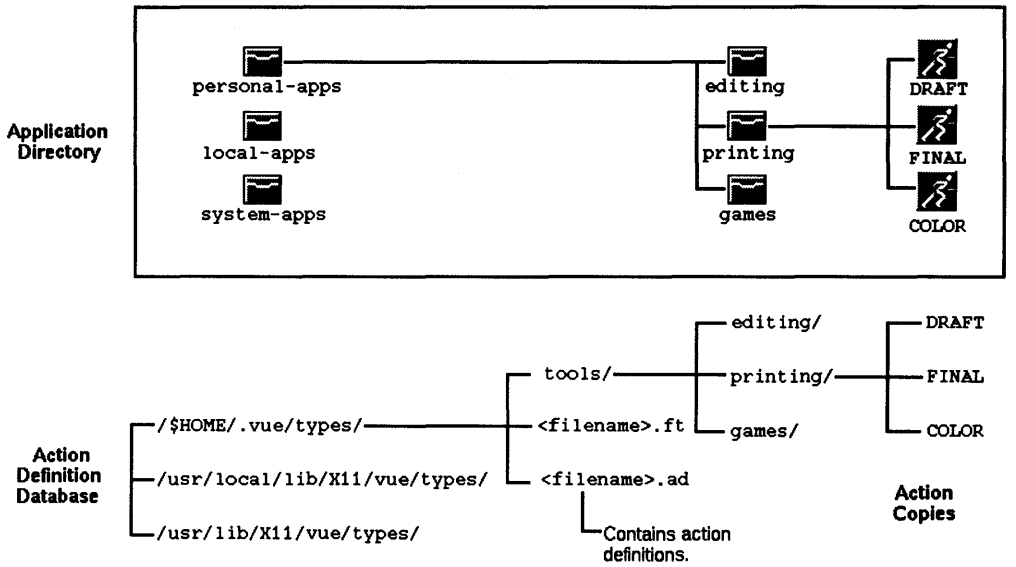


Figure 12-4. The Applications Directory and Action Definition Database.

Adding an Action Definition to the Database

To add an action definition to the database:

1. Choose which database file will contain the new action definition. (See "Location of Database Files," above.) The file name must follow the naming convention *filename*`.ad`.
2. If you are creating a database file in a directory other than one of the default directories, be sure to add its path to the list of directories specified by the `Vue.TypesDirs` resource. New directories do not appear in the applications directory until the user logs out and logs back in.
3. Add the action definition to the selected file. The syntax of the action definition database is described below. You can:
   - Create an entirely new action.
   - Copy an action from the system-wide action definition database and edit it.

4. Create a copy of the action in the same directory as the file containing the definition. This allows the user to access the action using the applications directory. Use the file manager File "New" function to create the file, or execute:

   `touch` *action_name*

12

   where *action_name* is the name specified in the Action Name field. This will create a new file with name *action_name*.

5. Make the action file executable. You can use the file manager File "Properties" function, or execute:

   `chmod 555` *action_name*

6. Move the action file to the `tools` directory or one of its subdirectories. This allows you to access it from a folder beneath the applications directory.

7. If you want to add the action to the Actions Menu for certain file types, add the *action_name* to the appropriate entries in the file types database.

8. To activate the action definition, reread the action definition database by double-clicking the REREAD_DB action icon in the `system_apps/sys_admin` applications directory.

How the Action Database is Searched

Actions with the same name can exist in different directories. For example, you could copy the action VIEW from the system-wide directory `/usr/lib/X11/vue/types` into a personal directory `$HOME/.vue/types` and then modify it to execute a different command. When that action name is invoked from the file manager Actions menu, the database is searched for a match.

Database directories are searched in the order specified by the `Vue.TypesDirs` resource.

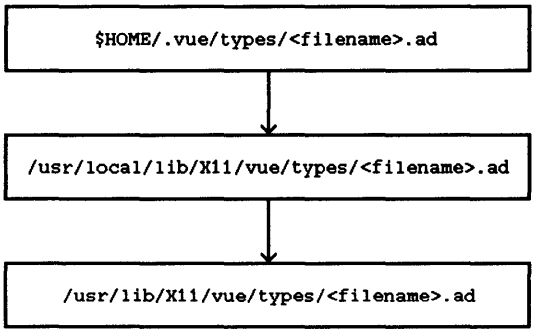


Figure 12-5. Default Search Scheme for the Action Definition Database.

The order of action definition entries within a single database file is preserved. However, the order of searching multiple files within a single directory is arbitrary.

Action Definition Fields

Below is a portion of a typical action definition database. It defines two actions.

```
BITMAP BM  ${ACTION_L}  ${ACTION_S}  MAP  BITMAP_DSIZE
BITMAP_DSIZE    \
    *           \
    ${ACTION_L} \
    ${ACTION_S} \
    COMMAND     \
    NO-STDIO    \
    %LocalHost% \
    /usr/bin/X11/bitmap %(File)Arg_1"File To Edit:"%
```

Each definition is on a separate line. (Use \ to indicate that the definition is continued on the next line.) Each action definition consists of 8 or 6 fields, depending on whether the action contains the command to be executed, or if is is mapped to another action:

Command actions have the syntax:

Name FileType(s) LargeIcon SmallIcon **COMMAND** *WindowType Exec-host Exec-string*

Mapped actions have the syntax:

Name FileType(s) LargeIcon SmallIcon **MAP** *AlternateAction*

12

Action Name

This is the name of the action.

File Type(s)

This field consists of a list of file types for which the action is valid. Individual file types are separated by commas. If this field contains only the special character *, the action is valid for all file types.

You can create an action that behaves differently for different file types by creating multiple action definitions, each of which specifies a different file type.

Large and Small Icon Fields

These fields contain the locations of the files containing the bitmaps for this action's large and small icons.

The locations can be:

- The absolute pathname.
- A relative pathname. The system will use the bitmap search path specified by the XBMLANGPATH environment variable to look for the file. (For additional information, see the XmGetPixmap(3X) man page.)

The large icon is used when the file manager view preference is set to "By Name and Icon." The small icon is used when the file manager view preference is set to "By Name and Small Icon."

Table 12-7. Size of Action Icons.

Icon Size	Icon Dimensions
Large	32 × 32
Small	16 × 16

Action Type

There are two action types.

Table 12-8. Action Types.

Type	Description
MAP	MAP-type actions are not executed directly. Instead, they are mapped to another action specified in the Alternate Action field of the action definition.
COMMAND	COMMAND-type actions are commands to be executed.

The contents of this field determines the requirements for the rest of the action definition.

- MAP types will have one more field—the Alternate Action Field.
- COMMAND types will have three more fields—Window Type, Exec-host, and Exec-string.

You may notice other action types in the system action definition files. These are used to access internal capabilities of HP VUE.

Alternate Action Field for MAP Action Types

If the Action Type field is MAP, then the next field is the Alternate Action field. This field contains the name of another action to invoke.

For example, if the actions definition database contains the following lines:

```
EDIT  *  edit.l.bm  edit.s.bm  MAP      VI
```

```
VI    * vi.l.bm    vi.s.bm    COMMAND  TERMINAL  %DataHost% \
      vi%(File)Arg_1"Edit what file?"%
```

the VI action is used when the EDIT action is invoked.

12 **Window Type**

The Window Type field is used with COMMAND action types. It specifies the windowing support required to execute this action.

Table 12-9. Window Type Values for COMMAND Action Types.

Value	Definition	Typical Use
NO-STDIO	No windowing support is required.	With X clients that handle their own user interface.
TERMINAL	The command must be executed in a terminal window. The window is closed when the command is exited.	Full-screen commands (for example, `vi`).
PERM-TERMINAL	The command must be executed in a permanent terminal emulation window (the window is not closed when the command terminates).	Commands that take some input, produce some output, then terminate.
The following terminal types apply only to operating systems using `hpterm` as the default terminal emulator.		
OUTPUT-ONLY	The command produces output that must be displayed, but requires no input.	Commands that execute, produce output, and then terminate, such as `ls` and `ps -ef`.
SHARED-OUTPUT	Similar to OUTPUT-ONLY, except only one window of this type is created per workspace. All SHARED-OUTPUT commands share this window.	Commands producing a small amount of output.

Exec-host

This field specifies the host where the command will be executed.

Table 12-10. Valid Exec-Host Values.

Value	Description	Typical Use
hostname	The named host.	Environments in which the action should always be invoked on one particular host.
%LocalHost%	The host where HP VUE is running.	Environments where all processing should be done on one common host.
%DataHost%	The host containing the first file argument.	Situations where you want to spread out the processing load, or where you anticipate problems with remote data access.
%DatabaseHost%	The host where the action definition for this action was found.	Where applications are installed on only one system in a network, and you cannot assume that the application performing the action resides on any other host other than the one that defined the action.
%"*prompt*"%	Prompts the user for the host name each time the action is invoked.	Environments where user needs control of the execution host.

Exec-string Field

The exec-string field contains the command line for the command to be executed.

For example, the following exec-string executes the client `xclock`. The command line requires no arguments, so the exec string is very simple:

```
xclock
```

The command is executed directly, rather than being passed to a shell. Therefore, if a command requires shell capabilities, you must specify the shell in the execution string—for example:

```
/bin/sh -c 'ps | lp'
```

12 **Action Arguments.** Some actions must accept arguments that are supplied by the user when the command is invoked. These arguments fall into two general categories:

- File or directory arguments. For example, when `xrdb` is used with the `-edit` option, it takes a file argument. The user can supply the argument by dragging a file icon to the action icon, or by selecting a file icon and choosing the action from the Actions menu.
- A non-file argument. An example of this is the `bitmap` client, which takes an argument that sets the size of the bitmap.

The following table describes the syntax for including file and non-file arguments in exec strings.

Table 12-11. Syntax for the Execution String.

Keyword	Meaning
`%(File)Arg_`*n*`%`	Substitute the *n*th file argument. If the file name is in the form *hostname*`:/`*path/filename*, convert the name to the form `/nfs/`*hostname/path/filename*.
`%Arg_`*n*`%`	Substitute the *n*th file argument. Do *not* translate names from the form *hostname*`:/`*path/filename* to the form `/nfs/`*hostname/path/filename* form.
`%(File)Arg_`*n*`"`*prompt*`"%`	Substitute the *n*th file argument. If it is not found, prompt the user for a file name and substitute the response. If the file name is in the form *hostname*`:/`*path/filename*, convert the name to the form `/nfs/`*hostname/path/filename*.
`%(File)"`*prompt*`"%`	Prompt the user for a file name and substitute the response. The user can enter file names in the form *hostname*`:/`*path/filename*. The file names will be translated into standard filenames.
`%(File)Args%`	Substitute all of the remaining arguments.
`%"`*prompt*`"%`	Prompt the user and substitute the response. Do not use this syntax when prompting for a file name.

For example, the following exec-string provides the ability to execute `xrdb -edit` by dragging a data file to the action:

```
xrdb -edit %(File)Arg_1%
```

With the next exec-string, the action can be started by dragging a file to the action icon or by double-clicking the action icon to display a prompt for the file.

```
xrdb -edit %(File)Arg_1"Enter file to cat:"%
```

Since the file manager does not provide a graphical interface for non-file arguments (for example, you cannot drag a geometry argument to an action), you must use the %"*prompt*"% syntax to supply non-file arguments.

For example, the following exec-string executes the `bitmap` client, which requires both a file name and a geometry option.

```
bitmap %(File)Arg_1"Enter filename:"% \
%"Enter geometry <width>x<height>:"%
```

The next exec-string prints a file with a banner containing the file name on an HP-UX system:

```
lp -t%Arg_1% %(File)Arg_1%
```

Note the different syntax for the two file arguments. The second argument (`%(File)Arg_1%`) converts an argument entered in the *hostname*`:`*/path/filename* syntax to correct operating system syntax. For example, if the user enters `hpcvaaa:/users/anna/myfile`, the name is converted to `/nfs/hpcvaaa/users/anna/myfile`. The first argument prints the unconverted name on the banner (`hpcvaaa:/users/anna/myfile`).

Specifying the Current Working Directory. When a command uses one or more file or directory arguments, the first argument determines the current working directory for the command:

- If the argument is a file name, the current working directory is set to the directory containing that file.
- If the argument is a directory name, the current working directory is set to that directory.

Invoking Actions with Multiple Arguments. When an action accepts only one argument, but the user invokes it with more than one argument (for example, the user selects several files and drags them to the action icon), the action is executed repeatedly, once for each argument.

When an action specifies more than one argument, only that number of arguments is allowed.

Executing Remote Files

HP VUE can be configured to provide a great deal of network transparency. For example, a user can drag the icon for a text file located on remote host A to an action icon for an application located on host B, and see the output on the user's own local system.

This section covers some aspects of remote execution. Chapter 4, "Distributed Operation," contains additional information.

Enabling Remote Hosts

Before the local system can run an application on a remote host, the remote host must be configured to allow use of its `spc` service. To do this, you must edit the file `/usr/adm/inetd.sec` on the remote host to include the local host. (See "Controlling Access and Security" in chapter 4.)

Accessing Remote Files

In order for HP VUE to provide the ability to execute an application on one host using data from a different host, NFS mounts must be established on the execution host to access the files on the data host. The mount point for the data host's file system must be `/nfs/`*hostname*.

Specifying the Remote Execution Host

Actions can be invoked on a remote execution host if any of the following conditions are met:

- The action uses a hard-coded execution host that is different from the local host.
- The action specifies `%DataHost%` as the execution host and that the data host is a remote host.
- The action specifies `%DatabaseHost%` as the execution host and that the action is defined on a remote host.

- The action contains a prompt-string for the execution host and the user supplies a remote hostname when prompted.

Order of Entries in the Action Definition Database

12

When an action is invoked with a file argument (by dragging a data icon to an action icon, or by selecting a file and then selecting an action from the Actions menu):

1. The system uses the file type database to determine the file type of the argument.
2. The system searches the action definition database for the first occurrence of the action that accepts that file type.

Thus, entries in the action definition database should be ordered according to how selective they are about file types, with the most selective first.

For example, consider the following entries in the Action Definition database (only the first two columns are shown):

```
EDIT GRAPHICS ...
EDIT *    ...
```

This database defines an EDIT action that behaves differently for GRAPHICS files than for all other file types.

If the entries were reversed, the first entry (`EDIT *`) would match all file types, and would be used for all files, including GRAPHICS files.

Example Action Definitions

Example action definitions are provided in `/usr/lib/X11/vue/examples/types/examples.ad`. See the `README` file in the same directory for additional information.

An Action Definition Example

There are often many different ways you can create an action database to solve a particular problem. The advantages and disadvantages of each approach may not be obvious.

The examples in this section may help you choose the best technique for your application. They show several different ways to modify the action and filetype databases to support a new application. The (fictitious) application is a spreadsheet editor named **spreadsheet**. It runs in a terminal emulator window and is invoked with the command line:

spreadsheet *filename*

Example 1: A Simple File Type and Action Definition

Before setting up the file management database for **spreadsheet**, you must decide on a naming convention for spreadsheet files. A reasonable choice is *filename*.**ss**. To create the file type definition, add the following entry to a *filename*.**ft** configuration file:

```
SS  *.ss  !d  ss-data.l.bm  ss-data.s.bm  SPREADSHEET
```

Next, create an action by adding the following entry to a *filename*.**ad** configuration file:

```
SPREADSHEET  SS  ss-app.l.bm  ss-app.s.bm COMMAND \
     TERMINAL  %LocalHost%  spreadsheet %(File)Arg_1%
```

These two entries create a database with the following properties:

- Any file named *filename*.**ss** is an SS file, and will be displayed using one of the "ss-data" icons.
- When an SS file is selected, the file manager Actions menu lists the single entry SPREADSHEET.
- If an SS file is double-clicked (or dragged onto the workspace window) the SPREADSHEET action is invoked with the SS file as an argument.
- If the SPREADSHEET action is invoked with no arguments (by double-clicking a SPREADSHEET icon), the **spreadsheet** command is executed with no arguments. This may or may not be a problem.

Example 2: An Action that Prompts for a File Argument

To change the behavior of the SPREADSHEET action so that double-clicking it opens a dialog box asking for a filename, use the following action entry:

```
SPREADSHEET  SS  ss-app.l.bm  ss-app.s.bm COMMAND \
  TERMINAL  %LocalHost%  spreadsheet %(File)Arg_1"Filename:"%
```

12

Example 3: An Action With Less Stringent File Type Requirements

In examples 1 and 2, the user cannot process a file with the SPREADSHEET action unless it follows the naming convention *filename*`.ss`. Attempting to drag a differently-named file onto a SPREADSHEET application icon displays a dialog box saying it could not find an appropriate action definition.

To solve this problem, change the SPREADSHEET action so that it accepts any type of file:

```
SPREADSHEET  *  ss-app.l.bm  ss-app.s.bm COMMAND \
  TERMINAL  %LocalHost%  spreadsheet %(File)Arg_1"Filename:"%
```

Because the SS filetype still lists SPREADSHEET as the default action, double-clicking an SS file will invoke SPREADSHEET. However, a user can now invoke the SPREADSHEET action with file that doesn't conform to the SS naming convention by dragging the file onto a SPREADSHEET icon.

Example 4: Configuring a Preexisting Action for a New File Type

The previous examples create the SPREADSHEET action and defined it as the default action for a type of file (SS files). A different approach is to leave EDIT as the default action for SS files and redefine how EDIT behaves with an SS file.

To do this, use the following file type definition in a *filename*`.ft` configuration file:

```
SS  *.ss  !d  ss-data.l.bm  ss-data.s.bm  EDIT
```

Add the following entry to a *filename*`.ad` configuration file:

```
EDIT  SS  ss-app.l.bm  ss-app.s.bm COMMAND \
  TERMINAL  %LocalHost%  spreadsheet %(File)Arg_1"Filename:"%
```

Make sure it is placed before a more general EDIT definition that would include SS files—for example, an action definition that begins with `EDIT *`.

Note that this example has the same limitation as example 2—there is no way to invoke the SPREADSHEET action on a non-SS file.

Example 5: Mapping One Action to Another

The following action definitions use mapping to remove the major limitation of example 4. It allows any file type to be dragged to the SPREADSHEET action. Furthermore, it creates an "intelligent" EDIT action that invokes SPREADSHEET when an SS file is dragged to it.

```
SPREADSHEET  *  ss-app.1.bm  ss-app.s.bm COMMAND \
  TERMINAL  %LocalHost%  spreadsheet %(File)Arg_1"Filename:"%

EDIT  SS  action.1.bm  action.s.bm  MAP  SPREADSHEET
```

Example 6: Providing a Second Action for a File Type

With the above examples, dragging an SS file onto the printer control in the workspace manager invokes the standard PRINT action. This will probably not produce the desired output. Thus, you will probably want a special action that prints spreadsheet files. This example assumes that spreadsheet files can be printed with the command line:

```
spreadsheet -print filename
```

First, the SS entry in the *filename*`.ft` configuration file is changed to:

```
SS  *.ss  !d  ss-data.1.bm  ss-data.s.bm  EDIT,PRINT
```

Then, the following entries are added to the *filename*`.ad` configuration file:

```
PRINT-SS  *  action.1.bm  action.s.bm COMMAND  NO-SDTIO \
  %LocalHost%  spreadsheet -print %(File)Arg_1"Filename:"%

PRINT  SS  action.1.bm  action.s.bm  MAP  PRINT-SS
```

13

Customizing the Workspace Manager

This chapter covers:

- Resources used to change the appearance and location of the workspace manager.
- Turning the workspace manager off and on.
- Rearranging the components in the panel.
- Changing the contents of the panel.
- Creating an entirely new workspace manager.

The workspace manager is also referred to as the "front panel." That designation is used for its resource and configuration specifications.

Overview

The workspace manager is a panel containing a variety of controls managed by the window manager. *Control options* provide a variety of ways to interact with the control:

- Read-only indicators. These show aspects of the system status—for example, the clock and date indicators.
- File indicators. These are read-only indicators that indicate when a designated file has changed—for example, the **Mail** control (**Mail** is also a toggle button.)

13

- Control buttons:
  - Push buttons. Push buttons cause the system to display a new window. When the user clicks on the button, it is visually depressed and immediately raised, ready to be pushed again.
  - Toggle buttons. In general, toggle buttons are used for processes that don't allow multiple occurrences. For example, the **Mail** toggle button cannot be used to display more than one Mail window. Clicking on the button when a Mail window is already open simply raises that window to the top of the window stack in the current workspace.
  - Drop zones. These are areas to which a user can drag a file icon in order to perform a task. An example of a drop zone is the trash can.
- Active client windows. For example, the load meter is an active **xload** client window.

A workspace manager component can have more than one control option. For example, the **Mail** control is both a file indicator and a toggle button; its bitmap changes when the mailbox receives mail, and clicking on the control displays the mail window.

Workspace Manager Resources

The window manager provides resources that let you customize the appearance and behavior of the workspace manager. Like other resources, workspace manager resources can be specified by:

- Using **xrdb** to modify the user's resource database.
- Editing **/usr/lib/X11/app-defaults/Vuewm**, for application-specific resources available more generally.

Changes made to workspace manager resources take effect when the workspace manager is restarted.

13

Syntax of Resources

The general syntax for specifying resources for the workspace manager is:

```
Vuewm*frontPanel [*row_name]*resource: value
```

where *row_name* is name of the row as specified in the portion of the window manager resource file that configures the workspace manager. The default row names are `Top` and `Bottom`.

Running HP VUE Without the Workspace Manager

The `useFrontPanel` resource determines whether or not a workspace manager is created. The resource is set to "True" in the file **/usr/lib/X11/app-defaults/Vuewm**. For example, the following line specifies that no workspace manager be displayed on any screen:

```
Vuewm*useFrontPanel:  False
```

Appearance Resources

Appearance resources affect:

- The position of the panel.
- The spacing of the panel elements.
- The color and background pattern.

Position of the Panel

The workspace manager has a **geometry** resource that specifies the initial placement of the workspace manager. It has the syntax:

=±*x-offset* ±*y-offset*

The sign of the offset values determines how they are interpreted:

Table 13-1.
Using the Geometry Resource for the Workspace Manager.

13

Offset	Value	Effect
x-offset	Positive	Panel is left-justified against the specified offset.
x-offset	±0	Panel is centered horizontally.
x-offset	Negative	Panel is right-justified against the offset.
y-offset	Positive or +0	Panel is placed that many pixels down from the top of the display.
y-offset	Negative or −0	Panel is placed that many pixels up from the bottom of the display.

For example, the following resource centers the workspace manager at the top of the display:

```
Vuewm*frontPanel*geometry:      =+0+0
```

The default is to center the workspace manager on the bottom of the display.

The **geometry** resource does not affect the workspace manager size; the size is controlled by the geometries of the individual components and the spacing between components.

Spacing Within the Panel

The following resource take values in pixel units. The default is 0 pixels.

Table 13-2. Appearance of the Workspace Manager.

Resource	Description
`borderWidth`	The width of the border between the frame and controls.
`controlSpacing`	The width between controls.
`rowSpacing`	The distance between the top and bottom rows.

For example, the following line sets the border width to 7 pixels:

```
Vuewm*frontPanel*borderWidth:  7
```

Workspace Manager Color

The `colorSetId` resource specifies the palette color set used for the background and shadow colors of each row of the workspace manager. It takes as its value a color set number (1 through 8).

The default color set depends on the display.

Table 13-3. Workspace Manager Default Colors.

Display Type	Top Row Color Set	Bottom Row Color Set
High Color	5	3
Medium Color	4	3
Low Color	1	2
Monochrome	1	2

For example, the following line causes the front panel to use color set 4 for both the top and bottom rows (named `Top` and `Bottom`).

```
Vuewm*frontPanel*Top*colorSetId:        4
Vuewm*frontPanel*Bottom*colorSetId:     4
```

Workspace Manager Background Pixmaps

The `rowBackgroundPixmap` resource specifies the pixmap used to tile the workspace manager background. For example, the following lines specifiy pixmaps used to tile both rows of the workspace manager.

13

```
Vuewm*frontPanel*Top*rowBackgroundPixmap:      vertical_tile
Vuewm*frontPanel*Bottom*rowBackgroundPixmap: ribbing.l.bm
```

Note that there are different ribbing bitmaps for medium and high resolution displays.

The default workspace manager uses the `50_foreground` tile for the top row and ribbing for the bottom row.

Specifying the Contents of the Workspace Manager

This section describes:

- The location of files that specify the content and organization of the workspace manager.
- The general syntax for specifying the rows and contents of the panel.
- The syntax for specifying the appearance and behavior of the individual components.

Window Manager Resource Files and Their Locations

The contents of the workspace manager is specified in a window manager configuration file.

Refer to the section "The Window Manager Configuration File" in chapter 11 for a description the configuration file.

General Syntax for Specifying the Workspace Manager

The contents and organization of the workspace manager is defined by an entry in the window manager resource file. The entry has the syntax:

```
FrontPanel panel_name
{
 Row [row_name]
 {
  panel_control
  panel_control
       ⋮
 }

 Row [row_name]
 {
  panel_control
  panel_control
       ⋮
 }

 ⋮

}
```

The panel is built from left to right, top to bottom. Each *panel_control* entry creates a control of some sort in the workspace manager. The *panel_control* entries for each row are listed in the same order that they are displayed in the panel. A geometry option for each entry lets you set the size, but not its location, in the panel. Nested rows are not allowed.

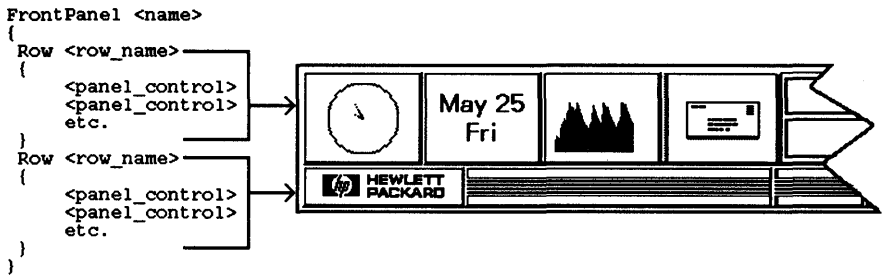


Figure 13-1. The Workspace Manager is Built Row by Row.

The default panel has two rows; however, you can add additional rows.

Types of Workspace Manager Controls

There are three basic categories of workspace manager controls:

- Predefined controls. These are controls created by window manager reserved words, and have predefined appearance and behavior. In certain cases, a predefined control can be modified (see "Predefined Controls" later in this chapter).
- Generic controls. The appearance and behavior of these controls are defined in the *panel_control* entry for the controls. Generally, these controls invoke actions in the actions definition database.
- Clients. These are windows of active clients running in the workspace manager.

The Syntax of Components

The entry for each item in the workspace manager consists of the control name followed by a number of optional fields that specify the appearance and behavior of the control.

control_name [*control_option(s)*] [*bitmap(s)*] [*geometry*] [*function(s)*]

Each of these fields is described in detail below.

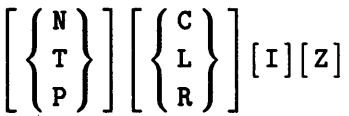
Control Name Field

Each control must have a unique name. Use the following guidelines for control names:

- If the control is a toggle button that displays a pop-up window, the control name should be the same as the resource name of the window it creates (usually, the name of the client). This ensures that the window manager knows when the window is open.
- If the component is a predefined control, use the window manager reserved word.
- If the component is a client, you may want to use the client resource name.

Control Options Field

The *control_options* field specifies how the user interacts with the control. It is enclosed in square brackets and has the syntax:



Individual entries in the control options field must be separated from one another by one or more spaces—for example, `[T C I]`.

13

Table 13-4. Control Options.

Control Option	Description
`N`	Read-only indicator (the normal default).
`T`	Toggle button.
`P`	Push button.
`C`	Center the label or image (the default).
`L`	Left-justify the label or image.
`R`	Right-justify the label or image.
`I`	The control is a file indicator. The file that is monitored is specified by the **`fileName`** resource. (See "Adding a File Indicator Control" later in this chapter.)
`Z`	The control is a drop zone.

For non-predefined controls, any combinations are possible except:

- Multiple justifications. If more than one justification is specified, only the last is used.
- Client windows cannot be push buttons or toggle buttons. When a client is running in the front panel, it receives all the pointer and keyboard input when it has input focus.
- Client windows cannot have a bitmap specification.

Some predefined controls allow you to specify control options in addition to the ones built into the definition.

Bitmap Field

The syntax for specifying a bitmap is:

$$[@]\left\{\begin{array}{l}\texttt{absolute path}\\ \texttt{relative path}\end{array}\right\}$$

If you use a relative pathname, the system will use the bitmap search path specified by the **XBMLANGPATH** environment variable to look for the file. (For additional information, see the XmGetPixmap(3X) man page.)

The bitmaps for the default workspace manager are located in **/usr/lib/X11/bitmaps/Vuewm**.

Geometry Field

The syntax for the geometry is:

=*width* [x*height*] [+*x-offset*+*y-offset*]

The offsets set the distance, in pixels, between the edge of the control and the control image (bitmap). If no offset is specified, the image fills the entire control.

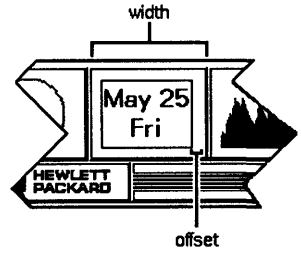


Figure 13-2. The Geometry of Workspace Manager Controls.

13 If no geometry is specified, the default is to use the width and height of the control to the left of the one being specified. The default geometry for the first control in any row is dependent on the display resolution.

Table 13-5. Default Control Geometries.

Display Resolution	Width	Height
Medium (1024x768)	80	80
High (1280x1024)	100	100

The `vuewmbusy` control takes two geometries. The first defines the size of the control; the second defines the size and position of the progress light within the control.

Functions Field

The Function field describes what happens when the user interacts with an a push button, toggle button, or drag and drop zone. The syntax of the Function field is:

{ `f.action` *action_name*
`f.exec "`*command*`"`
window_manager_function }

When the `f.action` form of the Function field is used, activating the control performs the specified action in the action definition database. (The large icon and small icon fields in the database are ignored.)

For example, the following entry in the workspace manager specification in **sys.vuewmrc** adds a drag and drop zone to the workspace manager.

```
VI [Z] pencil.m.bm =90  f.action VI
```

The action that takes place when a file is dropped on the control is specified in the action definition database:

```
VI      *      vi.l.bm      vi.s.bm    COMMAND  TERMINAL  \
%DataHost%     vi%(File)Arg_1"Edit what file?"%
```

Note that in order for actions to work properly with files on remote systems, NFS mounts must exist between the file systems on the two hosts.

The action definition database is described in chapter 12.

Specifying the Shell for 'f.exec' Commands. Commands executed with **f.exec** use the shell specified by:

- The **MWMSHELL** environment variable.
- The **SHELL** environment variable (if **MWMSHELL** doesn't exist).
- **/bin/sh** (if neither variable exists).

Specifying Multiple Functions. If a control is both a button (push button or toggle button) and a drop zone, the Function field for the button must be first. For example, the Trash control is specified by the line:

```
Trash[P Z] @trash.l.bm f.action SHOW_TRASH f.action THROW_AWAY
```

where SHOW_TRASH is the action for the push button and THROW_AWAY is the action for the drop zone.

Monitoring File Status with Panel Controls

When a panel control is a file indicator (control option `I`), the file it monitors is specified by the **fileName** resource, which has the syntax:

Vuewm*control_name***fileName:** *file_name*

For example, if the control panel contains:

```
schedule        [I P] @calendar.bm @exclaim.bm
```

then the resource entry:

13

```
Vuewm*schedule*fileName:    /users/yoder/schedules/projects.text
```

causes the control to monitor the specified file.

Monitored files must be local files or files to which NFS mounts exist.

Drop Effects

When a panel control is a drop zone, you can specify a series of bitmaps for the animated transition effect that is activated when a file is dropped on the control. Use the syntax:

```
DropEffects control_name
{
  bitmap_file   [time_delay]
  bitmap_file   [time_delay]
  bitmap_file   [time_delay]
  :
}
```

where *time_delay* is the number of milliseconds. The default is 200 milliseconds. When a *time_delay* is specified, it is used for the files beneath it in the list until another *time_delay* is specified.

Trash Can Files

When a file is dropped on the trash control:

- The file is changed to a hidden file in its directory of origin and given an encoded name.
- Its original and encoded names are written to **$HOME/.vue/.trashinfo**.

When the user requests trash information by clicking on the trash control, the Trash Can window displays original names of the contents of **.trashinfo**.

Predefined Controls

The following table describes the window manager reserved words that create predefined controls.

Table 13-6. Predefined Controls.

Reserved Word	Default Control Type	Description
`vuewmblank`	Read-only indicator	Blank space. If the row has background ribbing, it will not show.
`vuewmrib`	Read-only	Black space. If the row has background ribbing, it will show.
`vuewmbox`	Toggle button	Specifies use of an icon box; reopens the icon box when it has been closed.
`vuewmclock`	Read-only indicator	Clock displaying the system time.
`vuewmdate`	Read-only indicator	System date.
`vuewmname`	Toggle button	Displays the dialog box for renaming workspaces.
`vuewmswtch`	Read-only indicator	Contains buttons for switching workspaces.
`vuewmbusy`	Read-only indicator	Provides a system "progress" light.

The `vuewmblack` and `vuewmrib` controls can be used more than once. Only one of each of the other controls is allowed. For example, if you try to include two `vuewmclock` controls, only the first will be inserted into the workspace manager; the other one will be ignored.

Augmenting the Behavior of Predefined Controls

Predefined controls can be modified to some extent. For example, you can give `vuemwclock` push button behavior; that is, clicking on the clock will execute a certain action.

The following table shows the allowable customization.

- The **Control options** column lists the options that can be specified.
- The **Bitmap** and **Geometry** columns indicate whether these fields are allowed.
- The **Function** column indicates whether the control can have a Functions field, and which of the three types (`f.action`, `f.exec`, and other window manager functions) are allowed.

Table 13-7. Customizing Predefined Controls.

Reserved Word	Control options	Bitmap	Geometry	Function
`vuewmblank`	`Z P T`	no	yes	any
`vuewmrib`	`I Z P T`	no	yes	any
`vuewmbox`	`Z`	yes	yes	f.action
`vuewmclock`	`Z P T`	no	yes	any
`vuewmdate`	`Z P T`	no	yes	any
`vuewmname`	`Z`	yes	yes	f.action
`vuewmswitch`	none	no	yes	none
`vuewmbusy`	`I Z P T`	yes	yes	any

Modifying the Appearance of the Workspace Switch

The following resources customize the appearance of the Workspace Switch.

Table 13-8. Resources for the Workspace Switch Appearance.

Resource	Description
`wsButtonBorder`	The width in pixels of the border around the Workspace Switch.
`wsButtonShadowThickness`	The width in pixels of the top and bottom shadows of the buttons.
`wsButtonSpacing`	Horizontal spacing in pixels between the buttons.
`numberOfSwitchRows`	The number of rows of buttons.

For example, the following line specifies that the buttons will be arranged in three rows:

```
Vuewm*numberOfSwitchRows:  3
```

The default switch background color is obtained from the color specification for the row of the workspace manager where it is located. The button colors are matched to the colors of the backdrops.

Some additional resources for the workspace switch can be set using the component name `Switch`. For example, the following syntax defines a font for the workspace switch.

```
Vuewm*Switch*fontList:  font
```

Configuring 'vuewmbusy'

To following resource configures the progress indicator as a blinking light:

```
Vuewm*useBlinkingIndicator:    True
```

When the resource is set to "False," the progress indicator is a bitmap design specified by the second bitmap in the control.

The **vuewmbusy** control can take two geometries. The first defines the geometry of the control; the second specifies the size and offset of the progress light within the control. Thus, you can visually embed the progress light within some other control. In the default workspace manager, the **vuewmbusy** control consists of the progress light inside the push button for the action EXIT_SESSION (the logout button).

Changing the Appearance of 'vuewmdate'

The format for the date displayed by **vuewmdate** can include multiple lines and multiple fonts.

The syntax for **vuewmdate** resources is:

```
Vuew*vuewmdate*resource; value
```

Table 13-9. vuewmdate Resources.

Resource	Description
format	The format string for the date. See the vuewm(1X) and strftime(3) man pages for more information.
fontList	The font used for the text.
fontList*n*	The font used for the *n*th line, where *n* is an integer one through 5. The default value is the value of the **fontList** resource.

Controls in the Default Workspace Manager

The following two tables and accompanying illustration describe the controls in the default workspace manager.

Table 13-10. Default Workspace Manager—Top Row.

Name	Type	Option(s)	Description
`vuewmclock`	Reserved	Read-only	Analog clock
`vuewmdate`	Reserved	Read-only	System date
`vueload`	Client	none	Load meter. `vueload` displays the client window `xload` started with the resource name `vueload` (by the command line `xload -name vueload`).
`Mail`	Generic	File indicator; toggle button; drop zone	Mail box status; displays mailer window (default is `elm`); runs `mailx` with file argument.
`vuewmswtch`	Reserved	Read-only	Controls for switching workspaces.
`Printer-Info`	Generic	Drop zone; toggle button	Starts a print process; displays the print status.
`Home`	Generic	Push button	Displays the file manager— home directory.
`Application`	Generic	Push button	Displays the file manager— applications directory.
`Trash`	Generic	Drop zone; Push button	Removes the file; displays the contents of the trash can.

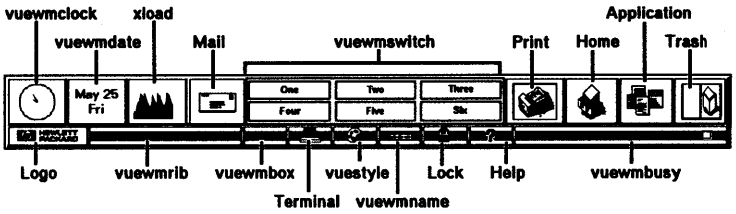


Figure 13-3. Controls in the Default Workspace Manager.

Table 13-11. Default Workspace Manager—Bottom Row.

`Logo`	Generic	Push button	Displays version of HP VUE.
`vuewmrib`	Reserved	Read only	Displays ribbing bitmap
`vuewmbox`	Reserved	Push button	Specifies use of an icon box; reopens the icon box when it has been closed.
`Terminal`	Generic	Push button	Displays a terminal emulator window (`hpterm` for HP-UX and OSF/1 systems, `xterm` for all others).
`vuestyle`	Generic	Toggle button	Displays the style manager.
`vuewmname`	Reserved	Push button	Displays the Rename Workspace dialog box.
`Lock`	Generic	Push button	Locks the session.
`Help`	Generic	Push button	Displays the help manager Help Index window.
`vuewmbusy`	Reserved	Push button	System progress light; exit session.

Customizing the Workspace Manager

This section covers:

- Adding and removing components from the workspace manager.
- Creating a new workspace manager.
- Creating a vertical workspace manager.

Changes to the workspace manager take effect when **vuewm** is restarted.

13

Removing a Control from the Workspace Manager

To remove a control, delete the line that creates that control. You may need to adjust the size of other elements to compensate for the item that has been removed. Note that each control is surrounded by bevels that create a 3-D effect. By default, the bevels are two pixels wide.

Adding a Control to the Workspace Manager

This section describes the steps for adding controls to the workspace manager. There are separate instructions for:

- Push button and toggle buttons.
- Client windows.
- File indicators.
- Drag and drop zones.

If the addition is a read-only client window, the client must be started after **vuewm** has been restarted.

Adding Push Buttons and Toggle Buttons

To add a push button or toggle button to the workspace manager:

1. Select the way the user will interact with the control—push button or toggle button.
2. Create an image file for the entry. Typically, this is a bitmap displayed in the panel.
3. Add the panel control to `vuewmrc`. If the control is a toggle button that displays a pop-up window, the control name must match the window resource name (usually, the name of the client).
4. Create an action in the action definition database that describes the action to be taken when the user interacts with the control. (If the control is not connected to an action—for example, it executes an operating system command—this step is unnecessary.)

Adding a Client Window to the Workspace Manager

To add a client window to the workspace manager:

1. Add the panel control to **vuewmrc**. Do not include an *image* or *label*.

2. Start the client from command line using the syntax:

   application_name **–name** *control_name*

The command line must be executed after the window manager is running, and there should be no other clients running with the same name.

13

For example, you could add the line:

```
terminal 200x90
```

to a the workspace manager specification in **vuewmrc**. Then, execute the following command in an existing terminal window to start the live terminal window in the workspace manager.

```
xterm -name terminal -xrm 'terminal*primaryColorSetId: 5'
```

The **-xrm** option is used to specify the resource **primaryColorSetId** for the window. This resource causes the window to be the same color as the other entries in the top row.

To use the window, position the pointer within it to give it focus.

You cannot configure a client window as a push button or toggle button; the client will receive all keyboard and pointer input. In addition, you cannot configure a client window as a drop zone.

Adding a File Indicator Control

File indicators can be used for local files and for nfs-mounted files.

To add an file indicator control to the workspace manager:

1. Create two image files for the entry. The control displays the second image when the monitored file is changed. It returns to the first image when one of the following happens:

   a. The user clicks the control.

   b. The monitored file shrinks. File size is monitored every 10 seconds.

   c. The window displayed by the control's action (if the control is a push button or toggle button) is closed.

2. Add the panel control to `vuewmrc`. The *control_option* field must include type `I`. If the control is simply a file indicator (it is not a push button, toggle button, etc.,), it does not have an action associated with it.

3. Use the `fileName` resource to specify the file to be monitored. It has the syntax:

   `Vuewm*`*control_name*`*fileName:` *path/filename*

   The value must be the absolute path of the file.

Adding a Drop Zone

To add a drop zone to the workspace manager:

1. Create an image file for the entry. This will be the bitmap displayed for the control in its rest state.

2. Add the panel control to `vuewmrc`.

3. If you want the control to provide a visual cue when it has been activated, create a series of transition bitmaps. Create a `Drop Effects` list for the effect in `vuewmrc`. (See "Drop Effects" earlier in this chapter.)

4. Create an action in the action definition database that describes the action to be taken when the user interacts with the control. (The Functions field for a drop zone cannot use `f.exec` commands or other window manager functions.)

Creating a New Workspace Manager

For certain systems, it may be more practical to create an entirely new workspace manager rather than edit the existing one. Similarly, it may be desirable to have several different workspace managers specified in **vuewmrc**, each with a different name.

To create a new workspace manager:

1. Use the syntax described in this chapter to specify its contents. Give the panel a unique *front_panel_name*.
2. Specify which workspace manager to use. This is done using the resource:

13

```
Vuewm*frontPanel*name: front_panel_name
```

Creating a Vertical Workspace Manager

To create a vertical workspace manager, design a panel that has many short rows. Specify both the width and height for the first control in each row; all the other controls in the row will be the same height.

An example vertical workspace manager, and the resources to support it, are located in directory **/usr/lib/X11/vue/examples/Vuewm**.

Changing Control Actions

The push buttons and toggle buttons in the front panel use the action definition database to start processes. Changing the action taken when you operate these controls involves configuring the system so that a different action is invoked when the user clicks on the control.

The three controls most commonly customized are:

- `Mail`, which, starts the MAIL action when clicked.
- `Printer-Info`, which starts the PRINT action when a file is dropped on it.
- `Terminal`, which starts the TERMINAL action.

The MAIL, PRINT, and TERMINAL actions are defined in the file `/usr/lib/X11/vue/types/user-prefs.ad`.

Table 13-12.

Action (in user-prefs.ad)	Mapped to:	Defined in:
MAIL	ELM	`/usr/lib/X11/vue/types/vue.ad`
PRINT	PRINT_PR	`/usr/lib/X11/vue/types/print.ad`
TERMINAL	TERM_LOCAL	`/usr/lib/X11/vue/types/vue.ad`

There are two ways to change the action associated with the control:

- You can change the action invoked in the Functions field of the panel control. For example, you can change `f.action PRINT` to some other action.
- You can leave the front panel control unchanged, but map its action differently in the action definition database. For example, the `PRINT` action in `/usr/lib/X11/vue/types/user-prefs.ad` can be remapped from `PRINT_PR` to some other action.

Modifying the action definition database requires that you create a personal or local database. For example, the following procedure modifies the Mail control so that is runs the XYZ mailer, rather than the `elm` mailer.

- Copy the action database file **/usr/lib/X11/vue/types/user-prefs.ad** to **$HOME/.vue/types/my_prefs.ad** .
- Change the mapping for the MAILER action by editing the line in **my_prefs.ad**:

```
MAILER   * ${ACTION_L} ${ACTION_S}  MAP     ELM
```

  to:

```
MAILER   * ${ACTION_L} ${ACTION_S}  MAP     XYZ
```

- Create an action definition for XYZ. You can create the definition in **my_prefs.ad**, or in some other *filename*.**ad** file in the same directory.

13

```
need action
```

- Reread the action definition database by double-clicking the REREAD_DB action icon in the **$HOME/.vue/apps/system_apps/sys_admin** directory.
- Restart the window manager.

Example Workspace Managers

The directory **/usr/lib/X11/vue/examples/Vuewm** contains several example workspace managers and the resources needed to implement them. For additional information about the examples, read the **README** file in that directory.

14

Running Non-HP VUE Environments

This chapter covers covers:

- How to run elements of HP VUE from a startup script (rather than from the system `init` process).
- How to stop HP VUE in order to run a non-HP VUE environment (such as X11 or Softbench).

Running an HP VUE Session from a Startup Script

Running the Session Manager from a Script

Ordinarily, HP VUE is started during the system `init` process. When HP VUE is booted by the system `init` process, the system automatically runs the HP VUE Login Manager (`vuelogin`) and Session Manager (`vuesession`), as well as the Broadcast Message Server.

This section explains how to run portions of the HP VUE session manager environment from a startup script, rather than from the `init` process. When HP VUE is started this way, the login manager is not run.

There are several scenarios where this might be desirable:

- You may want most of the HP VUE environment, but do not want the save and restore capabilities of the session manager.
- You may want the user to log in at a system console and then run a script that starts the Broadcast Message server, the `vuewm` window manager, and various HP VUE clients.

Starting an HP VUE session from a startup script is accomplished using the **vuesession** client's **norestore** option. To run **vuesession** from a startup script, place the following line in the script file:

```
/usr/lib/X11/vue/etc/vuesession -norestore
```

When **vuesession** is run with this option, it does not provide the ability to save and restore previous sessions.

Use these guidelines when running **vuesession** from a startup script:

- Use the full path name in the command that starts **vuesession**.
- The only client that can be started before **vuesession** is **xrdb**. You *must* run **xrdb** before **vuesession** if **vuesession** requires resources from the database—for example, if you are using **wmStartupCommand** to start a different window manager.
- The **vuesession** client must be run in the foreground.
- You cannot run a window manager from the startup script, since **vuesession** starts the workspace manager.

14

Caution

If you run the session manager from a startup script, you must make sure the script does not contain a line that starts the window manager.

When the session manager is started this way:

- It will run the **vuewm** workspace manager. To run a different window manager:
  - Use the resource **wmStartupCommand** to specify a different window manager.
  - Use **xrdb** to read in this resource prior to running **vuesession**.
- It will run the Broadcast Message Server. You will be able to run HP VUE clients, and they will respond to color changes made in the style manager. However, fonts may not be properly set for the HP VUE clients.
- It does not run a **vue.session** file. Thus, the only clients that are started are the workspace manager and clients specified in the startup script.
- It will not load resources from **vue.resources** or **sys.resources**. Clients will get their resources from the file loaded with **xrdb** in the startup script, or from system **app-defaults** files.
- The user cannot log out from the front panel logout control. To stop the server and return to a system console, press Shift Ctrl Reset (Shift Ctrl Pause for HP C1429A Enhanced Vectra keyboards).

14

Running a Startup Script Using No-Windows Mode

Note This section applies only to HP-UX operating systems.

The "No Windows" mode option in the login screen provides a system console from which you can run a startup script. In general, you use the same guidelines as desribed in the previous section.

Keep in mind that when the startup script is run from "No Windows" mode, **vuelogin** is still running. Thus, after the user has pressed Shift Ctrl Reset (Shift Ctrl Pause for HP C1429A Enhanced Vectra keyboards) to return to the system console, executing **exit** redisplays the login screen.

Running 'vuewm' Without HP VUE

Normally, **vuewm** is configured to run in an environment running the HP VUE Broadcast Message Server. To run **vuewm** in an environment that is not using the BMS, set the following resource:

```
Vuewm*useMessaging:  False
```

15

Moving From Softbench to HP VUE

This chapter describes how configure your system to run HP VUE if you are already running Softbench. It also describes how to configure your system so that Softbench and HP VUE can coexist.

Introduction

HP VUE and Softbench are both built on the same Tool Integration Platform, which includes:

- The Broadcast Message Server (BMS). It routes message between processes. It can automatically invoke the receiver of a message if it is not yet running.
- Sub-Process Control Daemon (SPCD). It allows one process to invoke and control another process on a local or remote host. It also allows the parent process to communicate with the stdin, stdout, and stderr channels of the sub-process.

The Tool Integration Platform includes the following configuration files, located in directory `/usr/softbench/config`.

Table 15-1. Tool Integration Platform Configuration Files.

File	Contents
`softenv`	Environment variables settings for processes started through the SPCD.
`softinit`	Specifications used by the BMS for starting messaging tools.
`softtypes/C`	Filetype specifications used by the BMS with Softbench. VUE 2.0 does not use this file.

Installing HP VUE and Softbench on the Same System

When the BMS fileset is installed, any existing Tool Integration Platform configuration files are saved in the directory:

15 `/usr/softbench/oldconfig/`*release*`/usr/softbench/config`

where *release* for HP VUE 2.0 is `vue2.0_A.00.00`.

If you have customized any of the configuration files, you may want to compare your old version of the file to the new one and make any desired modifications.

Installing HP VUE and Softbench 1.1

HP VUE 2.0 and Softbench 1.1 can be installed on the same system in either order without any problems. Their `softinit` files are combined.

Installing HP VUE and Softbench 1.0

If HP VUE 2.01 and Softbench 1.0 are installed on the same system, the **softinit** file will only contain entries for the product last loaded. In order to create a **softinit** file containing entries for both products, execute the following command as super-user:

```
cat /usr/softbench/config/softinitsrc/class-defaults/vue.softinit  \
/usr/softbench/oldconfig/release/usr/softbench/config/softinit \
> /usr/softbench/config/softinit
```

(The command is broken up into multiple lines for readability. You can enter it as a single long line.)

If you reinstall HP VUE 2.01 after executing this command, the merged **softinit** file is saved in **/usr/softbench/oldconfig**, and you can merely restore it. If you reinstall Softbench 1.0 after having performed this step, you will need to rexecute the command to merge the files.

Removing Softbench 1.0

When HP VUE 2.01 and Softbench 1.0 are installed on the same system, the SBENCH fileset file lists the Tool Integration Platorm files. If you use the **sysrm** command to remove Softbench, you will also remove the Tool Integration Platform files required by HP VUE. To recover from this situation you must reinstall the BMS fileset from the HP VUE distribution.

15

Personal Customizations for Softbench

The *Installing HP Softbench* manual explains how to customize Softbench by copying `/usr/softbench/config/softinit` to `$HOME/.softinit` and modifying it. If you have created a `$HOME/.softinit` file, you must modify this file using the following steps before running HP VUE.

1. Rename `$HOME/.softinit` file to some other name.
2. Copy `/usr/softbench/config/softinit` to `$HOME/.softinit`
3. Add any modifications you had previously made to `$HOME/.softinit` to your new copy.

Any time you modify your `$HOME/.softinit` file, you must restart both HP VUE and Softbench to make use of your modifications. The best way to accomplish this is to simply log out and restart your HP VUE session.

Quitting Softbench

When you are running HP VUE, you should stop Softbench applications by
15 selecting individual applications from the Tool Manager window and using the Tool menu's "Stop" selection.

Avoid using the Tool manager's "Quit" selection in its SoftBench menu. Clicking "Quit" displays a dialog box that allows you to quit Softbench, quit the Tool Manager, or cancel the operation. Selecting "SoftBench" kills all the BMS processes and all of the Softbench tools. When HP VUE detects that the BMS has unexpectedly died, it display a dialog box warning you that there are problems and you should save any work in process and log out. When the BMS dies, the logout button on the workspace manager becomes inoperative, and you must leave the session using Shift CTRL Reset (Shift CTRL Pause for the HP C1429A Enhanced Vectra keyboard). When you exit a session this way, the session manager does not save the session. The next time you log in, the last saved session is run.

Additional Information

Additional information about the compatibility of HP VUE and Softbench is contained in `/etc/newconfig/vue/doc/vue_softb.info`.

A

Reference

This section contains reference information about the HP VUE clients. The entries are arranged alphabeticaly, each starting with its own "page 1."

HP VUE Man Pages	
Clients:	hpterm (1X) vuecommand (1X) vuefile (1X) vuehello (1X) vuehelp (1X) vuelogin (1X) vuepad (1X) vuesession (1X) vuestyle (1X) vuewm (1X) xload (1X) xterm (1X)
File Formats:	vuewmrc (4X)

NAME

hpterm - X window system Hewlett-Packard terminal emulator.

SYNOPSIS

hpterm [-toolkitoption] [-option]

DESCRIPTION

The *hpterm* program is a terminal emulator for the X Window system. It provides a Term0 compatible terminal for programs that can't use the window system directly. It also emulates many of the block mode features of HP terminals. Refer to the WARNINGS section below for additional information about running block mode applications.

OPTIONS

The *hpterm* terminal emulator accepts all of the standard X Toolkit command line options along with additional options all of which are listed below (if the option begins with a '+' instead of a '-', the option is restored to its default value):

-b *number*
This option specifies the size of the inner border (the distance between the outer edge of the character and the window border) in pixels. Associated resource: ***borderWidth**.

-background *color*
This option specifies the color to use for the background of the window. Associated resource: ***background**.

-bd *color*
This option specifies the color to use for the border of the window. Associated resource: ***borderColor**.

-bg *color*
This option specifies the color to use for the background of the window. Associated resource: ***background**.

-borderwidth *number*
This option specifies the width in pixels of the border surrounding the window. Associated resource: ***TopLevelShell.borderWidth**.

-bs This option indicates that the "background" of the term0 text entry window should be the select color that corresponds to the specified background color. Associated resource: ***backgroundIsSelect**.

+bs This option indicates that the "background" of the term0 text entry window should be the specified background. Associated resource: ***backgroundIsSelect**.

-bw *number*
This option specifies the width in pixels of the border surrounding the window. Associated resource: ***TopLevelShell.borderWidth**.

-cr *color* This option specifies the color to use for the text cursor. Associated resource: ***cursorColor**.

-display *display*
This option specifies the X server to contact; see *X*(1). Associated resource: none.

-e *command* [*arguments ...*]
This option specifies the command (and its command line arguments) to be run in the *hpterm* window. The default is to start the user's shell. **This must be the last option on the command line.** Associated resource: none.

-fb *font* This option specifies a font to be used when displaying bold (alternate) text. This font must be the same height and width as the normal (primary) font. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Refer to the NLS section. Associated resource: ***boldFont**.

-fg *color* This option specifies the color to use for displaying text. Associated resource: ***foreground**.

-fn *font* This option specifies a font to be used when displaying normal (primary) text. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Refer to the NLS section. Associated resource: ***font**.

-font *font*
This option specifies a font to be used when displaying normal (primary) text. If only one of the normal (primary) or bold (alternate) fonts is specified, it will be used for both fonts. Associated resource: ***font**.

-foreground *color*
This option specifies the color to use for displaying text. Associated resource: ***foreground**.

-geometry *geometry*
This option specifies the preferred size and position of the *hpterm* window; see *X*(1). Associated resource: ***term0.geometry**.

-help This option will display a help message. Associated resource: none.

-i This option indicates that *hpterm* should supply the window manager with a bitmapped icon. Associated resource: **bitmapIcon**.

+i This option indicates that the window manager should generate its own icon for *hpterm*. Associated resource: **bitmapIcon**.

-iconic This option indicates that *hpterm* should be placed on the display in icon form. Associated resource: ***term0.iconic**.

+iconic This option indicates that *hpterm* should not be placed on the display in icon form. Associated resource: ***term0.iconic**.

-kshmode
This option indicates that *hpterm* should convert characters entered with the extend key pressed into a two character sequence consisting of an ASCII escape followed by the unextended character. Associated resource: ***kshMode**.

-l This option indicates that *hpterm* should send all terminal output to a log file as well as to the screen. Logging may not be enabled when the -L option is used. Associated resource: ***logging**.

+l This option indicates that *hpterm* should not do logging. Associated resource: ***logging**.

-lf *file* This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is **HptermLog***XXXXX* (where *XXXXX* is the process id of *hpterm*) and is created in the directory from which *hpterm* was started. Associated resource: ***logFile**.

-ls This option indicates that the shell that is started in the *hpterm* window should be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's /etc/profile and .profile (for ksh and sh) or /etc/csh.login and .login (for csh). Associated resource: ***loginShell**.

+ls This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal "subshell"). Associated resource: ***loginShell**.

-map This option indicates that *hpterm* should map (de-iconify) itself upon pty output if it is unmapped (iconified). An initial period of time during which *hpterm* will not map itself upon pty output may be specified via the **mapOnOutputDelay** resource. Associated resource: ***mapOnOutput**.

+map This option indicates that *hpterm* should not map (de-iconify) itself upon pty output if it is unmapped (iconified). Associated resource: ***mapOnOutput**.

-mb This option indicates that the pointer cursor should be put into blanking mode. In this mode, the cursor will turn on when the pointer is moved, and will be blanked either after

a selectable number of seconds or after keyboard input has occurred. The delay is set via the **pointerBlankDelay** resource. Associated resource: ***pointerBlank.**

+mb This option indicates that the pointer cursor should remain on. Associated resource: ***pointerBlank.**

-mc *mode*
This option determines how *hpterm* will generate the foreground color, shadow colors, and shadow tiles of the scrollbar and softkey widgets. Valid modes are "all", "shadow", and "none." Associated resource: ***makeColors.**

-ms *color*
This option specifies the color to be used for the pointer cursor. Associated resource: ***pointerColor.**

-name *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name ("hpterm"). Associated resource: **.name.**

-reverse This option indicates that reverse video should be simulated by swapping the foreground and background colors. Associated resource: ***reverseVideo.**

-rv This option indicates that reverse video should be simulated by swapping the foreground and background colors. Associated resource: ***reverseVideo.**

+rv This option indicates that reverse video should not be simulated. Associated resource: ***reverseVideo.**

-sb This option indicates that a scrollbar should be displayed. Associated resource: ***scrollBar.**

+sb This option indicates that a scrollbar should not be displayed. Associated resource: ***scrollBar.**

-sbbg *color*
This option specifies the color to use for the background of the scrollbar window. Associated resource: ***scrollBar.background.**

-sbfg *color*
This option specifies the color to use for the foreground of the scrollbar window. This value will be ignored if the **makeColors** resource is set to **all.** Associated resource: ***scrollBar.foreground.**

-skbg *color*
This option specifies the color to use for the background of the softkey window. Associated resource: ***softkey.background.**

-skfg *color*
This option specifies the color to use for displaying softkey text. This value will be ignored if the **makeColors** resource is set to **all.** Associated resource: ***softkey.foreground.**

-skfn *font*
This option specifies a font to be used when displaying softkey text. Associated resource: ***softkey.font.**

-sl *number*[*suffix*]
This option indicates the number of off screen lines to be saved in the terminal buffer. If no suffix is included or the suffix is **l**, the total length of the terminal buffer will be *number* plus the length of the terminal window. If the suffix is **s** the total length of the terminal buffer will be (*number* plus one) times the length of the terminal window. Associated resource: ***saveLines.**

-ti *name* This option specifies a name for *hpterm* to use when identifying itself to application programs. Refer to the WARNINGS section for additional information about using *hpterm* with block mode applications. Associated resource: ***termId.**

-title *name*
This option specifies a window title for *hpterm*. This string may be used by the window manager when displaying the application. Associated resource: **.TopLevelShell.title**.

-tm *string*
This option specifies a string containing terminal-setting keywords and the characters to which they may be bound. Associated resource: ***ttyModes**.

-tn *name*
This option specifies a name for *hpterm* to set the **$TERM** environment variable to. Associated resource: ***termName**.

-vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed. Associated resource: ***visualBell**.

+vb This option indicates that a visual bell should not be used. Associated resource: ***visualBell**.

-xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options. Associated resource: none.

-C This option indicates that the window should receive console output. The server must be authorized to receive console output. See "XConsoles" below for additional information. Associated resource: none.

-S*ccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, and the file descriptor of the pseudoterminal's master. This allows *hpterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications such as *pam*(1). This option will only work with pty names of the form "ttyxx." For example, "-S p01" specifies "ttyp0" on file descriptor 1. Associated resource: none.

-S*pty.fd* This option specifies the unique portion of the name of a pseudoterminal to use in slave mode, and the file descriptor of the pseudoterminal's master. This allows *hpterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications such as *pam*(1). This option will work for all pty names. For example, "-S p0.1" specifies "ttyp0" on file descriptor 1 and "-S p02.13" specifies "ttyp02" on file descriptor 13. Associated resource: none.

-U Reserved for internal use.

-W Reserved for internal use.

The following command line arguments are provided for compatibility with older versions. They may not be supported in future releases as the X Toolkit provides standard options that accomplish the same task.

#geometry
This option specifies the preferred position of the icon window. It is shorthand for specifying the ***iconGeometry** resource. Associated resource: **.iconGeometry**.

-T *string* This option specifies the title for *hpterm*'s window. It is equivalent to **-title** *string*. Associated resource: **.TopLevelShell.title**.

-n *string* This option specifies the icon name for *hpterm*'s windows. It is shorthand for specifying the ***iconName** resource. Associated resource: ***iconName**.

-r This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-reversevideo** or **-rv**. Associated resource: ***reverseVideo**.

+r This option indicates that reverse video should not be simulated. It is equivalent to **+rv**. Associated resource: ***reverseVideo**.

-w *number*
This option specifies the width in pixels of the border surrounding the window. It is

equivalent to **-borderwidth** *number* or **-bw** *number*. Associated resource: ***TopLevelShell.borderWidth**.

RESOURCES

The *hpterm* window consists of a Motif shell widget which contains a form widget. The form widget contains a term0 widget, scrollbar widget, and softkey widget. Resources specific to the shell widget are:

hpterm Resource Set

Name	Class	Type	Default
borderColor	BorderColor	Pixel	black
borderWidth	BorderWidth	int	2
geometry	Geometry	string	
iconGeometry	IconGeometry	string	
name	Name	string	hpterm
title	Title	string	Terminal emulator

borderColor
This resource defines the border color of the *hpterm* window.

borderWidth
This resource specifies the width of the *hpterm* window border. This value may be modified by the window manager.

geometry
This resource specifies the preferred size and position of the *hpterm* window.

iconGeometry
This resource specifies the preferred size and position of *hpterm* when iconified. It is not necessarily obeyed by all window managers.

name This resource specifies the name of the instance of the program. It is used when extracting resources from the resource database.

title This resource specifies the window title for *hpterm*. This string may be used by the window manager when displaying this application.

term0 Resource Set

Name	Class	Type	Default
allowSendEvents	AllowSendEvents	Boolean	FALSE
background	Background	Pixel	"white"
backgroundIsSelect	BackgroundIsSelect	string	FALSE
bitmap	Bitmap	string	
bitmapIcon	BitmapIcon	Boolean	FALSE
boldFont	Font	string	see NLS below
copyLine	CopyLine	string	"right"
cursorColor	Foreground	Pixel	"black"
cut	Cut	string	"left"
flashBorder	FlashBorder	Boolean	FALSE
font	Font	string	see NLS below
foreground	Foreground	Pixel	"black"
*fn*Attribute	SoftkeyAttribute	int	2
*fn*Label	SoftkeyLabel	string	see below
*fn*String	SoftkeyString	string	see below
halfBrightInhibit	HalfBrightInhibit	Boolean	FALSE
iconic	Iconic	Boolean	FALSE
internalBorder	BorderWidth	int	2
keyboardLanguage	KeyboardLanguage	string	see NLS below
keyboardLanguageList	KeyboardLanguageList	string	see NLS below
kshMode	KshMode	Boolean	FALSE
logFile	LogFile	string	"HptermLog*XXXXX*"
logging	Logging	Boolean	FALSE
loginShell	LoginShell	Boolean	FALSE
makeColors	MakeColors	string	"none"
mapOnOutput	AutoMap	Boolean	FALSE
mapOnOutputDelay	MapDelay	int	0
paste	Paste	string	"middle"
pointerBlank	PointerBlank	Boolean	FALSE
pointerBlankDelay	PointerBlankDelay	int	3
pointerColor	Foreground	Pixel	"black"
pointerShape	PointerShape	string	"xterm"
reverseVideo	ReverseVideo	Boolean	FALSE
roman8	Roman8	Boolean	TRUE
saveLines	SaveLines	string	"1s"
scrollBar	ScrollBar	Boolean	FALSE
softkeyInitialize16	SoftkeyInitialize16	Boolean	FALSE
softkeySelect	SoftkeySelect	string	"left"
stickyNextCursor	StickyCursor	Boolean	TRUE
stickyPrevCursor	StickyCursor	Boolean	TRUE
termId	TermId	string	"X-hpterm"
termName	TermName	string	"hpterm"
ttyModes	TtyModes	string	none
visualBell	VisualBell	Boolean	FALSE

allowSendEvents
This resource defines whether synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded.

background
This resource defines the background color of the text window.

backgroundIsSelect
This resource controls the color used as the "background" of the term0 text entry window and defaults to False. When False, the background is the color specified. When True, the background is the "select color" that corresponds to the background. For visual

consistency with other Motif-based applications, set this resource to True.

bitmap This resource defines whether *hpterm* will override its built in bitmap icon with a user specified bitmap icon. If the path does not begin with a "/" or "./", it will be processed relative to "/usr/lib/X11/bitmaps".

bitmapIcon
This resource defines whether *hpterm* will supply the window manager with a bitmapped icon. The supplied bitmap may be ignored by the window manager.

boldFont
This resource defines the font used for bold (alternate) text. See "NLS" below for defaults.

copyLine
This resource defines the pointer button/modifier combination to be used to activate the CopyLine function. See "POINTER USAGE" below.

cursorColor
This resource defines the text cursor color. The pointer cursor color is defined by the **pointerColor** resource.

cut This resource defines the pointer button/modifier combination to be used to activate the Cut function. See "POINTER USAGE" below.

flashBorder
This resource defines whether *hpterm* window border will change color when the pointer cursor enters or leaves the window.

font This resource defines the font used for normal (primary) text. See the "term0.*fontLanguage* (class Term0.FontLanguage) Resource Set" table and "NLS" below for defaults.

foreground
This resource defines the foreground (text) color of the text window.

f*n*Attribute
This resource defines the softkey attribute for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey attributes can be initialized. If it false, only the first 8 softkey attributes can be initialized.

f*n*Label
This resource defines the softkey label for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey labels can be initialized. If it false, only the first 8 softkey labels can be initialized. The default labels for softkeys 1 - 8 are "f1" - "f8." The default labels for softkeys 9 - 16 are empty.

f*n*String
This resource defines the softkey string for softkey *n* (1 - 16). If "softkeyInitialize16" is true, all 16 softkey strings can be initialized. If it false, only the first 8 softkey strings can be initialized. The default strings for softkeys 1 - 8 are "<esc>p" - "<esc>w." The default labels for softkeys 9 - 16 are empty.

halfBrightInhibit
This resource defines whether half-bright enhancements will be not be generated. When true, full-bright characters will be used instead of half-bright characters.

iconic This resource defines whether *hpterm* will start up in iconic form.

internalBorder
This resource defines the number of pixels between the characters and the window border.

keyboardLanguage
This resource defines the default keyboard language *hpterm* should use. See "NLS" below for details and defaults.

keyboardLanguageList
This resource defines the list of keyboard languages that may be selected from the terminal configuration menu. See "NLS" below for detail and defaults.

kshMode
This resource defines whether *hpterm* will operate in ksh mode. In ksh mode, *hpterm* converts characters entered with the extend key pressed into a two-character sequence consisting of an ASCII escape followed by the un-extended character.

logFile This resource defines the name of the file to which a terminal session is logged. The default is "**HptermLog***XXXXX*" (where *XXXXX* is the process id of *hpterm*).

logging This resource defines whether a terminal session will be logged. It is also available at runtime via the Device Control menu. Logging may not be enabled when the "-L" option is used.

loginShell
This resource defines whether the shell to be run in the window will be started as a login shell (i.e., the first character of argv[0] will be a dash, indicating to the shell that it should read the user's /etc/profile and .profile (for ksh and sh) or /etc/csh.login and .login (for csh).

makeColors
This resource is provided for backward compatibility with older versions of hpterm; since it may not be supported in future releases, it is no longer recommended for use. This resource defines how the **bottomShadowColor**, **foreground**, and **topShadowColor** resources of the scrollbar and softkey widgets will be generated and how the **foreground** resource of the term0 widget will be generated. If the value of this resource is "all", then *hpterm* will use the value of the **background** resource of the term0 widget to generate a value for the **foreground**, and the **background** resource of the softkey and scrollbar widgets to generate values for the **bottomShadowColor**, **foreground**, and **topShadowColor** resources such that there is a 3-D look. In this case the **topShadowTile** and **bottomShadowTile** are always set to "foreground." If the **makeColors** resource value is "shadow" the **bottomShadowColor** and **topShadowColor** will be generated but **foreground** will not be generated. If the **makeColors** resource value is set to "none" then no colors will be generated.

mapOnOutput
This resource defines whether *hpterm* will map (de-iconify) itself upon pty output if it is unmapped (iconified). An initial period of time during which *hpterm* will not map itself upon pty output may be specified to allow *hpterm* to not map itself upon initial shell output. The delay is set via the **mapOnOutputDelay** resource.

mapOnOutputDelay
This resource defines the number of seconds at startup during which *hpterm* will not map (de-iconify) itself upon pty output.

paste This resource defines the pointer button/modifier combination to be used to activate the Paste function. See "POINTER USAGE" below.

pointerBlank
This resource defines whether *hpterm* will put the pointer cursor into blanking mode. In blanking mode, the pointer cursor will turn on when the pointer is moved, and will be blanked either after a selectable number of seconds or after keyboard input has occurred. The delay is set via the **pointerBlankDelay** resource.

pointerBlankDelay
This resource defines the number of seconds to wait before blanking the pointer cursor after the pointer has been moved. When set to "0", the pointer will be blanked only upon keyboard input.

pointerColor
This resource defines the pointer cursor color. The text cursor color is defined by the **cursorColor** resource.

pointerShape
This resource defines the pointer cursor shape. Valid cursor shapes may be found in the file "/usr/include/X11/cursorfont.h." Shapes are specified as the name with the leading "XC_" dropped. Valid cursor shapes include "left_ptr", "crosshair", and "xterm."

reverseVideo
This resource defines whether reverse video will be simulated by swapping the foreground and background colors.

roman8
This resource controls the mapping of keys to characters and is effective only for western european keyboards. Roman8 encoding is used when set to TRUE, ISO 8859-1 encoding is used when set to FALSE. (It is the user's responsibility to ensure that correctly encoded fonts are used; refer to the discussion on fonts in your **Using the X Window System Manual** for more information on font characteristics.)

saveLines
This resource defines the number of lines in the terminal buffer beyond the length of the window. The resource value consists of a "*number*" followed by an optional "*suffix*." If no *suffix* is included or the *suffix* is "l" the total length of the terminal buffer will be *number* plus the length of the terminal window. If the suffix is "s" the total length of the terminal buffer will be (*number* plus one) times the length of the terminal window. *Hpterm* **will try to maintain the same buffer to window ratio when the window is resized larger.**

scrollBar
This resource defines whether the scrollbar will be displayed.

softkeyInitialize16
This resource enables initialization of all 16 softkeys. If false, only the first 8 softkeys can be initialized via resources.

softkeySelect
This resource defines the pointer button/modifier combination to be used for selecting softkeys. See "POINTER USAGE" below.

stickyNextCursor
This resource defines whether the cursor should be homed when the **Next** key is pressed. When true, the cursor will be in the same screen position after the key is pressed that it was in before pressing the key. When false, the cursor will be moved to the upper left hand corner of the screen after the key is pressed.

stickyPrevCursor
This resource defines whether the cursor should be homed when the **Prev** key is pressed. When true, the cursor will be in the same screen position after the key is pressed that it was in before pressing the key. When false, the cursor will be moved to the upper left hand corner of the screen after the key is pressed.

termId This resource defines the name for *hpterm* to use when identifying itself to application programs. Refer to the WARNINGS section for additional information about using *hpterm* with block mode applications.

termName
This resource defines the string for set the "**$TERM**" environment variable.

ttyModes
This resource specifies a string containing terminal-setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and Inext. Control characters may be specified as ^char (e.g. ^c or ^u), and ^? may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *hpterm* is started.

visualBell
This resource defines whether a visible bell (i.e. flashing) should be used instead of an

audible bell when Control-G is received.

term0.*fontLanguage* (class Term0.FontLanguage) Resource Set

Name	Class	Type	Default
primary.high	FontPosition.Size	string	see NLS below
primary.medium	FontPosition.Size	string	see NLS below
primary.low	FontPosition.Size	string	see NLS below
alternate.high	FontPosition.Size	string	see NLS below
alternate.medium	FontPosition.Size	string	see NLS below
alternate.low	FontPosition.Size	string	see NLS below

***fontLanguage*.primary.high**
This resource defines the default normal (primary) font for displays with high resolution monitors. See "NLS" below for additional information.

***fontLanguage*.primary.medium**
This resource defines the default normal (primary) font for displays with medium resolution monitors. See "NLS" below for additional information.

***fontLanguage*.primary.low**
This resource defines the default normal (primary) font for displays with low resolution monitors. See "NLS" below for additional information.

***fontLanguage*.alternate.high**
This resource defines the default bold (alternate) font for displays with high resolution monitors. See "NLS" below for additional information.

***fontLanguage*.alternate.medium**
This resource defines the default bold (alternate) font for displays with medium resolution monitors. See "NLS" below for additional information.

***fontLanguage*.alternate.low**
This resource defines the default bold (alternate) font for displays with low resolution monitors. See "NLS" below for additional information.

The following resources are specified as part of the "softkey" widget (name "softkey", class "Softkey"). For example, the softkey font resource would be specified one of:

```
HPterm*softkey*font:    hp8.8x16
HPterm*Softkey*font:    hp8.8x16
*Softkey*Font:          hp8.8x16
```

Additional resources and information can be found in the **XmPrimitive**(3X) and **CORE**(3X) man pages along with additional information about the various shadow options.

Softkey Resource Set

Name	Class	Type	Default
background	Background	Pixel	"white"
bottomShadowColor	Foreground	Pixel	"black" (see below)
bottomShadowTile	BottomShadowTile	string	"foreground" (see below)
font	Font	string	(see below)
foreground	Foreground	Pixel	"black" (see below)
topShadowColor	Background	Pixel	"white" (see below)
topShadowTile	TopShadowTile	string	"50_foreground" (see below)

background
This resource defines the background color of the softkey window.

bottomShadowColor
This resource defines the color that is combined with the bottom shadow tile and foreground color to create a pixmap used to draw the bottom and right sides of the softkey borders. This may be overridden by the term0 **makeColors** resource described above.

bottomShadowTile
This resource defines the tile used in creating the pixmap used for drawing the bottom and right shadows for the softkey borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

font This resource defines the font used for softkey text. The softkey font will default to the normal (primary) font of the text window.

foreground
This resource defines the foreground (text) color of the softkey window. This may be overridden by the term0 **makeColors** resource described above.

topShadowColor
This resource defines the color that is combined with the top shadow tile and foreground color to create a pixmap used to draw the top and left sides of the softkey borders. This may be overridden by the term0 **makeColors** resource described above.

topShadowTile
This resource defines the tile used in creating the pixmap used for drawing the top and left shadows for the softkey borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

The following resources are specified as part of the "Xmscrollbar" widget (name "scrollBar", class "ScrollBar"). Some example scrollbar resources are:

```
HPterm*scrollBar*initialDelay:   10
HPterm*ScrollBar*RepeatRate:     10
*ScrollBar*Granularity:          1
hpterm*scrollBar*width:          20
```

Additional resources and information can be found in the **XmPrimitive**(3X), **XmScrollBar**(3X), **XmValuator**(3X), and **Core**(3X) man pages along with additional information about the various shadow options.

Scrollbar Resource Set (name "scrollBar", class "ScrollBar")

Name	Class	Type	Default
background	Background	Pixel	"white"
bottomShadowColor	Foreground	Pixel	"black" (see below)
bottomShadowTile	BottomShadowTile	string	"foreground" (see below)
foreground	Foreground	Pixel	"black" (see below)
granularity	Granularity	int	2
initialDelay	InitialDelay	int	500
repeatRate	RepeatRate	int	100
topShadowColor	Background	Pixel	"white" (see below)
topShadowTile	TopShadowTile	string	"50_foreground" (see below)
width	Width	int	10

background
This resource defines the background color of the scrollbar window.

bottomShadowColor
This resource defines the color that is combined with the bottom shadow tile and foreground color to create a pixmap used to draw the bottom and right sides of the scrollbar borders. This may be overridden by the term0 **makeColors** resource described above.

bottomShadowTile
This resource defines the tile used in creating the pixmap used for drawing the bottom and right shadows for the scrollbar borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

foreground
This resource defines the foreground color of the scrollbar window. This may be overridden by the term0 **makeColors** resource described above.

granularity
This resource defines the number of lines to advance the slider when the button is being held down on an arrow. The value is defined in milliseconds.

initialDelay
This resource defines the delay to wait between the time the button is held down on an arrow before the slider starts its repetitive movement. The value is defined in milliseconds.

repeatRate
This resource defines the continuous repeat rate to use to move the slider while the button is being held down on an arrow. The value is also defined in milliseconds.

topShadowColor
This resource defines the color that is combined with the top shadow tile and foreground color to create a pixmap used to draw the top and left sides of the scrollbar borders. This may be overridden by the term0 **makeColors** resource described above.

topShadowTile
This resource defines the tile used in creating the pixmap used for drawing the top and left shadows for the scrollbar borders. Valid tile names are described in **XmCreateTile**(3X). This may be overridden by the term0 **makeColors** resource described above.

width This resource defines the width of the scrollbar in pixels.

POINTER USAGE

Hpterm allows you to cut and paste text within its own or other windows. All cutting and pasting is done using the PRIMARY selection. (To maintain compatability with previous versions of hpterm (and other applications that use cut buffers), the cutting and pasting is also done to/from the first global cut buffer. When pasting, *hpterm* gets its text from the PRIMARY selection; if the PRIMARY selection is not owned, or the current owner cannot supply the data as text, *hpterm* will try to get its data from the first global cut buffer.) The PRIMARY selection will be disowned (and the selected text unhighlighted) under the following conditions:

1) the cursor is moved anywhere before the end of the selected region
2) the beginning of the selected region is scrolled off the end of the terminal buffer
3) the selected region is scrolled across the boundaries of the locked region when memory lock is enabled.

The default button assignments may be changed via various resource strings. The cut and paste functions and their default button assignments are:

Cut The left button is used to "cut" text into the cut buffer. Move the pointer to the beginning of the text to cut, press the button, move the cursor to the end of the region, and release the button. The "cut" text will not include the character currently

under the pointer.

Paste The middle button "pastes" the text from the cut buffer, inserting it as keyboard input.

CopyLine The right hand button "cuts" the text from the pointer (at button release) through the end of line (including the new line), saving it in the cut buffer, and immediately "pastes" the line, inserting it as keyboard input. This provides a history mechanism.

The copyLine, cut, and paste key functions can be configured to any button and modifier combination desired via various resources. Each assignment consists of an optional combination of modifiers ("none" or any combination of "shift", "meta", "lock", "control", "mod1", ..., "mod5" separated by blanks), followed by a "|" and the name of the button ("left", "middle", "right", "button1", ..., "button5"). For example, if it is desired for the cut function to be associated with the middle button with shift and control pressed, one could use the following resource line:

*cut: shift control | middle

For a full list of resource names, see "RESOURCES" above.

NLS

Hpterm currently supports 23 different language versions of the HP keyboard. It is possible to switch between different languages via the "terminal configuration" menu. A list of language to choose from along with their order is specified via the "keyboardLanguageList" resource. The "keyboardLanguageList" resource consists of a list of keyboard languages separated by spaces, tabs, or new lines. Valid keyboard languages may be found in the file "/usr/lib/X11/XHPlib.h." Keyboard languages are specified as the language with the leading "KB_" dropped. The default value for the "keyboardLanguageList" resource is "US_English Belgian Canada_English Danish Dutch Finnish French Canada_French Swiss_French German Swiss_German Italian Norwegian Euro_Spanish Latin_Spanish Swedish UK_English Katakana Swiss_French2 Swiss_German2 Japanese Korean S_Chinese T_Chinese."

The initial keyboard language is specified via the "keyboardLanguage" resource. If the string is NULL, the language of the server's keyboard will be used. If the keyboard language specified is not included in the keyboardLanguageList resource, the first language included in the keyboardLanguageList resource will be used. The default is to use the language of the server's keyboard.

Hpterm will try to select default fonts which match your monitor and your keyboard language. If the normal (primary) and bold (alternate) fonts are specified, they will be used. If only one is specified (via either command line options or resources), it will be used for both the normal (primary) and bold (alternate) fonts. If neither normal (primary) or bold (alternate) fonts are specified, *hpterm* trys to find them based on the default keyboard language. The default keyboard language is indicated on the "terminal configuration" menu. The built in defaults may be overridden via the "term0.*fontLanguage*" resources. *FontLanguage* varies with the keyboard language as follows.

keyboard language	*fontLanguage*
Katakana	hp_kana8
Japanese	hp_japanese
Korean	hp_korean
T_Chinese	hp_chinese_t
S_Chinese	hp_chinese_s
all others HP keyboards	hp_roman8
non HP keyboards	iso_8859_1

The default font size used will depend upon the resolution of the monitor as follows:

monitor resolution	font size
72 DPI or less	low
greater than 72 DPI and less 100 DPI	medium
100 DPI or greater	high

For example, resource specifications for the US_English, German, and Finnish keyboards would be:

```
HPterm*hp_roman8.primary.high:        *courier-medium-r-normal--14*hp-roman8
HPterm*hp_roman8.primary.medium:      *courier-medium-r-normal--12*hp-roman8
HPterm*hp_roman8.primary.low:         *courier-medium-r-normal--8*hp-roman8
HPterm*hp_roman8.alternate.high:      *courier-bold-r-normal--14*hp-roman8
HPterm*hp_roman8.alternate.medium:    *courier-bold-r-normal--12*hp-roman8
HPterm*hp_roman8.alternate.low:       *courier-bold-r-normal--8*hp-roman8
```

For the Japanese keyboard, resource specifications would be:

```
HPterm*hp_japanese.primary.high:      jpn.8x18
HPterm*hp_japanese.primary.medium:    jpn.8x18
HPterm*hp_japanese.primary.low:       jpn.8x18
HPterm*hp_japanese.alternate.high:    math.8x18
HPterm*hp_japanese.alternate.medium:  math.8x18
HPterm*hp_japanese.alternate.low:     math.8x18
```

If these fonts can not be found, the font "fixed" will be used for both the normal (primary) and bold (alternate) fonts. These resources are for font defaults only and will be ignored if either the normal (primary) or bold (alternate) fonts are specified.

Control-N will switch to the bold (alternate) font and control-O will switch back to the normal (primary) font. *Hpterm* will switch back to the normal (primary) font automatically at the beginning of each line.

XCONSOLES

It is possible to configure a system to allow redirection of console output (and input) to an *hpterm* window. If the "-C" option is used, *hpterm* will redirect console output (and input) if:

Hpterm is displaying on the local system. *Hpterm* must be running on the same system as the server that is displaying the window.

The display is authorized. The display number of the display name (see "Display Specification" in *X*(1)) must be authorized to take control of the console via the file "/usr/lib/X11/Xconsoles." The file is parsed as follows:

A '#' and all following text on a line are ignored.

Blank lines are ignored.

Leading tabs and spaces are ignored.

A number matching the display number authorizes the server to redirect console output (and input).

An asterisk ('*') matches all display numbers and authorizes the server to redirect console output (and input).

If either condition is not met, a warning will be written to stderr and console output (and input) will not be redirected.

WARNINGS

When running block mode applications, it may be necessary for *hpterm* to identify itself to application programs as some terminal other than "X-hpterm." Most applications understand the

terminal id "2392A." Newer applications also understand the terminal id "700/92" while older applications may only understand the terminal id "2622A." To set the terminal identification string, use the "-ti" command line option, the "termId" resource, or the "TermId" class.

The overflow protect mode of memory lock is not supported.

ENVIRONMENT

Hpterm sets the environment variables "$LINES" and "$COLUMNS" to the number of lines and columns of the terminal screen. It also uses and sets the environment variable "$DISPLAY" to specify its server connection. The *resize*(1) command may be used to reset "$LINES" and "$COLUMNS" after the window size has been changed.

ORIGIN

Hewlett-Packard Company ITO.

SEE ALSO

X(1), resize(1), xset(1), xterm(1), pty(4), Core(3X), XmScrollBar(3X), XmPrimitive(3X), XmCreateTile(3X), XmValuator(3X), XmArrow(3X).

NAME

vuecommand - provide execution support for Vue products.

SYNOPSIS

vuecommand [options]

OPTIONS

-mincache *n*

Specifies the number of terminal windows that are prestarted. This is also the minimum number of terminal windows to cache. The default is zero. Note that increasing this default (perhaps to two), will result in an increased login time, while improving initial terminal start up time.

-maxcache *n*

Specifies the maximum number of terminal windows to cache. The default is four.

-terminal <*terminal_emulator*>

Specifies the terminal emulator to use for the terminal caching. The default is /usr/bin/X11/hpterm.

-help Prints the usage and then exits.

DESCRIPTION

Vuecommand is an HP VUE daemon which accepts requests to invoke operating system commands and to start applications which create their own windows. The output from the commands can be displayed in several window types including a terminal emulation window and a shared output window. The commands and applications can be started on any host which is properly configured to run HP Vue.

Vuecommand uses HP's X11 terminal emulator (see hpterm (1)) to display the output for all of its non-window smart requests. For performance reasons, *vuecommand* caches the terminal emulators. A fixed number of terminal windows are "pre-started" in an unmapped condition and when needed, the terminal windows are mapped with output from the associated request. When the user closes a terminal window *vuecommand* unmaps the window and reuses it. To change the appearance or behavior of *vuecommand's* windows, set the appropriate resource for *hpterm*.

RESOURCES

Vuecommand uses the following resources:

HP VUE Vuecommand Resource Set

Name	Class	Type	Default
maxCache	MaxCache	int	4
mapWindow	MapWindow	Boolean	True
minCache	MinCache	int	0
outputOnlyIcon	OutputOnlyIcon	XmString	Output
outputOnlyName	OutputOnlyName	XmString	Output Only Window
permTerminalIcon	PermTerminalIcon	XmString	Perm-Term
permTerminalName	PermTerminalName	XmString	Permanent Terminal Window
sharedOutputIcon	SharedOutputIcon	XmString	Shared
sharedOutputName	SharedOutputName	XmString	Shared Output Window
shellTerminalIconShell	TerminalIcon	XmString	Terminal

shellTerminalName	ShellTerminalName	XmString	Terminal Window
terminal	Terminal	XmString	/usr/bin/X11/hpterm
terminalIcon	TerminalIcon	XmString	Terminal
terminalName	TerminalName	XmString	Terminal Window

maxCache
Specifies the maximum number of terminal windows to cache.

mapWindow
Specifies whether vuecommand should explicitly map the terminal windows. If this resource is set to False, the terminal emulator's "mapOnOutput" must be set to True (it is False by default). If both of these resources are False, an action may not get its' window mapped.

minCache
Specifies the number of terminal windows that are prestarted. This is also the minimum number of terminal windows to cache.

outputOnlyIcon
Specifies the icon title for output only requests.

outputOnlyName
Specifies the window title for output only requests.

permTerminalIcon
Specifies the icon title for permanent terminal requests.

permTerminalName
Specifies the window title for permanent terminal requests.

sharedOutputIcon
Specifies the icon title for shared output requests.

sharedOutputName
Specifies the window title for shared output requests.

terminal
Specifies the terminal emulator to use.

shellTerminalIcon
Specifies the icon title for shell terminal requests.

shellTerminalName
Specifies the window title for shell terminal requests.

terminalIcon
Specifies the icon title for terminal requests.

terminalName
Specifies the window title for terminal requests.

MISCELLANEOUS

Error messages produced by *vuecommand* are logged in the file $HOME/.vue/errorlog.

Login (1) cannot be run in *vuecommand's* cached terminals.

ENVIRONMENT

Vuecommand uses the $DISPLAY environment variable.

COPYRIGHT

ORIGIN

Hewlett-Packard Company ITO.

RELATED INFORMATION

X(1), vuefile(1X), hpterm(1X).

NAME

vuefile – The HP VUE File Manager.

SYNOPSIS

vuefile [options]

DESCRIPTION

The HP VUE File Manager (*vuefile*) is HP VUE's primary interface to the file system. It provides application execution and file manipulation.

The File Manager can display many main windows, each of which contains the files of a single directory. Each file is presented as a labeled icon. A menu bar provides file operations.

A direct manipulation paradigm is used to perform operations on the displayed files. Double-click performs a specified action on a file. Selection and multiple selection activate a file or a set of files. This is primarily used to indicate the active object for menu operations. Multiple selection is also used by drag for defining the set of files to be dragged. Drag is used to copy or move a file or files between File Manager windows. It is also used to drag data to another cooperating client.

OPTIONS

Vuefile defines three command-line options that can be used when *vuefile* is run.

-noview

This runs *vuefile* in "server mode," which means that no directory views are initially displayed. *Vuefile* waits for a cooperating client to tell it to display a view.

-session *session file*

This option takes the name of a session file as an additional parameter. *Vuefile* is run with the specified session file name. The session file is a file that was previously saved by *vuefile* through a session shutdown.

-dir *directory*

This option takes a single or multiple directory specification as an additional parameter. This specification is of the form hostname:path,hostname:path,... or path,path,... or any combination of the two (no spaces are allowed in the additional parameter string). This option runs *vuefile* and causes it to display a directory view for each directory specified.

Both the **-noview** and **-session** options are normally used by the session manager to start *vuefile*.

The command-line options are treated exclusively and have a priority of session, dir, noview. If more than one of the options is specified, the lower priority option(s) is(are) ignored. If *vuefile* is run with no command-line parameters, the user's current directory is displayed.

FILE TYPES and ACTIONS

Each File Manager main window displays a set of icons, each representing a single file. Each file has an associated file type that is determined by the file name: data file, executable file, or directory. The set of file types defined for the File Manager are contained in the users and system file type files.

Each file type has a set of actions defined for it. An action is an executable or application that can be run with a file of that type. To make the action readily available, each of the actions defined for a file type are displayed in an **Actions** menu pane in the File Manager's main window. The set of actions defined for the File Manager are contained in the users and system action files.

Refer to **HP Visual Environment System Administration Manual** for a description of file type and action files.

FEATURES

Opening a File

Double-clicking a file opens it, which causes the default action defined for the file type of the file to be run. By default, opening an application will execute it, opening a directory will cause the contents of the directory to be displayed, and opening a data file will cause a text editor to be run with the data file being edited.

Dragging a File

A file or set of files can be dragged to another File Manager window or to any other cooperating client. This provides for file copy, move and execution.

File Filtering

Directories can contain files of many different types and sizes. The File Manager's filtering mechanism provides the means by which the user can selectively display sets of files. The types of filtering available include: by type, by size, by modification (or creation) date.

Directory Traversal

Simple directory traversal is available by double-clicking a directory icon. This displays the contents of the directory. The File Manager provides a display mode that displays the directory hierarchy as a graph that can be easily traversed. A dialog is also available that can search the file system for a directory or directories matching user-supplied search criteria. Another dialog can be used to list directories frequently used; a double-click on a directory in the list changes to that directory.

Setting Display Preferences

Both iconic and non-iconic visual representations for file display are provided. The non-iconic form can be used to display directories with large numbers of files or where display space is at a premium.

Finding Files

Directories can be searched for a file or set of files that match search criteria. Search criteria can include file name, size and modification (or creation) date. When files are located, a File Manager window opens to display the directory that contains the file.

Smart Trash Can

The File Manager provides access to VUE's smart trash can. The user can place files into the trash can, but they are not permanently removed until the user asks that the trash can be emptied or the HP VUE session is terminated at logout. Files dragged to the trash can also be restored, but only during the current session (before logout) and before being permanently removed.

COPYRIGHT

ORIGIN

Hewlett-Packard Company ITO.

NAME

vuehello - The HP VUE Startup Transition Program.

SYNOPSIS

vuehello [options]

DESCRIPTION

Vuehello is a client that provides a transitory visual during the start up of the Hewlett-Packard Visual User Environment (HP VUE). *Vuehello* is designed to cover the interval between login (see **vuelogin(1X)**) and the start up of the window manager (see **vuewm(1X)**).

Vuehello covers the screen with a window and displays a message centered in the screen. The message consists of two parts. The first part is a short welcome string. This string may contain literal new-line characters. The second part is the content of one or more (up to five) files. Each line of text from these two parts is centered on the screen.

Vuehello will only keep the first 100 lines of text it accumulates. It does not scroll or try to prevent the text from getting clipped if the text does not fit on the screen.

Vuehello is designed to be invoked just after login. A reasonable file to contain the invocation of vuehello is /usr/lib/X11/Vuelogin/Xsession.

OPTIONS

-display *display_name*

This option names the display on which to run. The default display is taken from the $DISPLAY environment variable.

-bg *background_color*

This option specifies the background color of the window. The default value is "black" for B&W and screens with limited color (16 or less) and "cornflowerblue" for screens with more colors (more than 16). The background color can also be specified by setting the **background** (class **Background**) resource.

-fg *foreground_color*

This option specifies the foreground color (for the text). The default value is "white." The foreground color can also be specified by setting the **foreground** (class **Foreground**) resource.

-file *file_name*

This option specifies a file from which to read message text. This option may appear up to five times to include text from five different files. *Vuehello* keeps only the first 100 lines of text from the combination of the message string and files. Each line is centered on the screen. If no file option is specified, the file "/etc/copyright" is displayed. The **file** (class **File**) resource is recognized, but only one file can be specified this way.

-font *font_name*

This option specifies the font to use with which to render the text string. The default value is "system23." The equivalent resource is **font** (class **Font**).

-string *text_string*

This option specifies the message text to display. Messages with embedded spaces must be quoted. Literal new-line characters can be embedded in the text string to specify line breaks. Each line will be centered on the screen. The default text is:

Starting the
Hewlett-Packard
Visual User Environment

The message text can also be specified with the **string** (class **String**) resource.

-timeout *num_seconds*
This option specifies the timeout value in seconds. *Vuehello* sets a timer and then waits for the **ReparentNotify** event that signals that the window manager is up. If the timer expires, *vuehello* stops waiting on the window manager and terminates immediately. The default value is 240 seconds. The equivalent resource is **timeout** (class **Timeout**).

ENVIRONMENT
Vuehello may use the $DISPLAY environment variable.

FILES
/usr/lib/X11/vue/Vuelogin/Xsession
/usr/lib/X11/vue/Vuelogin/Xstartup

COPYRIGHT

ORIGIN
Hewlett-Packard Company ITO.

RELATED INFORMATION
vuelogin(1X), **vuewm(1X)**.

NAME

vuehelp – The VUE Help Manager.

SYNOPSIS

vuehelp [options]

DESCRIPTION

The HP VUE Help Manager (*vuehelp*) is an X11 Motif client that provides access to on-line help documentation. *Vuehelp* can be run standalone or started via an application. The Help Manager is used by all HP VUE Tools and is accessible by any other motif application via the Help API.

The Help Manager is based on a shared window help facility. All tools using the Help Manager will share its two main windows. As a result of this shared window facility, the Help Manager will let only one executable per display run at a time. This one executable will service all incoming help requests.

The Help Manager has two main windows: Help Index and Help Viewer. The Help Index window serves as a browser into the system's on-line help documentation. The Help Viewer window allows the user to view specific help files, system documentation, and system man pages.

Two other Help windows are also provided for static text viewing: Snapshot window and File Viewer window. The Snapshot window allows the user to copy the current help text from the Help Viewer and 'post it' on the screen in a snapshot window for prolonged viewing. The File Viewer window (called from the File Manager, *vuefile* (*1X*)), allows the user to view an ASCII text file.

OPTIONS

-fp *helpFilePath/CatFile*

This option specifies the help system to display in the Help Viewer the given help file "CatFile". "helpFilePath" must be relative to the 'tools' directory in the Help file tree.

-unmapped

This option specifies that the help system be started and initialized in an unmapped state.

RESOURCES

Vuehelp uses the following resources:

HP VUE Help Files Resource Set

Name	Class	Type	Default
fileSize	FileSize	XmInt	512000
helpFiles	HelpFiles	XmString	Default
searchPath	SearchPath	XmString	Application must Set

fileSize This resource is set to a default value of 512000 characters. The Help Manager File Viewer can deal with much larger files. However, the resources required can cause severe performance degradation while the Help Manager is processing the file.

helpFiles

This resource is used to set the absolute path to the help files located on your system. The default resource path is "/usr/lib/X11/vue/help" (default language is 'C').

searchPath

This resource must be defined in the application's app-defaults file. It is a relative path from the 'tools' directory within the Help file tree pointing to the application's help files.

HP VUE Help Fonts Resource Set

Name	Class	Type	Default
systemFont	SystemFont	FontList	Default
userFont	UserFont	FontList	Default

systemFont
This resource is used for setting text portions of the interface that are not user manipulated (e.g. labels, buttons, titles).

userFont
This resource is used for setting text portions of the interface that are manipulated by the user (e.g. Text Edit areas).

APPEARANCE

The following sections describe the basic default behavior of the different types of VUE Help system windows. The appearance and behavior of these windows will only change based on the Help text being viewed.

HELP VIEWER WINDOW

The HP VUE Help server builds and manages one Help Viewer window. This window is shared by all applications/tools using the Help system.

The first invocation of help from within a workspace causes the Help Server to map the Help Viewer window in that workspace. If the Help Viewer window is already mapped in the workspace, the Help server repaints the new help text in the window. The window is resizable by the user; however, it has a default minimum size.

The Help Viewer window contains four areas. From top to bottom they are:

Pulldown Menu Bar The menu bar contains three menu buttons (File, Browse, and Help) used for accessing the menu items available within the Help Viewer window.

Text This area contains a scrollable Help text document that relates to the previously selected function.

Related Topics This area contains a scrollable list of related topics.

History and Options This area contains two buttons and a menu. Two buttons for history manipulation and a menu to access the Word Search, Man Page Search and Glossary options.

HELP INDEX WINDOW

The Help server builds and manages one Help Index window. This window is shared by all applications/tools using help.

Invoke the Help Index from the "?" button in the workspace manager or from a Help Viewer's Browse menu. If the Help Index is already mapped in the current workspace, the Help server repaints the new help index information in the existing Help Index window. If the Help Index exists but is unmapped or in another workspace, the Help server updates the text and maps the window in the current workspace. The window is resizable by the user; however, it has a default minimum size.

The Help Index window contains three areas. From top to bottom they are:

Pulldown Menu Bar The pulldown menu contains two menu buttons (File, Help) used for accessing the menu items available within the Index Window.

Index This area contains a scrollable index of help topics and navigation buttons.

Viewer and Search This area contains two buttons: Viewer and Show Search Option. The Viewer button opens the Help Viewer window, and the Show Search Option shows the Index area search option.

HELP SNAPSHOT WINDOW

The Help Snapshot window provides the user with a means of saving a copy of the current Help Viewer help text. The user creates a snapshot from the Help Viewer window by selecting "Snapshot" (camera) button on the Help Viewer window. At any given time, a user may have numerous snapshots and one Help Viewer window.

Help Snapshot windows do not browse further into the help system, but the do support all the other Help window functions: text scrolling, word searching, and printing. The intent of the snapshot windows are to give the user the ability to compare/save multiple Help text views at the same time.

Once a Snapshot window is created, its existence is managed by the user. That is, it will be the user's responsibility to close the view. Exiting the Help Window will leave any snapshot windows intact.

The Snapshot window contains three areas. From top to bottom they are:

Text This area contains a copy of the Help text in the current Help Viewer window.

Word Search A word search mechanism provided here is the same as the word search mechanisms in the Help window. The intent is to give the user the ability to search through the Snapshot window Help text for a user-selected word or phrase. The **Find Next** and **Find Previous** buttons allow the user to search forwards and backwards within the text for the given word or phrase. As each instance of the search string is found, it is highlighted.

Command Options This area displays three buttons: a **Print** button, which prints a copy of the current Snapshot Help-Text; a **Help** button, which causes the Help Viewer window to display help information about the Snapshot Window; and a **Close** button, which closes the window.

FILE VIEWER WINDOW

The File Viewer Window provides an application with a means of viewing a text file in a window. Access the File Viewer window from the File Manager (*vuefile* (*1X*)) by choosing "VIEW" from the "Actions" menu. There is no direct way via the help system that a user can request that a file be viewed. Files may only be viewed by an application making the proper help API call. The File Viewer window looks and works just like the Snapshot windows.

Once a File Viewer is created, its existence is managed by the user. That is, it is the user's responsibility to close the view.

The File Viewer window contains three areas. From top to bottom they are:

Text This area contains the text of the file requested.

Word Search A word search mechanism provided here is the same as the word search mechanisms in the Help Window. This allows the user the ability to search through the File Viewer text for user-selected words or phrases. The **Find Next** and **Find Previous** buttons allow the user to search forwards and backwards within the text for the given word or phrase. As each instance of the search string is found, it is highlighted.

Command Options This area displays three buttons: a **Print** button, which prints a copy of the current File View Help-Text; a **Help** button, which causes the Help Window to display help information about the File Viewer window; and a **Close** button, which closes the window.

X DEFAULTS

VUE Help is configured from its resource database. This database is built from the following sources. They are listed in order of precedence, low to high:

app-defaults/Vuehelp
RESOURCE_MANAGER root window property or $HOME/.vue/vue.resources
XENVIRONMENT variable or $HOME/.vue/vue.resources

Entries in the resource database may refer to other resource files for specific types of resources.

Vuehelp is the resource class name of *vuehelp* and **vuehelp** is the resource name used by vuehelp to look up resources.

COPYRIGHT

ORIGIN

Hewlett-Packard Company ITO.

SEE ALSO

X(1), vuefile(1X).

NAME

vuelogin – The HP VUE Login Manager.

SYNOPSIS

vuelogin [-config *configuration_file*] [-daemon] [-debug *debug_level*] [-error *error_log_file*] [-nodaemon] [-resources *resource_file*] [-server *server_entry*] [-session *session_program*]

DESCRIPTION

Vuelogin manages a collection of X displays, both local and possibly remote. The emergence of X terminals guided the design of several parts of this system, along with the development of the X Consortium standard XDMCP (*X Display Manager Control Protocol*). *Vuelogin* provides services similar to those provided by *init*(1M), *getty*(1M) and *login*(1) on character terminals: prompting for login and password, authenticating the user, and running a "session."

A "session" is defined by the lifetime of a particular process; in the traditional character-based terminal world, it is the user's login shell process. In the HP VUE context, it is the HP VUE Session Manager. This is because in a windowing environment, a user's login shell process does not necessarily have any terminal-like interface with which to connect.

If the HP VUE Session Manager is not used, the typical *vuelogin* substitute is either a window manager with an exit option, or a terminal emulator running a shell, where the lifetime of the terminal emulator is the lifetime of the shell process that it is running; thus reducing the X session to an emulation of the character-based terminal session.

When the session is terminated, *vuelogin* resets the X server and (optionally) restarts the whole process.

Because *vuelogin* provides the first interface that users see, it is designed to be simple to use and easy to customize to the needs of a particular site.

OPTIONS

All options, except **-config**, specify values that can also be specified in the configuration file as resources. Typically, customization is done via the configuration file rather than command line options. The options are most useful for debugging and one-shot tests.

-config *configuration_file*
: Specifies a resource file that specifies the remaining configuration parameters. If no file is specified and the file */usr/lib/X11/vue/Vuelogin/Xconfig* exists, *vuelogin* uses it.

-daemon
: Specifies "true" as the value for the **daemonMode** resource. This makes *vuelogin* close all file descriptors, disassociate the controlling terminal and put itself in the background when it first starts up (just like the host of other daemons).

-debug *debug_level*
: Specifies the numeric value for the **debugLevel** resource. A non-zero value causes *vuelogin* to print piles of debugging statements to the terminal; it also disables the **daemonMode** resource, forcing *vuelogin* to run synchronously.

-error *error_log_file*
: Specifies the value for the **errorLogFile** resource. This file contains errors from *vuelogin* as well as anything written to *stderr* by the various scripts and programs run during the progress of the session.

-nodaemon
: Specifies "false" as the value for the **daemonMode** resource.

-resources *resource_file*
: Specifies the value for the **resources** resource. This file is loaded using *xrdb* (*1*) to specify configuration parameters for the authentication screen.

-server *server_entry*
: Specifies the value for the **servers** resource. See **servers** below for more detail.

-udpPort *port_number*
Specifies the value for the **requestPort** resource. This sets the port-number that *vuelogin* monitors for XDMCP requests. Since XDMCP uses the registered well-known udp port 177, this resource should probably not be changed except for debugging.

-session *session_program*
Specifies the value for the **session** resource. This indicates the program to run when the user has logged in as the session.

CONTROLLING THE SERVER

Vuelogin controls local servers using POSIX signals. SIGHUP is expected to reset the server, closing all client connections and performing other clean up duties. SIGTERM is expected to terminate the server. If these signals do not perform the expected actions, *vuelogin* does not perform properly.

To control remote servers not using XDMCP, *vuelogin* searches the window hierarchy on the display and uses the KillClient protocol request in an attempt to clean up the terminal for the next session. This may not actually kill all of the clients, since only those that have created windows are noticed. XDMCP provides a more sure mechanism; when *vuelogin* closes its initial connection, the session is over and the terminal is required to close all other connections.

CONTROLLING VUELOGIN

Vuelogin responds to two signals: SIGHUP and SIGTERM. When sent a SIGHUP, *vuelogin* rereads the configuration file and the file specified by the **servers** resource and determines whether entries have been added or removed. If a new entry has been added, *vuelogin* starts a session on the associated display. Entries that have been removed are disabled immediately, meaning that any session in progress is terminated without notice, and no new session is started.

When sent a SIGTERM, *vuelogin* terminates all sessions in progress and exits. This can be used when shutting down the system.

ENVIRONMENT

Vuelogin invokes the user's session with the following default environment:

DISPLAY	is set to the associated display name
EDITOR	is set to /usr/bin/vi
HOME	is set to the home directory of the user
LANG	is set to the current NLS language (if any)
LOGNAME	is set to the user name
MAIL	is set to /usr/mail/$USER
PATH	is set to the value of the **userPath** resource
USER	is set to the user name
SHELL	is set to the user's default shell (from /etc/passwd)
TERM	is set to hpterm
TZ	is set to the value of the **timeZone** resource
XAUTHORITY	may be set to an authority file

Three methods are available to modify or add to this list depending on the desired scope of the resulting environment variable.

The **environment** resource is available in the *vuelogin* configuration file to allow setting of environment variables on a global or per-display basis. Variables specified by this method are available to both the display's X server process and the user's session and override any default settings. The resource accepts a string of <name> = <value> pairs separated by at least one space or tab. The values specified must be constants because no shell is used to parse the string. See the **RESOURCES** section below for details on setting this resource.

For example:

```
Vuelogin*environment:    SB_DISPLAY_ADDR=0xB00000 \
                         WMSHMSPC=0x200000
```

Note: The environment variables LANG and TZ have their own dedicated resources in the configuration file and should not be set via **environment**.

Environment variables that require processing by a shell or are dependent on the value of another environment variable can be specified in the startup script *Xsession*. These variables are loaded into the environment of all users on the display, but not to the X server process. They override any previous settings of the same variable. The *Xsession* script accepts ksh syntax for setting environment variables.

For example.

```
MAIL=/usr/mail/$USER
```

Finally, personal environment variables can be set on a per-user basis in the script file $HOME/*.vueprofile*. *Vuelogin* accepts either sh, ksh, or csh syntax for the commands in this file. The commands should only be those that set environment variables, not any that perform terminal I/O, excepting *tset(1)* or *stty(1)*. If the first line of *.vueprofile* is #!/bin/sh, #!/bin/ksh, or #!/bin/csh, *vuelogin* uses the appropriate shell to parse *.vueprofile*. Otherwise, the user's default shell ($SHELL) is used.

INTERNATIONALIZATION

All labels and messages are localizable. The message catalog *vuelogin.cat* contains the localized representations of the default labels and messages. *Vuelogin* reads the appropriate message catalog indicated by the **LANG** environment variable and displays the localized strings. An option on the authentication screen allows the user to override the default language for the subseqent session. If the authentication screen has been localized for the selected language, it is redisplayed in that language; otherwise, it is displayed in the default language. In either case, the **LANG** environment variable is set appropriately for the resulting session.

A resource **language** is available in the *vuelogin* configuration file to change the default language for a display.

RESOURCES

The actions of *vuelogin* can be controlled through the use of various configuration files, which are in the X resource format. Some resources control the behavior of *vuelogin* in general, some can be specified for a particular display, and others control the appearance of the authentication screen. The general and display-specific resources are specified in the configuration file named by the **-config** command line option. All resources should be prepended with the application name **Vuelogin**.

Vuelogin General Resource Set

Name	Class	Type	Default
autoRescan	AutoRescan	Boolean	True
daemonMode	DaemonMode	Boolean	False
debugLevel	DebugLevel	Int	0
errorLogFile	ErrorLogFile	String	NULL
keyFile	KeyFile	String	/usr/lib/X11/vue/Vuelogin/Xkeys
lockPidFile	LockPidFile	Boolean	True
pidFile	PidFile	String	NULL
remoteAuthDir	RemoteAuthDir	String	/usr/lib/X11/vue/Vuelogin

removeDomainname	RemoveDomainname	Boolean	True
requestPort	RequestPort	Int	177
servers	Servers	String	:0 Local local /usr/bin/X11/X :0
timeZone	TimeZone	String	MST7MDT

The *vuelogin* general resources are not display-specific and are applied to all displays where appropriate.

autoRescan
This boolean controls whether *vuelogin* rescans the configuration file and server file after a session terminates and the files have changed. You can force *vuelogin* to reread these files by sending a SIGHUP to the main process.

daemonMode
Normally, *vuelogin* attempts to make itself into an unassociated daemon process. This is accomplished by forking and leaving the parent process to exit, then closing file descriptors and mangling the controlling terminal. This is inconvenient when attempting to debug *vuelogin*. Setting this resource to "false" disables **daemonMode**.

debugLevel
A non-zero value specified for this integer resource enables reams of debugging information to be printed. It also disables daemon mode, which redirects the information into the bit-bucket. Specifying a non-zero debug level also allows non-root users to run *vuelogin*, which is not normally useful.

errorLogFile
Error output is normally directed at the system console. To redirect it, set this resource to any file name. A method to send these messages to syslog should be developed for systems that support it; however, the wide variety of "standard" interfaces precludes any system-independent implementation. This file also contains any output directed to stderr by *Xstartup*, *Xsession* and *Xreset*, so it contains descriptions of problems in those scripts as well.

keyFile XDM-AUTHENTICATION-1 style XDMCP authentication requires that a private key be shared between *vuelogin* and the terminal. This resource specifies the file containing those values. Each entry in the file consists of a display name and the shared key. By default, *vuelogin* does not include support for XDM-AUTHENTICATION-1 because it requires DES, which is not generally distributable.

lockPidFile
This is the resource that controls whether *vuelogin* uses file locking to prevent multiple logins.

pidFile The filename specified is created to contain an ASCII representation of the process-id of the main *vuelogin* process. This is quite useful when reinitializing the system. *Vuelogin* also uses file locking to attempt to prevent more than one daemon running on the same machine.

remoteAuthDir
This is a directory name that *vuelogin* uses to temporarily store authorization files for displays using XDMCP.

removeDomainname
When computing the display name for XDMCP clients, the resolver typically creates a fully qualified host name for the terminal. As this is sometimes confusing, *vuelogin* removes the domain name portion of the host name if it is the same as the domain name for the local host when this variable is set.

requestPort
This indicates the UDP port number that *vuelogin* uses to listen for incoming XDMCP requests. Unless you need to debug the system, leave this with its default value.

servers This resource either specifies a file name full of server entries, one per line (if the value starts with a slash), or a single server entry. Each entry indicates a display that should constantly be managed and that is not using XDMCP. Each entry consists of at least three parts: a display name, a display class, a display type, and (for local servers) a command line to start the server. A typical entry for local display number 0 is:

:0 Local local /usr/bin/X11/X :0

The display types are:

local	a local display, i.e. one that has a server program to run
foreign	a remote display, i.e. one that has no server program to run

The display name must be something that can be passed in the **-display** option to any X program. This string is used in the display-specific resources to specify the particular display, so be careful to match the names (e.g., use ":0 local /usr/bin/X11/X :0" instead of "localhost:0 local /usr/bin/X11/X :0" if your other resources are specified as "Vuelogin._0.session"). The display class portion is also used in the display-specific resources as the class portion of the resource. This is useful if you have a large collection of similar displays (a group of X terminals, for example) and want to set resources for groups of them. When using XDMCP, the display is required to specify the display class, so perhaps your X terminal documentation describes a reasonably standard display class string for your device.

timeZone
This resource specifies the local time zone for *vuelogin*. It is loaded into the environment of *vuelogin* as the value of the variable TZ and inherited by all subsequent sessions.

Note: The TZ environment variable is often also set in the file */etc/profile* for terminal based logins. The system administrator should ensure that the two settings are identical.

Vuelogin Display Resource Set

Name	Class	Type	Default
authorize	Authorize	Boolean	False
authName	AuthName	String	MIT-MAGIC-COOKIE-1
authFile	AuthFile	String	/usr/lib/X11/vue/Vuelogin/auth-server
cpp	Cpp	String	/lib/cpp
environment	Environment	String	NULL
failsafeClient	FailsafeClient	String	/usr/bin/X11/xterm
gettyLine	GettyLine	String	console
gettySpeed	GettySpeed	String	console
grabServer	GrabServer	Boolean	True
grabTimeout	GrabTimeout	Int	3 sec.
language	Language	String	NULL
openDelay	OpenDelay	Int	5 sec.
openRepeat	OpenRepeat	Int	5 sec.
openTimeout	OpenTimeout	Int	30 sec.
pingInterval	PingInterval	Int	5 min.
pingTimeout	PingTimeout	Int	5 min.
reset	Reset	String	NULL
resetForAuth	ResetForAuth	Boolen	False
resetSignal	Signal	Int	1 (SIGHUP)
resources	Resources	String	NULL
session	Session	String	NULL

startAttempts	StartAttempts	Int	4
startup	Startup	String	NULL
systemPath	SystemPath	String	/usr/bin/X11:/bin:/usr/bin:/etc
systemShell	SystemShell	String	/bin/sh
terminateServer	TerminateServer	Boolean	False
termSignal	Signal	Int	15 (SIGTERM)
userAuthDir	UserAuthDir	String	/tmp
userPath	UserPath	String	/usr/bin/X11:/bin:/usr/bin:/usr/contrib/bin:/usr/local/b
xrdb	Xrdb	String	/usr/bin/X11/xrdb

vuelogin display resources can be specified for all displays or for a particular display. To specify a particular display, the display name is inserted into the resource name between "Vuelogin" and the final resource name segment. For example, **Vuelogin.expo_0.startup** is the name of the resource defining the startup shell file on the "expo:0" display. The resource manager separates the name of the resource from its value with colons, and separates resource name parts with dots, so *vuelogin* uses underscores for the dots and colons when generating the resource name.

Resources can also be specified for a class of displays by inserting the class name instead of a display name. A display that is not managed by XDMCP can have its class affiliation specified in the file referenced by the **servers** resource. A display using XDMCP supplies its class affiliation as part of the XDMCP packet.

authorize
authorize is a boolean resource that controls whether *vuelogin* generates and uses authorization for the server connections. (See **authName**.)

authName
If **authorize** is used, **authName** specifies the type of authorization to be used. Currently, *vuelogin* supports only MIT-MAGIC-COOKIE-1 authorization, XDM-AUTHORIZATION-1 could be supported, but DES is not generally distributable. XDMCP connections state which authorization types are supported dynamically, so **authName** is ignored in this case. (See **authorize**.)

authFile
This file is used to communicate the authorization data from *vuelogin* to the server, using the *-auth* server command line option. It should be kept in a write-protected directory to prevent its erasure, which would disable the authorization mechanism in the server.

cpp This specifies the name of the C preprocessor that is used by xrdb.

environment
This resource can contain a set of <name> = <value> pairs separated by a space or tab. Each item is loaded into the environment of the server and session. See **ENVIRONMENT** for details.

failsafeClient
If the default session fails to execute, *vuelogin* falls back to this program. This program is executed with no arguments, but executes using the same environment variables as the session would have had. (See **The Xsession File** below.)

gettyLine (See **gettySpeed**.)

gettySpeed
On local bitmaps, the user may choose a "No Windows" option via the login screen, which temporarily suspends the server and runs a standard login sequence (getty/login) on the bitmap. The user can then log in and perform non-X related tasks. When the user finishes and logs out, the server restarts, and the *vuelogin* screen is redisplayed. The **gettyLine** and **gettySpeed** resources are passed to the */etc/getty* process to indicate the desired device for the login sequence. The default value for each is "console." Another typical value for **gettyLine** is **ite[0-3]**. See *getty*(1M) for details.

grabServer (See **grabTimeout.**)

grabTimeout

To eliminate obvious security shortcomings in the X protocol, *vuelogin* grabs the server and keyboard while reading the name and password. The **grabServer** resource specifies if the server should be held while the name and password is read. When FALSE, the server is ungrabbed after the keyboard grab succeeds; otherwise, the server is grabbed until just before the session begins. The **grabTimeout** resource specifies the maximum time *vuelogin* will wait for the grab to succeed. The grab may fail if some other client has the server grabbed, or possibly if the network latencies are very high. The **grabTimeout** resource has a default of 3 seconds; be cautious when using this resource, since a user can be deceived by a look-alike window on the display. If the grab fails, *vuelogin* kills and restarts the server (if possible) and session.

language

This resource specifies the default setting for the **LANG** environment variable. If the *vuelogin* screen is localized for that language, it is displayed appropriately; otherwise, it is displayed in the language "C". The user may temporarily override this setting via an option on the login screen. When the subsequent session terminates, the **LANG** variable reverts to this setting.

openDelay (See **startAttempts.**)

openRepeat (See **startAttempts.**)

openTimeout (See **startAttempts.**)

pingInterval (See **pingTimeout.**)

pingTimeout

To discover when remote displays disappear, *vuelogin* occasionally "pings" them, using an X connection and sending XSync requests. **pingInterval** specifies the time (in minutes) between successive ping attempts, and **pingTimeout** specifies the maximum wait time (in minutes) for the terminal to respond to the request. If the terminal does not respond, the session is terminated. *Vuelogin* does not ping local displays. Although it may seem harmless, it is undesirable when the workstation session is terminated as a result of the server hanging for NFS service and not responding to the ping.

reset This specifies a program that is run (as root) after the session terminates. By default no program is run. The conventional name is *Xreset*. See **The Xreset File** below.

resetForAuth

The original implementation of authorization in the sample server reread the authorization file at server reset time, instead of when checking the initial connection. Since *vuelogin* generates the authorization information just before connecting to the display, an old server does not get current authorization information. This resource causes *vuelogin* to send SIGHUP to the server after setting up the file, causing an additional server reset to occur, during which time the new authorization information is read.

resetSignal

This resource specifies the signal to be sent to *vuelogin* to cause it to reread the configuration file and the file specified by the **servers** resource.

resources

This resource specifies the name of the file to be loaded by *xrdb* (*1*) as the resource database onto the root window of screen 0 of the display. This resource data base is loaded just before the authentication procedure is started, so it can control the appearance of the "login" window. See the section below on the authentication screen, which describes the various resources that are appropriate to place in this file. There is no default value for this resource, but the conventional name is */usr/lib/X11/vue/Vuelogin/Xresources*.

session This specifies the session to be executed (not running as root). By default, */usr/bin/X11/xterm* is run. The conventional name is *Xsession*. See **The Xsession File** below.

startAttempts
Four numeric resources control the behavior of *vuelogin* when attempting to open reluctant servers: **openDelay, openRepeat, openTimeout,** and **startAttempts. openDelay** is the duration (in seconds) between successive attempts; **openRepeat** is the number of attempts to make; **openTimeout** is the amount of time to wait while actually attempting the opening (i.e., the maximum time spent in the *connect* (2) syscall); and **startAttempts** is the number of times the entire process occurs before giving up on the server. After **openRepeat** attempts have been made, or if **openTimeout** seconds elapse in any particular attempt, *vuelogin* terminates and restarts the server, attempting to connect again. This process is repeated **startAttempts** time, at which point the display is declared dead and disabled. (See **openDelay, openRepeat,** and **openTimeout.**)

startup This specifies a program that is run (as root) after the authentication process succeeds. By default, no program is run. The conventional name for a file used here is *Xstartup*. See the **Xstartup** section below.

systemPath
Vuelogin sets the PATH environment variable for the startup and reset scripts to the value of this resource. Note the conspicuous absence of "." from this entry. This is a good practice to follow for root; it avoids many system penetration schemes.

systemShell
Vuelogin sets the SHELL environment variable for the startup and reset scripts to the value of this resource.

terminateServer
This boolean resource specifies whether the X server should be terminated when a session terminates (instead of resetting it). This option can be used when the server tends to grow without bound over time in order to limit the amount of time the server is run.

termSignal
This resource specifies the signal to be sent to *vuelogin* to cause it to terminate all sessions and exit.

userAuthDir
When *vuelogin* cannot write to the usual user authorization file ($HOME/.Xauthority), it creates a unique file name in this directory and points the environment variable XAUTHORITY at the created file.

userPath
Vuelogin sets the PATH environment variable for the session to this value. It should be a colon-separated list of directories; see *sh(1)* for a full description.

xrdb Specifies the program used to load the resources.

AUTHENTICATION SCREEN RESOURCES

The authentication screen reads a name-password pair from the keyboard. As this is a Motif toolkit client, colors, fonts and some layout options can be controlled with resources. Resources for this screen should be put into the file named by the **resources** resource.

The default logo on the authentication screen may be replaced with a bitmap of the user's choice. The following resources are available in addition to the standard Motif set in order to control positioning of the logo and the drop shadow. The resources should be prefaced with the string **Vuelogin*logo*** when specified.

Logo Resource Set

Name	Class	Type	Default
bitmapFile	BitmapFile	String	NULL

dropShadowBackground	DropShadowBackground	Pixel	Black
dropShadowForeground	DropShadowForeground	Pixel	White
dropShadowBackgroundPixmap	DropShadowBackgroundPixmap	String	25_foreground
dropShadowThickness	DropShadowThickness	Int	400
verticalOffset	VerticalOffset	Int	20
x	X	Position	-1
y	X	Position	-1

bitmapFile
Specifies the absolute path name to the bitmap file to be used for the logo.

dropShadowBackground
Specifes the background color for the drop shadow.

dropShadowForeground
Specifes the foreground color for the drop shadow.

dropShadowBackgroundPixmap
Specifes the pixmap to be used for the drop shadow. This can either be a built-in Motif pixmap or the absolute path name to a bitmap to be used as the tile for the drop shadow.

dropShadowThickness
Specifes the thickness of the drop shadow in units of 100th_millimeters.

verticalOffset
Specifes the percentage of the logo to be positioned vertically off the main matte. By default the logo is centered horizontally and positioned vertically by this amount above the matte. This resource is ignored if y is specified.

x Specifes the *x* origin for the logo in units of 100th_millimeters. This resource overrides the default horizontal centering of the logo.

y Specifes the *y* origin for the logo in units of 100th_millimeters. This resource overrides the default vertical positioning of the logo.

SESSION STARTUP

Three files are provided to assist in session startup. They can be replaced by other mechanisms via *vuelogin* resources.

The Xstartup File

This file is typically a shell script. It is run as "root" and should be very careful about security. This is the place to put commands that mount users' home directories from file servers, display the message of the day, or do other system-level functions on behalf of the user. Various environment variables are set for the use of this script:

DISPLAY	is set to the associated display name
HOME	is set to the home directory of the user
PATH	is set to the value of the **systemPath** resource
USER	is set to the user name
SHELL	is set to the value of the **systemShell** resource

No arguments of any kind are passed to the script. *Vuelogin* waits until this script exits before starting the user session. If the exit value of this script is non-zero, *vuelogin* discontinues the session immediately and starts another authentication cycle.

The Xsession File

This script reads in the user's personal environment from $HOME/*.vueprofile* and then invokes the desired session manager. It is run with the permissions of the authorized user, and has several environment variables pre-set. See the **ENVIRONMENT** section for a list of the pre-set variables.

The Xreset File

Symmetrical with *Xstartup*, this script is run after the user session has terminated. Run as root, it should probably contain commands that undo the effects of commands in *Xstartup*, such as unmounting directories from file servers. The collection of environment variables that were passed to *Xstartup* are also given to *Xreset*.

TYPICAL USAGE

Vuelogin is designed to operate in a wide variety of environments. The following setup is a good place to start, but may not be "typical" in many environments.

First off, the *vuelogin* configuration file should be set up. A good thing to do is to make a directory (ex. */usr/lib/X11/vue/Vuelogin*) that contains all of the relevant files. Here is a typical configuration file, which could be named *Xconfig*:

```
Vuelogin.servers:          /usr/lib/X11/vue/Vuelogin/Xservers
Vuelogin.errorLogFile:     /usr/lib/X11/vue/Vuelogin/Xerrors
Vuelogin.pidFile:          /usr/lib/X11/vue/Vuelogin/Xpid
Vuelogin*resources:        /usr/lib/X11/vue/Vuelogin/Xresources
Vuelogin*session:          /usr/lib/X11/vue/Vuelogin/Xsession
```

As you can see, this file simply contains references to other files. Note that some of the resources are specified with "*" separating the components. These resources can be made unique for each different display, by replacing the "*" with the display-name. See **RESOURCES** earlier for a complete discussion.

The first file */usr/lib/X11/vue/Vuelogin/Xservers* contains the list of displays to manage. Most workstations have only one display, numbered 0, so the file looks like this:

```
:0 Local local /usr/bin/X11/X :0
```

This keeps */usr/bin/X11/X* running on this display and manage a continuous cycle of sessions.

The file */usr/lib/X11/vue/Vuelogin/Xerrors* contains error messages from *vuelogin* and anything output to stderr by *Xstartup, Xsession or Xreset*. When you have trouble getting *vuelogin* working, check this file to see if *vuelogin* has any clues to the trouble. *Xerrors* can become quite large and should be trimmed periodically.

The next configuration entry, */usr/lib/X11/vue/Vuelogin/Xresources*, is loaded onto the display as a resource database using *xrdb* (*1*). As the authentication screen reads this database before starting up, it usually contains parameters for that screen.

SOME OTHER POSSIBILITIES

You can also use *vuelogin* to run a single session at a time by specifying the server on the command line:

```
vuelogin -server ":0 HP-TVRX local /usr/bin/X11/X :0"
```

Or, you might have a file server and a collection of X terminals. The configuration for this could look identical to the sample above, except the *Xservers* file might look like:

```
extol:0 HP700X foreign
exalt:0 NCD-19 foreign
explode:0 NCR-TOWERVIEW3000 foreign
```

This directs *vuelogin* to manage sessions on all three of these terminals. See "Controlling Vuelogin" above for a description of using signals to enable and disable these terminals much like

init(1M).

If you have an X terminal that supports the XDMCP protocol, an entry for that terminal in *Xservers* is not required. If you have a file server and all X terminals support XDMCP, then *Xservers* would contain no entries.

Configurations may contain combinations of local servers, X terminals without XDMCP, and X terminals with XDCMP.

COPYRIGHT

AUTHOR

Vuelogin is based on the MIT client *XDM*, authored by Keith Packard. Additional modifications were developed by Hewlett Packard.

ORIGIN

Hewlett-Packard Company ITO.

SEE ALSO

connect(2), **login(1)**, **getty(1M)**, **sh(1)**, **stty(1)**, **tset(1)**, **X(1)**, **xdm(1)**, **xinit(1M)**, **xrdb(1)**, and **XDMCP**.

NAME

vuepad - The HP VUE Text Editor .

SYNOPSIS

vuepad [*file ...*]

DESCRIPTION

Vuepad is a text editor for use within the X Window System environment. Vuepad allows the use of the mouse for moving the edit cursor and for selecting portions of text for editing operations. Vuepad also supports the VUE drag and drop, allowing the user to drag file icons from the VUE file manager onto the vuepad window for editing.

OPTIONS

The following option is available from the command line

-session *session file*

This option takes the name of a session file as an additional parameter. *Vuepad* is run with the specified session file name. The session file is a file that was previously saved by *vuepad* through a session shutdown.

RESOURCES

The following table and descriptions define resources used to specify data for *vuepad*.

Vuepad Resource Set

Name	Class	Type	Default
session	Session	String	NULL

ORIGIN

Hewlett-Packard Company ITO.

NAME

vuesession – The HP VUE Session Manager.

SYNOPSIS

vuesession [options]

DESCRIPTION

Vuesession provides session management functionality during an HP VUE session, the time from login to logout. It allows for saving a session, restoring a session, locking a session, and allocating colors for HP VUE compatible clients.

When a session is saved, client information, server settings, and resources are retained. Client information consists of all clients currently active in the session. Server settings consist of settings (such as the beeper or keyboard) that are modified by a call to the server. Resources consist of all resources currently stored on the **RESOURCE_MANAGER** property on the root window of the default screen.

The session is locked by pressing the "lock" button on the workspace manager (front panel). When the display is locked, no further input is accepted until a correct password is entered. Correct passwords include the user who locked the display, the root password for the system, and any users specified in the **keys** resource.

Color allocation provides each client with a set of colors which are used in creating visual components. These colors can then be dynamically changed by using the HP VUE Style Manager (*vuestyle(1X)*).

Vuesession is automatically invoked by the HP VUE Login Manager (*vuelogin(1X)*). If the HP VUE Login Manager is not currently being used, *vuesession* can be invoked by putting the following command in an .xsession or .x11start file:

/usr/lib/X11/vue/etc/vuesession -norestore

WARNING: This command must be the first X11 client in the file (other than xrdb). It must be run in the foreground. When using this command, no save and restore functionality is available. Any calls that start up window managers must be removed from the startup file. The default window manager can only be changed by setting the **wmStartupCommand** resource.

OPTIONS

-norestore

This option allows users to use *vuesession* in a limited way. The advantage of using this option is that *vuesession* can be started directly from an .x11start or .xsession script. See use and warning above.

CUSTOMIZATION

Vuesession's behavior can be customized through the use of the HP VUE Style Manager's startup dialog. The following is the behavior that can be customized.

HP VUE Session Manager Customization Options

Name	Default
Display Logout Confirmation	On
Set Home Session	N/A
Startup Behavior	Resume Current Session

Display Logout Confirmation

"Display Logout Confirmation" allows you to toggle the appearance of the logout confirmation box that appears after you click on the logout button located in the workspace manager, or choose the "Log out..." option available in the *vuewm* root menu. Setting the button displays the logout dialog. Clearing the button causes immediate exit with no confirmation. The default behavior is that the logout confirmation dialog will be displayed.

Set Home Session

The "Set Home Session" button allows you to save a home session for use later. The home session is a "snapshot" session that you can return to by changing logout options.

Startup Behavior

The startup behavior allows you to specify which state the HP VUE Session Manager will restore upon startup (login). The default is that it automatically restarts the state you were in at logout (Resume Current Session). The other states available are "Return to Home Session" and "Ask Me at Logout." "Return to Home Session" returns you to the last saved home session at next startup. "Ask Me at Logout" allows you to choose which behavior you would prefer at logout time. If "Return to Home Session" is chosen, a home session has to have been previously saved. Otherwise the default system session will appear at next login. Note that "Ask Me at Logout" cannot be active while "Display Logout Confirmation" is turned off.

RESOURCES

Vuesession uses the following resources.

HP VUE Session Manager Resource Set

Name	Class	Type	Default
alarmTime	AlarmTime	unsigned int	10
colorUse	ColorUse	int	dynamic*
coverScreen	CoverScreen	Boolean	False
dynamicColor	DynamicColor	Boolean	True
foregroundColor	ForegroundColor	int	dynamic*
keys	Keys	unsigned char	NULL
queryServerSettings	QueryServerSettings	Boolean	False
shadowPixmaps	ShadowPixmaps	int	dynamic*
wmStartupCommand	WmStartupCommand	executable path	NULL
writeXrdbColors	writeXrdbColors	Boolean	True

*The display type determines default.

alarmTime

This resource specifies (in seconds) the amount of time before the lock dialog is removed from the screen. When the display is locked, the pointer shows a lock cursor, and a dialog appears which asks for the user password. If no activity from the pointer or keyboard is detected for **alarmTime** seconds, the dialog is removed from the screen. The lock dialog is redisplayed as soon as a pointer or keyboard event is detected. An **alarmTime** of 0 leaves the lock dialog in place for the entire time the display is locked. The default value is 10 seconds.

colorUse

This resource specifies the number of colors to use for the user interface. Valid types are:

- **B_W** – Specifies a black and white system. The color palettes will use two color cells for the user interface. In this configuration only two color palettes are available: BlackWhite and WhiteBlack. These palettes cannot dynamically change. To change a palette, all applications using that color palette must be restarted. This resource value forces **ShadowPixmaps** to True, and **ForegroundColor** to either black or white depending on the palette chosen.
- **LOW_COLOR** – Specifies a low color system. The color palettes will have two color sets and use a maximum of 12 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.
- **MEDIUM_COLOR** – Specifies a medium color system. The color palettes will have 4 color sets and use a maximum of 22 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.

- **HIGH_COLOR** – Specifies a high color system. The color palettes will have 8 color sets and use a maximum of 42 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**. The default value for this resource will be determined by querying the X server for the number of color cells on the system.

planes	ColorUse
2-3	B_W
4	LOW_COLOR
6	MEDIUM_COLOR
8+	HIGH_COLOR

coverScreen

This resource specifies whether or not the screens of the display will be covered when the display is locked. The default value is False, which means that the screens will not be covered.

dynamicColor

This resource can have values of True or False and is used to reduce the number of color cells being used. Once a palette has been selected and the user will not be changing palettes often, **dynamicColor** can be set to False. When the session comes up next time, the color server will use Read Only color cells that can be shared by non-Motif1.1 clients, thus reducing total number of color cells used. The default value is True.

foregroundColor

This resource can have values of White, Black or Dynamic. **ForegroundColor** causes all text (foreground) to use either pixel 0 or 1 (Black or White) or to have a color cell dedicated to foreground and changes in response to the background color (Dynamic) for each ColorSet. If set to White or Black, the number of color cells used per ColorSet is reduced by 1.

keys This resource is a list of "keyholders" who have the ability to unlock the screen any time it is locked by the user. The list is a list of user id's separated by commas. For example if user kimd has the following resource active during a session:

Vuesession*keys: fredh,keith

Users fredh and keith have the ability to unlock the display when kimd locks it.

queryServerSettings

This resource specifies whether *vuesession* queries the server at logout for all its settings or saves only those settings set by using the HP VUE Style Manager. Querying the server ensures that all settings are saved; however, there is a degradation in performance when a full query is done. The default value is False, which means that the server will not be queried.

shadowPixmaps

For color systems, this resource can have a value of True or False. If True, **topShadowColor** and **bottomShadowColor** use the same pixel as background and **topShadowPixmap** and **bottomShadowPixmap** are specified instead of solid color to create the 3D look. This reduces the number of color cells per ColorSet by 2. **ShadowPixmaps** defaults to True for systems with 4 or less color planes (16 or less color cells), and False for systems with more than 4 color planes.

wmStartupCommand

This resource allows for an alternate window manager to be started at login. If this display is NULL, vuesession starts /usr/bin/X11/vuewm. An alternate startup might look like:

Vuesession*wmStartupCommand: /usr/bin/X11/mwm

Note that the command should not have any commands to a shell in it, and that it should not be surrounded by quotes. Also, if any other window manager other than *vuewm* is used, clients will be restored, but may not be restored to the correct position. By default, this resource contains a NULL value.

writeXrdbColors

This resource should only be used if you are running non HP Motif 1.1 clients that have color schemes that conflict with HP VUE colors. This has no impact on clients linked with the Motif 1.1 library. This resource specifies whether color resource information should be written out. If set to False, **background** and **foreground** resources will not be written out with the values of the current palette. This means that non HP Motif 1.1 clients will not get HP VUE colors when they are started. The default value is True.

COPYRIGHT

ORIGIN

Hewlett-Packard Comany ITO.

RELATED INFORMATION

X(1), vuestyle(1X), vuewm(1X), vuelogin(1X).

NAME

vuestyle - The HP VUE Style Manager.

SYNOPSIS

vuestyle

DESCRIPTION

Vuestyle is the HP VUE Style Manager application that provides an interactive method of customizing sessions and workspaces. *Vuestyle* starts the Style Manager window from which push buttons open dialog boxes for the following:

Audio Change the beeper's volume, tone, and duration. Setting the volume to zero turns the beeper off. The beeper sounds when the mouse button is released after moving any of the sliders.

Backdrop Choose from a list of backdrop bitmaps to change workspace background patterns. Distinguish workspaces by using a unique backdrop for each.

Color Choose from a list of color palettes to change workspace colors. Colors dynamically update on the display when a different palette is chosen. The display type (monochrome, low-, medium-, or high-color) determines the number of colors available.

Fonts Change font size to small, medium, or large. The display type, in part, determines the exact font sizes. For example, a large font size on a high-resolution display may look different from a large font size on a medium-resolution display. Font size changes are not dynamic. New font sizes are used the next time an HP VUE session starts (logout, then login) with "Resume current session" set in the **Session Startup** dialog box. New font sizes can also be used by clients started after a new font size selection if the resource **Vuestyle*writeXrdbImmediate** is set to True.

Hosts Add or remove hosts (other computers on your network) to or from a list that allows them to show information on your display. Enable or disable security with the toggle button.

Mouse Reverse mouse button order, change mouse pointer speed, or change double-click speed. A mouse picture shows a two- or a three-button mouse with the buttons numbered. The mouse pictured depends on the type of mouse attached to the computer. A toggle button reverses button order.

"Acceleration" and "Threshold" sliders define how fast the pointer moves when the mouse is moved. Changes are dynamic (except for double click) and take place after the slider is released. The "Double-Click" slider defines the time between the first and second click in a double-click action. The range is .1 to 1.0 seconds. The mouse picture is a button to test double-click response. New double-click speed is used the next time your HP VUE session starts (logout, then login) with "Resume current session" set in the **Session Startup** dialog box. New double-click speed can also be used by clients started after a new double-click selection if the resource **Vuestyle*writeXrdbImmediate** is set to True.

Keyboard Set key click volume or character auto repeat. The "Click Volume" slider ranges from zero (off) to 100 percent. There is also an auto-repeat toggle button. The selected actions take place immediately to allow testing while the dialog box is still open.

Screen Prevent bright colors from burning into your display by using screen-saver. Turn screen saver on or off and specify time out intervals. A "Time Out" slider ranges from zero (off) to 120 minutes. Toggle the "Logo" button to either display an X logo or a blank screen during a time out.

Startup Specify how your next HP VUE session starts. Radio buttons offer choices to resume the current session, to restore the home session, or to query on exit. Select "Display logout confirmation dialog box" to confirm at logout that you really want to log out. If unselected, immediate logout occurs without the dialog box query. The "Set Home

Session" push button opens an information dialog box to confirm the home session change.

RESOURCES

The following tables and descriptions define resources used to specify data for *vuestyle*.

Backdrop Resource Set

Name	Class	Type	Default
backdropDirectory	BackdropDirectory	string	/usr/lib/X11/bitmaps/Vuebackdrops

backdropDirectory

Specifies the directory location for the bitmaps listed in the backdrop dialog box.

Color Resource Set: Global (applies to all clients)

Name	Class	Type	Default
colorUse	ColorUse	int	dynamic*
dynamicColor	DynamicColor	Boolean	true
foregroundColor	ForegroundColor	int	dynamic*
paletteDirectories	PaletteDirectories	string	NULL
shadowPixmaps	ShadowPixmaps	int	dynamic*
writeXrdbColors	WriteXrdbColors	Boolean	true

*The display type determines default.

colorUse

Specifies the number of colors to use for the user interface. Valid types are:

- **B_W** – Specifies a black and white system. The color palettes will use two color cells for the user interface. In this configuration only two color palettes are available: BlackWhite and WhiteBlack. These palettes cannot dynamically change. To change a palette, all applications using that color palette must be restarted. This resource value forces **ShadowPixmaps** to True, and **ForegroundColor** to either black or white depending on the palette chosen.
- **LOW_COLOR** – Specifies a low color system. The color palettes will have two color sets and use a maximum of 12 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.
- **MEDIUM_COLOR** – Specifies a medium color system. The color palettes will have 4 color sets and use a maximum of 22 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**.
- **HIGH_COLOR** – Specifies a high color system. The color palettes will have 8 color sets and use a maximum of 42 color cells for the user interface, including black and white (color cells 0 and 1). The number of color cells can be reduced by using the resources **ShadowPixmaps** and **ForegroundColor**. The default value for this resource will be determined by querying the X server for the number of color cells on the system.

planes	ColorUse
2-3	B_W

4	LOW_COLOR
6	MEDIUM_COLOR
8+	HIGH_COLOR

dynamicColor

This resource can have values of True or False. **DynamicColor** is used to reduce the number of color cells being used. Once a palette has been selected and the user will not be changing palettes often, **DynamicColor** can be set to False. When the session comes up next time, the color server will use Read Only color cells that can be shared by non-Motif1.1 clients, thus reducing total number of color cells used.

foregroundColor

This resource can have values of White, Black or Dynamic. **ForegroundColor** causes all text (foreground) to use either pixel 0 or 1 (Black or White) or to have a color cell dedicated to foreground and changes in response to the background color (Dynamic) for each ColorSet. If set to White or Black, the number of color cells used per ColorSet is reduced by 1.

paletteDirectories

This resource specifies the directories where color palettes may be located. These palette files are used to generate the list of color palettes that a user may select from for color generation.

shadowPixmaps

For color systems, this resource can have a value of True or False. If True, **topShadowColor** and **bottomShadowColor** use the same pixel as background and **topShadowPixmap** and **bottomShadowPixmap** are specified instead of solid color to create the 3D look. This reduces the number of color cells per ColorSet by 2. **ShadowPixmaps** defaults to True for systems with 4 or less color planes (16 or less color cells), and False for systems with more than 4 color planes.

writeXrdbColors

This resource should only be used if you are running non HP Motif 1.1 clients that have color schemes that conflict with HP VUE colors. This has no impact on clients linked with the Motif 1.1 library. This resource specifies whether color resource information should be written out. If set to False, **background** and **foreground** resources will not be written out with the values of the current palette. This means that non HP Motif 1.1 clients will not get HP VUE colors when they are started. The default value is True.

Color Resource Set: Client Specific (specified on a per client basis)

Name	Class	Type	Default
primaryColorSetId	PrimaryColorSetId	int	3
secondaryColorSetId	SecondaryColorSetId	int	4

primaryColorSetId

This resource specifies the primary color for an application. The primary color is used for the main background areas of the application and all children of the main area. The value of this resource is a number from one to eight that represents a specific color set in a palette.

secondaryColorSetId

This resource specifies the secondary color for an application. The secondary color is used for the menubar and all menus and dialogs of the application. This allows dialogs on the screen to be visually associated with its parent application by matching the dialog color to the menubar. The value of this resource is a number from one to eight that represents a specific color set in a palette.

Color Resource Set: Workspace Manager

Name	Class	Type	Default
activeColorSetId	ActiveColorSetId	int	1
inactiveColorSetId	InactiveColorSetId	int	2

activeColorSetId
Specifies the active frame color for *vuewm*. The value of this resource is a number from one to eight, that represents a specific color set in a palette.

inactiveColorSetId
Specifies the inactive frame color for *vuewm*. The value of this resource is a number from one to eight that represents a specific color set in a palette.

Fonts Resource Set

Name	Class	Type	Default
largeUserFontHR	FontList	XmFontList	-*-prestige-medium-r-normal-*-160-72-*
largeUserFontMR	FontList	XmFontList	-*-prestige-medium-r-normal-*-128-*
largeSysFontHR	FontList	XmFontList	-*-swiss*742-medium-r-normal-*-140-*-p-110-*
largeSysFontMR	FontList	XmFontList	-*-swiss*742-bold-r-normal-*-140-*-p-100-*
mediumUserFontHR	FontList	XmFontList	-*-prestige-medium-r-normal*-128-*
mediumUserFontMR	FontList	XmFontList	-*-prestige-medium-r-normal*-120-*
mediumSysFontHR	FontList	XmFontList	-*-swiss*742-bold-r-normal-*-140-*-p-100-*
mediumSysFontMR	FontList	XmFontList	-*-swiss*742-bold-r-normal-*-120-*
smallUserFontHR	FontList	XmFontList	-*-prestige-medium-r-normal-*-120-*
smallUserFontMR	FontList	XmFontList	-*-courier-medium-r-*-120-*
smallSysFontHR	FontList	XmFontList	-*-swiss*742-bold-r-normal-*-120-*
smallSysFontMR	FontList	XmFontList	-*-helvetica-bold-r-normal-*-100-*

largeUserFontHR
Specifies the large font used for editable areas such as text edit fields and lists for high-resolution displays.

largeUserFontMR
Specifies the large font used for editable areas such as text edit fields and lists for medium-resolution displays.

largeSysFontHR
Specifies the large font used for label areas such as window frames, menus, and push buttons for high-resolution displays.

largeSysFontMR
Specifies the large font used for label areas such as window frames, menus, and push buttons for medium-resolution displays.

mediumUserFontHR
Specifies the medium font used for editable areas such as text edit fields and lists for high-resolution displays.

mediumUserFontMR
Specifies the medium font used for editable areas such as text edit fields and lists for medium-resolution displays.

mediumSysFontHR
Specifies the medium font used for label areas such as window frames, menus, and push buttons for high-resolution displays.

mediumSysFontMR
Specifies the medium font used for label areas such as window frames, menus, and push buttons for medium-resolution displays.

smallUserFontHR
Specifies the small font used for editable areas such as text edit fields and lists for high-resolution displays.

smallUserFontMR
Specifies the small font used for editable areas such as text edit fields and lists for medium-resolution displays.

smallSysFontHR
Specifies the small font used for label areas such as window frames, menus, and push buttons for high-resolution displays.

smallSysFontMR
Specifies the small font used for label areas such as window frames, menus, and push buttons for medium-resolution displays.

Session Behavior Resource

Name	Class	Type	Default
Vuestyle*writeXrdbImmediate	Vuestyle*writeXrdbImmediate	Boolean	False

Vuestyle*writeXrdbImmediate
Controls when new font size or new mouse double-click time resources are used. If True, new resources are used when new clients start. If False, new resources are used at the next session (after logout, with "Resume Current Session" selected in the session startup dialog box, and login).

COPYRIGHT

ORIGIN
Hewlett-Packard Company ITO.

NAME

vuewm - The HP VUE Workspace Manager.

SYNOPSIS

vuewm [options]

DESCRIPTION

Vuewm is an X11 window manager based upon the Motif Window Manager (*mwm, version 1.1*). Vuewm is an integral part of the Hewlett-Packard Visual User Environment (HP VUE). It communicates with and facilitates access to the other components in the environment.

Access to many other VUE components is provided throughout the *front panel* (workspace manager). The front panel can be configured to contain controls and indicators for tasks that you perform frequently. These controls include file management, session management, customization, printing, trash, help, or other functions that may be user defined.

Workspace window management represents a significant enhancement of mwm. Workspaces can be thought of as virtual screens. Multiple workspaces gives you a larger, virtual screen area to place your windows on.

Vuewm provides the same window management and limited session management functionality as mwm. It gives you (the user or programmer) the ability to control window size, position, state (iconic or normal), input focus ownership, etc. It also provides session management functions such as stopping a client.

OPTIONS

-display *display*

This option specifies the display to use, typically of the form **hostname:display_number.screen_number**. For example, **-display oregon:2.0** specifies the first screen of the second display of host **oregon**.

If no *screen_number* is specified, the first screen (screen "0") becomes the default, or *VUE screen*, of the display. The front panel is available on the VUE screen. See also *X(1)*.

-xrm *resourcestring*

This option specifies a resource string to use. For example, **vuewm -xrm "vuewm*workspaceList: A B C D"** specifies the resource names (and indirectly, the number) of workspaces that **vuewm** should create at runtime.

For example, **vuewm -xrm "vuewm*keyboardFocusPolicy: pointer"** specifies that **vuewm** should automatically set the input focus to the window under the pointer.

You can specify multiple resources by adding one **-xrm** option after another.

-multiscreen

This option causes vuewm to manage all the screens on the display. By default, vuewm manages only the default screen of the display. For example, **vuewm -display local:0.1 -multiscreen** instructs **vuewm** to manage all the screens on display 0 and to place the front panel on the second screen (i.e., screen "1").

-screens *name* [*name...*]

This option specifies the screens names to use for fetching screen-based resources. If -multiscreen is not specified, then the only the first name of the list of names will be used. If -multiscreen is specified, all the names will be used up to the number of screens managed. The first screen on the display (screen 0) will use the first name, the second screen will use the second name, and so on. If there are more screens to manage than names in the list, the first name on the list will be reused for the remaining screens.

If no names are specified, **vuewm** constructs the screen name(s) dynamically, based on two screen characteristics:

Screen Resolution The screen resolution may be *HighRes* (typically, 1280 by 1024 pixels), *MedRes* (typically, 1024 by 768 pixels), or *LowRes* (fewer pixels).

Screen Color Type The screen color type may be *Mono* (2-4 colors), *LowColor* (16 colors), *MedColor* (64 colors), or *HighColor* (256 colors).

For example, at run-time **vuewm** constructs the screen name *HighResHighColor* for an 8-plane (2^8, or 256-color) 1280x1024 HP 98550A display subsystem.

The screen name is important because it controls the number and names of your workspaces, your color schemes and backgrounds, your menus, your key and button bindings, and the icon box operations on a per screen basis.

The **/usr/lib/X11/app-defaults/Vuewm** application defaults file shipped with **vuewm** includes reasonable screen settings for most screen resolution and color types.

STARTUP

Normally, the session manager (vuesession) starts **vuewm** by running the executable as */usr/bin/X11/vuewm*. You can change this invocation by modifying a vuesession resource, **wmStartupCommand**.

For example, the following line from the **/usr/lib/X11/app-defaults/Vuesession** file causes **vuewm** to manage all the screens on the display:

```
Vuesession*wmStartupCommand: /usr/bin/X11/vuewm -multiscreen
```

(By default, vuewm manages only one screen.) Using command line options is one way of controlling the behavior of **vuewm**. Using *resources* is another.

APPEARANCE

The following sections describe the basic default behaviors of windows, icons, the icon box, input focus, and window stacking. The appearance and behavior of the window manager can be altered by changing the configuration of specific resources. Resources are defined under the heading "X DEFAULTS."

WINDOWS

Default vuewm window frames have distinct components with associated functions:

Title Area In addition to displaying the client's title, the title area is used to move the window. To move the window, place the pointer over the title area, press button 1 and drag the window to a new location. A wire frame is moved during the drag to indicate the new location. When the button is released, the window is moved to the new location.

Title Bar The title bar includes the title area, the minimize button, the maximize button and the window menu button.

Minimize Button To turn the window back into its icon, do a button 1 click on the minimize button (the frame box with a *small* square in it).

Maximize Button To make the window fill the screen (or enlarge to the largest size allowed by the configuration files), do a button 1 click on the maximize button (the frame box with a *large* square in it).

Window Menu Button The window menu button is the frame box with a horizontal bar in it. To pop up the window menu, press button 1. While pressing, drag the pointer on the menu to your selection, then release the button when

your selection is highlighted. Alternately, you can click button 1 to pop up the menu and keep it posted; then position the pointer and select.

Default Window Menu

Selection	Accelerator	Description
Restore	Alt+F5	Inactive (not an option for windows).
Move	Alt+F7	Allows the window to be moved with keys or mouse.
	Shift + Move Grow	Apollo keyboard.
Size	Alt+F8	Allows the window to be resized.
	Move Grow	Apollo keyboard.
Minimize	Alt+F9	Turns the window into an icon.
	Shift + Pop	Apollo keyboard.
Maximize	Alt+F10	Makes the window fill the screen.
	Ctrl + Move Grow	Apollo keyboard.
Lower	Alt+F3	Moves window to bottom of window stack.
Occupy...	Alt+O	Display the Workspace Presence dialog to control the presence of the client in workspaces.
Occupy All	Alt+Shift+O	Put the client in all workspaces.
Remove from WS	Alt+Shift+F4	Remove the client from the current workspace. (Inactive if the client is in only one workspace.)
Close	Alt+F4	Removes client from vuewm management.

Resize Border Handles To change the size of a window, move the pointer over a resize border handle (the cursor will change), press button 1, and drag the window to a new size. When the button is released, the window is resized. While dragging is being done, a rubber-band outline is displayed to indicate the new window size.

Matte An optional matte decoration can be added between the client area and the window frame. A matte is not actually part of the window frame. There is no functionality associated with a matte.

ICONS

Icons are small graphic representations of windows. A window can be minimized (iconified) using the minimize button on the window frame. Icons provide a way to reduce clutter on the screen.

Pressing mouse button 1 when the pointer is over an icon will cause the icon's window menu to pop up. Releasing the button (press + release without moving mouse = click) will cause the menu to stay posted. The menu contains the following selections:

Icon Window Menu

Selection	Accelerator	Description
Restore	Alt+F5	Opens the associated window.
	Shift+ Pop	Apollo keyboard.
Move	Alt+F7	Allows the icon to be moved with keys.
	Shift + Move Grow	Apollo keyboard.
Size	Alt+F8	Inactive (not an option for icons).
Minimize	Alt+F9	Inactive (not an option for icons).
Maximize	Alt+F10	Opens the associated window and makes it fill the screen.
	Ctrl + Pop	Apollo keyboard.
Lower	Alt+F3	Moves icon to bottom of icon stack.
Occupy...	Alt+O	Display the Workspace Presence dialog to control the presence of the client in workspaces.
Occupy All	Alt+Shift+O	Put the client in all workspaces.
Remove from WS	Alt+Shift+F4	Remove the client from the current workspace. (Inactive if the client is in only one workspace.)
Close	Alt+F4	Removes client from vuewm management.

Double-clicking button 1 on an icon normalizes the icon into its associated window. Double-clicking button 1 on the icon box's icon opens the icon box and allow access to the contained icons. (In general, double-clicking a mouse button offers a quick way to have a function performed. Another example is double-clicking button 1 with the pointer on the window menu button. This closes the window.)

ICON BOX

When icons begin to clutter the screen, they can be packed into an "icon box." (To use an icon box, vuewm must be started with the icon box configuration already set.) The icon box is a window manager window that holds client icons. There is one icon box for each workspace. Icons in the icon box can be manipulated with the mouse. The following table summarizes the behavior of this interface. Button actions apply whenever the pointer is on any part of the icon.

Button Action	Description
Button 1 click	Selects the icon.
Button 1 double click	Normalizes (opens) the associated window.
Button 1 double click	Raises an already *open* window to the top of the stack.
Button 1 drag	Moves the icon.

The window menu of the icon box differs from the window menu of a client window: The "Close" selection is replaced with the "PackIcons Alt+Shift+F7" selection. When selected, PackIcons packs the icons in the box to achieve neat rows with no empty slots.

If the front panel is used with the icon box in vuewm, the behavior of the icon box can be changed. If the **vuewmbox** control is in the front panel, closing the icon box will cause it to disappear from the workspace. Pressing on the **vuewmbox** control will cause the icon box to reappear if it is closed, or to be raised if it is open. In this way, the icon box is made to appear like other clients that are connected to controls in the front panel. (See the /usr/lib/X11/vue/Vuewm/sys.vuewmrc file for more information).

INPUT FOCUS

Vuewm supports (by default) a keyboard input focus policy of explicit selection. This means when a window is selected to get keyboard input, it continues to get keyboard input until the window is withdrawn from window management, another window is explicitly selected to get keyboard input, or the window is iconified. There are numerous resources that control the input focus. The client window with the keyboard input focus has the active window appearance with a visually distinctive window frame.

The following tables summarize the keyboard input focus selection behavior:

Button Action	Object	Function Description
Button 1 press	Window / window frame	Keyboard focus selection
Button 1 press	Icon	Keyboard focus selection

Key Action	Function Description
[Alt][Tab]	Move input focus to next window in window stack.
[Next Wndw]	Apollo keyboard.
[Alt][Shift][Tab]	Move input focus to previous window in window stack.
[Shift][Next Wndw]	Apollo keyboard.

WINDOW STACKING

The stacking order of windows may be changed as a result of setting the keyboard input focus, iconifying a window, or by doing a window manager window stacking function.

When a window is iconified, the window's icon is placed on the bottom of the stack.

The following table summarizes the default window stacking behavior of the window manager:

Key Action	Function Description
[Alt][ESC]	Put bottom window on top of stack.
[Ctrl][Pop]	Apollo keyboard.
[Alt][Shift][ESC]	Put top window on bottom of stack.
[Alt][Pop]	Apollo keyboard.
[Pop]	Apollo keyboard. Put bottom window on top of stack; put top window on bottom of stack.

A window can also be raised to the top when it gets the keyboard input focus (e.g., by doing a button 1 press on the window or by using [Alt][Tab]) if this auto-raise feature is enabled with the **focusAutoRaise** resource.

FRONT PANEL

The front panel is a central feature of the VUE environent. In it are controls that connect you with useful system functions and other VUE components. The default VUE environment comes with the pre-configured front panel described below.

In the center of the front panel is the workspace switch which you use to switch between workspaces. The top row of the front panel also includes a clock, today's date, mail, system load, printer, file management, application management, and trash. The bottom row of the front panel provides access to terminal windows, system customization, workspace renaming, screen locking, help, and session termination.

A control in the front panel is typically one of four types:

INDICATOR An indicator monitors the state of a file, typically your incoming mailbox, and changes its image based on whether the file has changed.

PUSH BUTTON A push button starts an action each time it is pushed. For example, each time you depress the terminal button, a new hpterm window will appear.

TOGGLE BUTTON A toggle button is tied to one instance of a client. If a client is not running, pressing the toggle button will start the client. If a client is running, pressing the toggle button will bring the client into the current workspace, make sure the client is in the normal (not iconic) state, and raise it to the top of the window stack.

VUEWM BUTTON A **vuewm** button has specialized behavior defined within **vuewm**. Examples include the clock and calendar readouts, the icon box button, and the workspace switch.

The front panel can be configured for custom installations. (See the /usr/lib/X11/vue/Vuewm/sys.vuewmrc file for more information).

WORKSPACES

Vuewm will create as many workspaces as are defined in the **workspaceList** resource. Workspaces may be renamed dynamically from the **Workspace Manager** dialog. This dialog can be called up either from a button on the front panel or by activating a button binding or key binding that you have defined (see **f.workspace_rename**).

Switching between workspaces is normally done by clicking on the workspace switch, which is usually located centrally in the front panel. Workspace switching can also be accomplish via key and button bindings (see **f.next_workspace** and **f.prev_workspace**).

Windows may inhabit one or many workspaces. This can be controlled on a per-window basis by calling up the **Workspace Presence** dialog. This dialog is normally posted from the window menu, but may also be posted with a button or key binding (see **f.workspace_presence**). This dialog lets you select the workspaces you want the window to be in from a list of available workspaces. It is also useful to see what workspaces a window is currently in.

DYNAMIC CUSTOMIZATION

When vuewm is running as part of HP VUE, you have access to dynamic customization of some aspect of vuewm through **vuestyle**. Vuestyle is started by pressing the vuestyle button (the artist's palette) on the front panel. Color and backdrops can be customized dynamically. Font changes will not take effect until the next session or until vuewm is restarted.

NOTE:

The dynamic customization of colors operates on a fixed set of color resources allocated specifically for HP VUE. You will not be able to dynamically customize the colors of components for which you have set color resources explicitly (see X DEFAULTS).

X DEFAULTS

Vuewm is configured from its resource database. This database is built from the following sources. They are listed in order of precedence, low to high:

app-defaults/Vuewm
RESOURCE_MANAGER root window property or $HOME/.Xdefaults
XENVIRONMENT variable or $HOME/.Xdefaults-host
vuewm **-xrm** and **-screens** command line options

Entries in the resource database may refer to other resource files for specific types of resources. These include files that contain bitmaps, fonts, and *vuewm* specific resources such as menus and behavior specifications (i.e., button and key bindings).

Vuewm is the resource class name of vuewm and **vuewm** is the resource name used by vuewm to look up resources. In the following discussion of resource specification "Vuewm" and "vuewm" can be used interchangeably. However, if conflicting resources are specified, resources specified by resrouce name ("vuewm") will take precedence over resource specified by resource class ("Vuewm").

Vuewm uses the following types of resources:

Component Appearance Resources:

These resources specify appearance attributes of window manager user interface components. They can be applied to the appearance of window manager menus, feedback windows (e.g., the window reconfiguration feedback window), client window frames, and icons. Generally, these resources are specified on a per-screen basis (the same for all workspaces in a screen).

Specific Appearance and Behavior Resources:

These resources specify vuewm appearance and behavior (e.g., window management policies). They are not set separately for different vuewm user interface components. Some of these resoures are global in scope; others may be specified on a per-screen basis.

Client Specific Resources:

These vuewm resources can be set for a particular client window or class of client windows. They specify client-specific icon and client window frame appearance and behavior.

Resource identifiers can be either a resource name (e.g., foreground) or a resource class (e.g., Foreground). If the value of a resource is a filename and if the filename is prefixed by "~/", then it is relative to the path contained in the *$HOME* environment variable (generally the user's home directory). This is the only environment variable vuewm uses directly ($XENVIRONMENT is used by the resource manager).

RESOURCE HIERARCHY

The resource hierarchy for vuewm takes into account the management of multiple screens and multiple workspaces per screen. Screen resources are subresources of vuewm. Workspace resources are subresources of each screen name. Depending on the resource being specified, you may need to add a screen or workspace component to the resource name to make sure you are being specific enough.

The syntax for a workspace specific resource is:

Vuewm*screen_name***workspace_name***resource_id*

If you only want to be screen specific, then the *workspace_name* component should be removed. Currently, the only workspace specific resources deal with the workspace backdrop.

COMPONENT APPEARANCE RESOURCES

The syntax for specifying *component appearance resources* that apply to window manager icons, menus, and client window frames is

Vuewm*resource_id*

For example, **Vuewm*foreground** is used to specify the foreground color for vuewm menus, icons, and client window frames.

The syntax for specifying *component appearance resources* that apply to a particular vuewm component is

Vuewm*[menu|icon|client|feedback]*resource_id*

If *menu* is specified, the resource is applied only to vuewm menus; if *icon* is specified, the resource is applied to icons; and if *client* is specified, the resource is applied to client window frames. For example, **Vuewm*icon*foreground** is used to specify the foreground color for vuewm icons, **Vuewm*menu*foreground** specifies the foreground color for vuewm menus, and **Vuewm*client*foreground** is used to specify the foreground color for vuewm client window frames.

The appearance of the title area of a client window frame (including window management buttons) can be separately configured. The syntax for configuring the title area of a client window frame is:

Vuewm*client*title*resource_id*

For example, **Vuewm*client*title*foreground** specifies the foreground color for the title area. Defaults for title area resources are based on the values of the corresponding client window frame resources.

The appearance of menus can be configured based on the name of the menu. The syntax for specifying menu appearance by name is:

Vuewm*menu*_menu_name_*resource_id

For example, **Vuewm*menu*my_menu*foreground** specifies the foreground color for the menu named **my_menu**.

The following *component appearance resources* that apply to all window manager parts can be specified:

Component Appearance Resources - All Window Manager Parts

Name	Class	Value Type	Default
background	Background	color	varies*
backgroundPixmap	BackgroundPixmap	string**	varies*
bottomShadowColor	Foreground	color	varies*
bottomShadowPixmap	BottomShadowPixmap	string**	varies*
fontList	FontList	string***	"fixed"
foreground	Foreground	color	varies*
saveUnder	SaveUnder	T/F	F
topShadowColor	Background	color	varies*
topShadowPixmap	TopShadowPixmap	string**	varies*

*The default is chosen based on the visual type of the screen. **Pixmap image name. See XmInstallImage(3X). ***X11 R3 Font description.

background (class **Background**)
This resource specifies the background color. Any legal X color may be specified. The default value is chosen based on the visual type of the screen.

backgroundPixmap (class **BackgroundPixmap**)
This resource specifies the background Pixmap of the vuewm decoration when the window is inactive (does not have the keyboard focus). The default value is chosen based on the visual type of the screen.

bottomShadowColor (class **Foreground**)
This resource specifies the bottom shadow color. This color is used for the lower and right bevels of the window manager decoration. Any legal X color may be specified. The default value is chosen based on the visual type of the screen.

bottomShadowPixmap (class **BottomShadowPixmap**)
This resource specifies the bottom shadow Pixmap. This Pixmap is used for the lower and right bevels of the window manager decoration. The default is chosen based on the visual type of the screen.

fontList (class **Font**)
This resource specifies the font used in the window manager decoration. The character encoding of the font should match the character encoding of the strings that are used. The default is "fixed."

foreground (class **Foreground**)
This resource specifies the foreground color. The default is chosen based on the visual type of the screen.

saveUnder (class **SaveUnder**)
This is used to indicate whether "save unders" are used for vuewm components. For this to have any effect, save unders must be implemented by the X server. If save unders are implemented, the X server will save the contents of windows obscured by windows that have the save under attribute set. If the saveUnder resource is True, vuewm will set the save under attribute on the window manager frame of any client that has it set. If saveUnder is False, save unders will not be used on any window manager frames. The default value is False.

topShadowColor (class **Background**)
This resource specifies the top shadow color. This color is used for the upper and left bevels of the window manager decoration. The default is chosen based on the visual type of the screen.

topShadowPixmap (class **TopShadowPixmap**)
This resource specifies the top shadow Pixmap. This Pixmap is used for the upper and left bevels of the window manager decoration. The default is chosen based on the visual type of the screen.

The following *component appearance resources* that apply to frame and icons can be specified:

Frame and Icon Components

Name	Class	Value Type	Default
activeBackground	Background	color	varies*
activeBackgroundPixmap	BackgroundPixmap	string**	varies*
activeBottomShadowColor	Foreground	color	varies*
activeBottomShadowPixmap	BottomShadowPixmap	string**	varies*
activeForeground	Foreground	color	varies*
activeTopShadowColor	Background	color	varies*
activeTopShadowPixmap	TopShadowPixmap	string**	varies*

*The default is chosen based on the visual type of the screen. **See XmInstallImage(3X).

activeBackground (class **Background**)
This resource specifies the background color of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeBackgroundPixmap (class **ActiveBackgroundPixmap**)
This resource specifies the background Pixmap of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeBottomShadowColor (class **Foreground**)
This resource specifies the bottom shadow color of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeBottomShadowPixmap (class **BottomShadowPixmap**)
This resource specifies the bottom shadow Pixmap of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeForeground (class **Foreground**)
This resource specifies the foreground color of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeTopShadowColor (class **Background**)
This resource specifies the top shadow color of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

activeTopShadowPixmap (class **TopShadowPixmap**)
This resource specifies the top shadow Pixmap of the vuewm decoration when the window is active (has the keyboard focus). The default is chosen based on the visual type of the screen.

GLOBAL APPEARANCE AND BEHAVIOR RESOURCES

The *specific appearance and behavior resource* have been divided according to their scope. The first set of resources can only be specified globally. The second set can be specified on a per-screen basis.

The syntax for specifying *global* appearance and behavior resources is

Vuewm***resource_id***

For example, **Vuewm*keyboardFocusPolicy** specifies the window manager policy for setting the keyboard focus to a particular client window.

The following *specific appearance and behavior resources* can be specified on a global basis only:

Global Specific Appearance and Behavior Resources

Name	Class	Value Type	Default
autoKeyFocus	AutoKeyFocus	T/F	T
autoRaiseDelay	AutoRaiseDelay	millisec	500
bitmapDirectory	BitmapDirectory	directory	/usr/include/X11/bitmaps
clientAutoPlace	ClientAutoPlace	T/F	T
colormapFocusPolicy	ColormapFocusPolicy	string	keyboard
configFile	ConfigFile	file	(see description)
deiconifyKeyFocus	DeiconifyKeyFocus	T/F	T
doubleClickTime*	DoubleClickTime	millisec.	multiClickTime
enableWarp	EnableWarp	T/F	T
enforceKeyFocus	EnforceKeyFocus	T/F	T
iconAutoPlace	IconAutoPlace	T/F	T
iconClick	IconClick	T/F	T
interactivePlacement	InteractivePlacement	T/F	F
keyboardFocusPolicy	KeyboardFocusPolicy	string	explicit
lowerOnIconify	LowerOnIconify	T/F	T
moveThreshold	MoveThreshold	pixels	4
multiScreen	MultiScreen	T/F	F
passButtons	PassButtons	T/F	F
passSelectButton	PassSelectButton	T/F	T
positionIsFrame	PositionIsFrame	T/F	T
positionOnScreen	PositionOnScreen	T/F	T
quitTimeout	QuitTimeout	millisec.	1000
raiseKeyFocus	RaiseKeyFocus	T/F	F
screens	Screens	string	varies
showFeedback	ShowFeedback	string	all
startupKeyFocus	StartupKeyFocus	T/F	T
useFrontPanel	UseFrontPanel	T/F	F
useMessaging	UseMessaging	T/F	F
wMenuButtonClick	WMenuButtonClick	T/F	T
wMenuButtonClick2	WMenuButtonClick2	T/F	T

*Included for backward compatiblity. You are encouraged to use the Xt resource **multiClickTime**.

autoKeyFocus (class **AutoKeyFocus**)
This resource is only available when the keyboard input focus policy is explicit. If autoKeyFocus is given a value of True, then when a window with the keyboard input focus is withdrawn from window management or is iconified, the focus is set to the previous window that had the focus. If the value given is False, there is no automatic setting of the keyboard input focus. The default value is True.

autoRaiseDelay (class **AutoRaiseDelay**)
This resource is only available when the focusAutoRaise resource is True and the keyboard focus policy is pointer. The autoRaiseDelay resource specifies the amount of time (in milliseconds) that vuewm will wait before raising a window after it gets the keyboard focus. The default value of this resource is 500 (ms).

clientAutoPlace (class **ClientAutoPlace**)
This resource determines the position of a window when the window has not been given a user specified position. With a value of True, windows are positioned with the top left corners of the frames offset horizontally and vertically. A value of False causes the

currently configured position of the window to be used. In either case, vuewm will attempt to place the windows totally on-screen. The default value is True.

colormapFocusPolicy (class ColormapFocusPolicy)
This resource indicates the colormap focus policy that is to be used. If the resource value is explicit then a colormap selection action is done on a client window to set the colormap focus to that window. If the value is pointer then the client window containing the pointer has the colormap focus. If the value is keyboard then the client window that has the keyboard input focus will have the colormap focus. The default value for this resource is keyboard.

configFile (class ConfigFile)
The resource value is the path name for a *vuewm resource description file*. If none is specified, then **vuewm** searches the following list in order:

```
$HOME/.vue/$LANG/vuewmrc
$HOME/.vue/vuewmrc
/usr/lib/X11/vue/Vuewm/$LANG/sys.vuewmrc
/usr/lib/X11/vue/Vuewm/sys.vuewmrc
```

Upon encountering the first such file, **vuewm** stops the search and uses it.

deiconifyKeyFocus (class DeiconifyKeyFocus)
This resource only applies when the keyboard input focus policy is explicit. If a value of True is used, a window will receive the keyboard input focus when it is normalized (deiconified). True is the default value.

iconAutoPlace (class IconAutoPlace)
This resource indicates whether icons are automatically placed on the screen by vuewm, or are placed by the user. Users may specify an initial icon position and may move icons after initial placement; however, vuewm will adjust the user-specified position to fit into an invisible grid. When icons are automatically placed, vuewm places them into the grid using a scheme set with the iconPlacement resource. If the iconAutoPlace resource has a value of True, then vuewm does automatic icon placement. A value of False allows user placement. The default value of this resource is True.

iconClick (class IconClick)
When this resource is given the value of True, the system menu is posted and left posted when an icon is clicked. The default value is True.

interactivePlacement (class InteractivePlacement)
This resource controls the initial placement of new windows on the screen. If the value is True, then the pointer shape changes before a new window is placed on the screen to indicate to the user that a position should be selected for the upper-left hand corner of the window. If the value is False, then windows are placed according to the initial window configuration attributes. The default value of this resource is False.

keyboardFocusPolicy (class KeyboardFocusPolicy)
If set to pointer, the keyboard focus policy is to have the keyboard focus set to the client window that contains the pointer (the pointer could also be in the client window decoration that vuewm adds). If set to explicit, the policy is to have the keyboard focus set to a client window when the user presses button 1 with the pointer on the client window or any part of the associated vuewm decoration. The default value for this resource is explicit.

lowerOnIconify (class LowerOnIconify)
If this resource is given the default value of True, a window's icon appears on the bottom of the window stack when the window is minimized (iconified). A value of False places the icon in the stacking order at the same place as its associated window.

moveThreshold (class MoveThreshold)
This resource is used to control the sensitivity of dragging operations that move windows

and icons. The value of this resource is the number of pixels that the locator will be moved with a button down before the move operation is initiated. This is used to prevent window/icon movement when a click or double-click is done and there is unintentional pointer movement with the button down. The default value of this resource is 4 (pixels).

multiScreen (class **MultiScreen**)
This resource, if True, enables vuewm to manage all the screens on the display. If False, vuewm will only manage a single screen on the display. The default value for this resource is False.

passButtons (class **PassButtons**)
This resource indicates whether or not button press events are passed to clients after they are used to do a window manager function in the client context. If the resource value is False, then the button press will not be passed to the client. If the value is True, the button press is passed to the client window. The window manager function is done in either case. The default value for this resource is False.

passSelectButton (class **PassSelectButton**)
This resource indicates whether or not the keyboard input focus selection button press (if keyboardFocusPolicy is explicit) is passed on to the client window or used to do a window management action associated with the window decorations. If the resource value is False then the button press will not be used for any operation other than selecting the window to be the keyboard input focus; if the value is True, the button press is passed to the client window or used to do a window management operation, if appropriate. The keyboard input focus selection is done in either case. The default value for this resource is True.

positionIsFrame (class **PositionIsFrame**)
This resource indicates how client window position information (from the WM_NORMAL_HINTS property and from configuration requests) is to be interpreted. If the resource value is True then the information is interpreted as the position of the vuewm client window frame. If the value is False then it is interpreted as being the position of the client area of the window. The default value of this resource is True.

positionOnScreen (class **PositionOnScreen**)
This resource is used to indicate that windows should initially be placed (if possible) so that they are not clipped by the edge of the screen (if the resource value is True). If a window is larger then the size of the screen then at least the upper left corner of the window will be on-screen. If the resource value is False, then windows are placed in the requested position even if totally off-screen. The default value of this resource is True.

quitTimeout (class **QuitTimeout**)
This resource specifies the amount of time (in milliseconds) that vuewm will wait for a client to update the WM_COMMAND property after vuewm has sent the WM_SAVE_YOURSELF message. This protocol will only be used for those clients that have a WM_SAVE_YOURSELF atom and no WM_DELETE_WINDOW atom in the WM_PROTOCOLS client window property. The default value of this resource is 1000 (ms). (Refer to the f.kill function for additional information.)

raiseKeyFocus (class **RaiseKeyFocus**)
This resource specifies if the keyboard focus should be given to window that has been raised with the **f.normalize_and_raise** function. If True, then the window will get the keyboard focus. The default value is False.

screens (class **Screens**)
This resource is used to provide resource names for the screens to be managed by vuewm. The value of this resource is a string containing a blank-separated list of names. If fewer names are specified than there are screens on the display, then the remaining screens will use the first name in the list.

There is no default value for this resource. The default names for screens are generated dynamically depending on the screen type. The dynamic name is a concatenation of the

following:

Name	Description
LowRes	less than 1024x768
MedRes	1024x768
HighRes	1280x1024 or better

Name	Description
Mono	less than 16 colors
LowColor	16 colors
MedColor	64 colors
HighColor	more than 64 colors

For example, a 1024x768, 256-color screen would get a default name of HighResHighColor.

showFeedback (class **ShowFeedback**)

This resource controls when feedback information is displayed. It controls both window position and size feedback during move or resize operations and initial client placement. It also controls window manager message and dialog boxes. The value for this resource is a list of names of the feedback options to be enabled; the names must be separated by a space. The names of the feedback options are shown below:

Name	Description
all	Show all feedback. (Default value.)
behavior	Confirm behavior switch.
kill	Confirm the killing of vuewm with signal.
move	Show position during move.
none	Show no feedback.
placement	Show position and size during initial placement.
quit	Confirm the quitting (f.quit_mwm) of vuewm.
resize	Show size during resize.
restart	Confirm vuewm restart.

The following command line illustrates the syntax for showFeedback:

Vuewm*showFeedback: placement resize behavior restart

This resource specification provides feedback for initial client placement and resize, and enables the dialog boxes to confirm the restart and set behavior functions. It disables feedback for the move function.

startupKeyFocus (class **StartupKeyFocus**)

This resource is only available when the keyboard input focus policy is explicit. When given the default value of True, a window gets the keyboard input focus when the window is mapped (i.e., initially managed by the window manager).

useFrontPanel (class **UseFrontPanel**)

This resource enables the display of the front panel (also called the "Workspace Manager") if True. The default value is False.

useMessaging (class **UseMessaging**)

This resource controls the connection of vuewm to the Broadcast Message Server (BMS). If set to True, vuewm will try to connect to the BMS and will participate in inter-process messaging with other HP VUE clients. The default value of this resource is False.

wMenuButtonClick (class WMenuButtonClick)
This resource indicates whether a click of the mouse when the pointer is over the window menu button will post and leave posted the system menu. If the value given this resource is True, then the menu will remain posted. True is the default value for this resource.

wMenuButtonClick2 (class WMenuButtonClick2)
When this resource is given the default value of True, a double-click action on the window menu button will do an f.kill function.

SCREEN APPEARANCE AND BEHAVIOR RESOURCES

The syntax for specifying *per-screen* appearance and behavior resources is

Vuewmscreen_name*resource_id**

For example, **Vuewm*1*keyBindings** specifies the key bindings to use for the screen "1".

Screen Specific Appearance and Behavior Resources

Name	Class	Value Type	Default
buttonBindings	ButtonBindings	string	NULL
cleanText	CleanText	T/F	T
fadeNormalIcon	FadeNormalIcon	T/F	F
frameBorderWidth	FrameBorderWidth	pixels	5
iconBoxGeometry	IconBoxGeometry	string	6x1+0-0
iconBoxName	IconBoxName	string	iconbox
iconBoxSBDisplayPolicy	IconBoxSBDisplayPolicy	string	all
iconBoxTitle	IconBoxTitle	string	Icons
iconDecoration	IconDecoration	string	varies
iconImageMaximum	IconImageMaximum	wxh	50x50
iconImageMinimum	IconImageMinimum	wxh	32x32
iconPlacement	IconPlacement	string	left bottom
iconPlacementMargin	IconPlacementMargin	pixels	varies
keyBindings	KeyBindings	string	Motif
limitResize	LimitResize	T/F	T
maximumMaximumSize	MaximumMaximumSize	wxh (pixels)	2X screen w&h
moveOpaque	MoveOpaque	T/F	F
pushButtonClickTime	PushButtonClickTime	millisec.	4 x multiClickTime
resizeBorderWidth	ResizeBorderWidth	pixels	10
resizeCursors	ResizeCursors	T/F	T
transientDecoration	TransientDecoration	string	system title
transientFunctions	TransientFunctions	string	-minimize -maximize
useBlinkingIndicator	UseBlinkingIndicator	T/F	T
useIconBox	UseIconBox	T/F	F
workspaceList	WorkspaceList	string	NULL

bitmapDirectory (class BitmapDirectory)
This resource identifies a directory to be searched for bitmaps referenced by vuewm resources. This directory is searched if a bitmap is specified without an absolute pathname. The default value for this resource is "/usr/include/X11/bitmaps".

Note that all images used in the front panel are fetched using the directory search path of **XmGetPixmap()**. By default, front panel images are found in directory **/usr/lib/X11/bitmaps/Vuewm.**

buttonBindings (class ButtonBindings)
This resource identifies the set of button bindings for window management functions. The named set of button bindings is specified in the *vuewm resource description file.* These button bindings are *merged* with the built-in default bindings. The default value for this resource is NULL (i.e., no button bindings are added to the built-in button

bindings).

cleanText (class **CleanText**)
This resource controls the display of window manager text in the client title and feedback windows. If the default value of True is used, the text is drawn with a clear (no stipple) background. This makes text easier to read on monochrome systems where a backgroundPixmap is specified. Only the stippling in the area immediately around the text is cleared. If False, the text is drawn directly on top of the existing background.

fadeNormalIcon (class **FadeNormalIcon**)
If this resource is given a value of True, an icon is grayed out whenever it has been normalized (its window has been opened). The default value is False.

frameBorderWidth (class **FrameBorderWidth**)
This resource specifies the width (in pixels) of a client window frame border without resize handles. The border width includes the 3-D shadows. The default value is 5 pixels.

iconBoxGeometry (class **IconBoxGeometry**)
This resource indicates the initial position and size of the icon box. The value of the resource is a standard window geometry string with the following syntax:

[=][*widthxheight*][{+-}*xoffset*{+-}*yoffset*]

If the offsets are not provided, the iconPlacement policy is used to determine the initial placement. The units for width and height are columns and rows. The actual screen size of the icon box window will depend on the iconImageMaximum (size) and iconDecoration resources. The default value for size is (6 * iconWidth + padding) wide by (1 * iconHeight + padding) high. The default value of the location is +0 -0.

iconBoxName (class **IconBoxName**)
This resource specifies the name that is used to look up icon box resources. The default name is iconbox.

iconBoxTitle (class **IconBoxTitle**)
This resource specifies the name that is used in the title area of the icon box frame. The default value is Icons.

iconBoxSBDisplayPolicy (class **IconBoxSBDisplayPolicy**)
This resource is used to set the scroll bar display policy for the icon box. The possible values for this resource are:

all	Display both scroll bars.
horizontal	Display only horizontal scroll bar.
vertical	Display only vertical scroll bar.

If only one scroll bar is displayed, the icons in the icon box are laid out so that all can be accessed and the scroll bar is displayed only when needed. The default value is all.

iconDecoration (class **IconDecoration**)
This resource specifies the general icon decoration. The resource value is label"(only the label part is displayed) or image (only the image part is displayed) or label image (both the label and image parts are displayed). A value of activelabel can also be specified to get a label (not truncated to the width of the icon) when the icon is selected. The default icon decoration for icon box icons is that each icon has a label part and an image part (label image). The default icon decoration for stand-alone icons is that each icon has an active label part, a label part and an image part (activelabel label image).

iconImageMaximum (class **IconImageMaximum**)
This resource specifies the maximum size of the icon *image*. The resource value is *widthxheight* (e.g., 64x64). The maximum supported size is 128x128. The default value of this resource is 50x50.

iconImageMinimum (class **IconImageMinimum**)
This resource specifies the minimum size of the icon *image*. The resource value is

widthxheight (e.g., 32x50). The minimum supported size is 16x16. The default value of this resource is 32x32.

iconPlacement (class **IconPlacement**)
This resource specifies the icon placement scheme to be used. The resource value has the following syntax:

primary_layout secondary_layout

The layout values are one of the following:

top	Lay the icons out top to bottom.
bottom	Lay the icons out bottom to top.
left	Lay the icons out left to right.
right	Lay the icons out right to left.

A horizontal (vertical) layout value should not be used for both the *primary_layout* and the *secondary_layout* (e.g., don't use top for the *primary_layout* and bottom for the *secondary_layout*). The *primary_layout* indicates whether, when an icon placement is done, the icon is placed in a row or a column and the direction of placement. The *secondary_layout* indicates where to place new rows or columns. For example, top right indicates that icons should be placed top to bottom on the screen and that columns should be added from right to left on the screen. The default placement is left bottom (icons are placed left to right on the screen, with the first row on the bottom of the screen, and new rows added from the bottom of the screen to the top of the screen).

iconPlacementMargin (class **IconPlacementMargin**)
This resource sets the distance between the edge of the screen and the icons that are placed along the edge of the screen. The value should be greater than or equal to 0. A default value (see below) is used if the value specified is invalid. The default value for this resource is equal to the space between icons as they are placed on the screen (this space is based on maximizing the number of icons in each row and column).

keyBindings (class **KeyBindings**)
This resource identifies the set of key bindings for window management functions. If specified these key bindings *replace* the built-in default bindings. The named set of key bindings is specified in *vuewm resource description file*. The default value for this resource is the set of compatible key bindings.

limitResize (class **LimitResize**)
If this resource is True, the user is not allowed to resize a window to greater than the maximum size. The default value for this resource is True.

maximumMaximumSize (class **MaximumMaximumSize**)
This resource is used to limit the maximum size of a client window as set by the user or client. The resource value is *widthxheight* (e.g., 1024x1024) where the width and height are in pixels. The default value of this resource is twice the screen width and height.

moveOpaque (class **MoveOpaque**)
This resource is used to set the style of feedback provided during window moves. If True, the entire window is moved as feedback (at the cost of some degradation in performance). If False, only a window outline is moved. The default value of this resource is False.

pushButtonClickTime (class **PushButtonClickTime**)
This resource controls the amount of time (in milliseconds) that pushbuttons in the front panel will be disabled after their initial press. The default value of this resource is 4 times the multiClickTime set for the window manager, or 2 seconds of wait time. The reason for the delay is to suppress multiple application invocation from users doubleclicking on pushbuttons.

resizeBorderWidth (class **ResizeBorderWidth**)
This resource specifies the width (in pixels) of a client window frame border with resize handles. The specified border width includes the 3-D shadows. The default is 10

(pixels).

resizeCursors (class **ResizeCursors**)
This is used to indicate whether the resize cursors are always displayed when the pointer is in the window size border. If True the cursors are shown, otherwise the window manager cursor is shown. The default value is True.

transientDecoration (class **TransientDecoration**)
This controls the amount of decoration that Vuewm puts on transient windows. The decoration specification is exactly the same as for the **clientDecoration** (client specific) resource. Transient windows are identified by the WM_TRANSIENT_FOR property which is added by the client to indicate a relatively temporary window. The default value for this resource is menu title (i.e., transient windows will have resize borders and a titlebar with a window menu button).

transientFunctions (class **TransientFunctions**)
This resource is used to indicate which window management functions are applicable (or not applicable) to transient windows. The function specification is exactly the same as for the **clientFunctions** (client specific) resource. The default value for this resource is -minimize -maximize.

useBlinkingIndicator (class **UseBlinkingIndicator**)
This resource enables the operation of the blinking activity light in the front panel. If True, the blinking activity light will flash if configured into the front panel. If False, the light will not flash; instead, the second bitmap specified for the **vuewmbusy** control will be installed in the front panel for the wait duration. The default value is True.

useIconBox (class **UseIconBox**)
If this resource is given a value of True, icons are placed in an icon box. When an icon box is not used, the icons are placed on the root window (default value).

workspaceList (class **WorkspaceList**)
This resource specifies the resource names and number of the workspaces to create for a screen. The value of this resource is a list of names separated by blanks. The names specified in this resource are used to fetch resources with. Consequently, a workspace name should not contain any of the following characters: asterisk (*), period (.), colon (:), quote ("), or backslash (\).

Example:

Vuewm*0*workspaceList: A B C D E F

This creates six workspaces for screen 0.

CLIENT SPECIFIC RESOURCES

The syntax for specifying *client specific resources* is

Vuewm[*screen_name***]*****client_name_or_class*****resource_id**

For example, **Vuewm*mterm*windowMenu** is used to specify the window menu to be used with mterm clients.

The syntax for specifying *client specific resources* for all classes of clients is

Vuewm***resource_id**

Specific client specifications take precedence over the specifications for all clients. For example, **Vuewm*windowMenu** is used to specify the window menu to be used for all classes of clients that don't have a window menu specified.

The syntax for specifying resource values for windows that have an unknown name and class (i.e.

the window does not have a WM_CLASS property associated with it) is

Vuewm*defaults*_resource_id_

For example, **Vuewm*defaults*iconImage** is used to specify the icon image to be used for windows that have an unknown name and class.

The following *client specific resources* can be specified:

Client Specific Resources

Name	Class	Value Type	Default
clientDecoration	ClientDecoration	string	all
clientFunctions	ClientFunctions	string	all
focusAutoRaise	FocusAutoRaise	T/F	T
iconImage	IconImage	pathname	(image)
iconImageBackground	Background	color	icon background
iconImageBottomShadowColor	Foreground	color	icon bottom shadow
iconImageBottomShadowPixmap	BottomShadowPixmap	string*	icon bottom shadow pixmap
iconImageForeground	Foreground	color	icon foreground
iconImageTopShadowColor	Background	color	icon top shadow color
iconImageTopShadowPixmap	TopShadowPixmap	string*	icon top shadow pixmap
matteBackground	Background	color	background
matteBottomShadowColor	Foreground	color	bottom shadow color
matteBottomShadowPixmap	BottomShadowPixmap	string*	bottom shadow pixmap
matteForeground	Foreground	color	foreground
matteTopShadowColor	Background	color	top shadow color
matteTopShadowPixmap	TopShadowPixmap	string*	top shadow pixmap
matteWidth	MatteWidth	pixels	0
maximumClientSize	MaximumClientSize	wxh	fill the screen
useClientIcon	UseClientIcon	T/F	F
windowMenu	WindowMenu	string	string

*See XmInstallImage(3X).

clientDecoration (class **ClientDecoration**)

This resource controls the amount of window frame decoration. The resource is specified as a list of decorations to specify their inclusion in the frame. If a decoration is preceded by a minus sign, then that decoration is excluded from the frame. The *sign* of the first item in the list determines the initial amount of decoration. If the sign of the first decoration is minus, then vuewm assumes all decorations are present and starts subtracting from that set. If the sign of the first decoration is plus (or not specified), then vuewm starts with no decoration and builds up a list from the resource.

Name	Description
all	Include all decorations (default value).
border	Window border.
maximize	Maximize button (includes title bar).
minimize	Minimize button (includes title bar).
none	No decorations.
resizeh	Border resize handles (includes border).
menu	Window menu button (includes title bar).
title	Title bar (includes border).

Examples:

Vuewm*XClock*clientDecoration: -resizeh -maximize

This removes the resize handles and maximize button from XClock windows.

Vuewm*XClock*clientDecoration: menu minimize border

This does the same thing as above. Note that either **menu** or **minimize** implies **title**.

clientFunctions (class **ClientFunctions**)

This resource is used to indicate which vuewm functions are applicable (or not applicable) to the client window. The value for the resource is a list of functions. If the first function in the list has a minus sign in front of it, then vuewm starts with all functions and subtracts from that set. If the first function in the list has a plus sign in front of it, then vuewm starts with no functions and builds up a list. Each function in the list must be preceded by the appropriate plus or minus sign and be separated from the next function by a space.

The table below lists the functions available for this resource:

Name	Description
all	Include all functions (default value)
none	No functions
resize	f.resize
move	f.move
minimize	f.minimize
maximize	f.maximize
close	f.kill

focusAutoRaise (class **FocusAutoRaise**)

When the value of this resource is True, clients are made completely unobscured when they get the keyboard input focus. If the value is False, the stacking of windows on the display is not changed when a window gets the keyboard input focus. The default value is True.

iconImage (class **IconImage**)

This resource can be used to specify an icon image for a client (e.g., Vuewm*myclock*iconImage). The resource value is a pathname for a bitmap file. The value of the (client specific) useClientIcon resource is used to determine whether or not user supplied icon images are used instead of client supplied icon images. The default value is to display a built-in window manager icon image.

iconImageBackground (class **Background**)

This resource specifies the background color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon background color (i.e., specified by Vuewm*background or Vuewm*icon*background).

iconImageBottomShadowColor (class **Foreground**)

This resource specifies the bottom shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow color (i.e., specified by Vuewm*icon*bottomShadowColor).

iconImageBottomShadowPixmap (class **BottomShadowPixmap**)

This resource specifies the bottom shadow Pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon bottom shadow Pixmap (i.e., specified by Vuewm*icon*bottomShadowPixmap).

iconImageForeground (class **Foreground**)

This resource specifies the foreground color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon foreground color (i.e., specified by Vuewm*foreground or Vuewm*icon*foreground).

iconImageTopShadowColor (class **Background**)

This resource specifies the top shadow color of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow color (i.e., specified by Vuewm*icon*topShadowColor).

iconImageTopShadowPixmap (class **TopShadowPixmap**)
This resource specifies the top shadow Pixmap of the icon image that is displayed in the image part of an icon. The default value of this resource is the icon top shadow Pixmap (i.e., specified by Vuewm*icon*topShadowPixmap).

matteBackground (class **Background**)
This resource specifies the background color of the matte, when **matteWidth** is positive. The default value of this resource is the client background color (i.e., specified by Vuewm*background or Vuewm*client*background).

matteBottomShadowColor (class **Foreground**)
This resource specifies the bottom shadow color of the matte, when **matteWidth** is positive. The default value of this resource is the client bottom shadow color (i.e., specified by Vuewm*bottomShadowColor or Vuewm*client*bottomShadowColor).

matteBottomShadowPixmap (class **BottomShadowPixmap**)
This resource specifies the bottom shadow Pixmap of the matte, when **matteWidth** is positive. The default value of this resource is the client bottom shadow Pixmap (i.e., specified by Vuewm*bottomShadowPixmap or Vuewm*client*bottomShadowPixmap).

matteForeground (class **Foreground**)
This resource specifies the foreground color of the matte, when **matteWidth** is positive. The default value of this resource is the client foreground color (i.e., specified by Vuewm*foreground or Vuewm*client*foreground).

matteTopShadowColor (class **Background**)
This resource specifies the top shadow color of the matte, when **matteWidth** is positive. The default value of this resource is the client top shadow color (i.e., specified by Vuewm*topShadowColor or Vuewm*client*topShadowColor).

matteTopShadowPixmap (class **TopShadowPixmap**)
This resource specifies the top shadow Pixmap of the matte, when **matteWidth** is positive. The default value of this resource is the client top shadow Pixmap (i.e., specified by Vuewm*topShadowPixmap or Vuewm*client*topShadowPixmap).

matteWidth (class **MatteWidth**)
This resource specifies the width of the optional matte. The default value is 0, which effectively disables the matte.

maximumClientSize (class **MaximumClientSize**)
This is a size specification that indicates the client size to be used when an application is maximized. The resource value is specified as *widthxheight*. The width and height are interpreted in the units that the client uses (e.g., for terminal emulators this is generally characters). If this resource is not specified then the maximum size from the WM_NORMAL_HINTS property is used if set. Otherwise the default value is the size where the client window with window management borders fills the screen. When the maximum client size is not determined by the maximumClientSize resource, the maximumMaximumSize resource value is used as a constraint on the maximum size.

useClientIcon (class **UseClientIcon**)
If the value given for this resource is True, then a client supplied icon image will take precedence over a user supplied icon image. The default value is False, making the user supplied icon image have higher precedence than the client supplied icon image.

windowMenu (class **WindowMenu**)
This resource indicates the name of the menu pane that is posted when the window menu is popped up (usually by pressing button 1 on the window menu button on the client window frame). Menu panes are specified in the *vuewm resource description file* file. Window menus can be customized on a client class basis by specifying resources of the form **Vuewm***_client_name_or_class_***windowMenu** (see **vuewmrc(4)**). The default value of this resource is the name of the built-in window menu specification.

FRONT PANEL RESOURCES

The front panel is an integral part of vuewm. However, from a resource perspective, it looks more like a client. The syntax for specifying front panel resources is:

Vuewm[**screen_name***]****front_panel_name*resource_id*

For example, **Vuewm*frontPanel*numberOfSwitchRows** is used to specify the number of rows of workspace buttons.

The following resources are used to configure the front panel.

Front Panel Resources

Name	Class	Value Type	Default
borderWidth	BorderWidth	pixels	0
clientTimeoutInterval	ClientTimeoutInterval	millisec	20000
controlSpacing	ControlSpacing	pixels	0
geometry	Geometry	string	dynamic
name	Name	string	varies
numberOfSwitchRows	NumberOfSwitchRows	integer	2
rowSpacing	RowSpacing	pixels	0
wsButtonBorder	WsButtonBorder	pixels	10
wsButtonShadowThickness	WsButtonShadowThickness	pixels	2
wsButtonSpacing	WsButtonSpacing	pixels	4
waitingBlinkRate	WaitingBlinkRate	millisec	500

borderWidth (class **BorderWidth**)
This resource controls the border, or margin, between the rows of controls and the surrounding front panel frame. The default value is 0.

clientTimeoutInterval (class **ClientTimeoutInterval**)
This resource specifies the time that vuewm will wait for a window to appear that is invoked by a front panel action. In many cases, there is a delay between the time a control is activated and the time a client window appears. This resource specifies the time that vuewm will wait for the window before assuming that the activation failed. The default value of this resource is 60000 ms, or 60 seconds.

Typically, the activity light on the front panel blinks and a front panel hourglass appears until either the timeout interval has expired or the client window appears on the screen.

controlSpacing (class **ControlSpacing**)
This resource specifies the spacing between controls in pixels, including the beveling. The default value is 0. (See also **shadowThickness**.)

geometry (class **Geometry**)
This resource specifies the placement of the front panel on the screen, using a simplified form of the standard window geometry. The value is determined dynamically. The default is to center the front panel at the bottom of the screen.

Of the standard window geometry string:

[=][*width*x*height*][{+-}*xoffset*{+-}*yoffset*]

--only the plus and minus forms of the *xoffset* and *yoffset* are used. A given *width* or *height* will be ignored, because **vuewm** determines the overall width and height of the front panel from the controls within it.

For example, **=-10+5** places the front panel at the upper right of the screen, indented from the left edge by 10 pixels and down from the top edge by 5 pixels.

Note that the *top row* of controls determines the overall width of the front panel--this

width will be determined by the width of controls in the top row.

If the *xoffset* is positive, then the front panel will be left-justified by the specified pixel spacing; if the *xoffset* is negative, the front panel will be right-justified; if the *xoffset* is *-0* or *+0*, the front panel will be centered horizontally.

If the *yoffset* is positive (including *+0*), the front panel will be offset from the top of the screen by the specified number of pixels; if the *yoffset* is negative (including *-0*), the front panel will be offset from the bottom of the screen.

The default geometry is *-0-0*.

name (class **Name**)
This resource sets the name of the front panel. This name is used to look up the front panel specification in the **vuewmrc** file. The default value is dynamically determined based on the screen resolution type--it may be *LargeFrontPanel* (typically, 1280 by 1024 pixels), *MediumFrontPanel* (typically, 1024 by 768 pixels), or *SmallFrontPanel*.

numberOfSwitchRows (class **NumberOfSwitchRows**)
This resource sets the number of rows in the workspace switch. The default value is 2. The **numberOfSwitchRows** front panel resource and the **workspaceList** screen resource work together to determine the button placement within the workspace switch. For example, together **Vuewm*numberOfSwitchRows: 4** and **Vuewm*workspaceList:** *<name1> <name2> <name3> <name4> <name5> <name6> <name7> <name8>* specify a workspace switch having eight workspace buttons arranged into four rows of two buttons.

The overall *size* of the workspace switch (which determines the maximum workspace button size) is set in the resource description file.

rowSpacing (class **RowSpacing**)
This resource sets the spacing in pixels between rows of controls in the front panel. The default value is 0--**vuewm** stacks the controls so that there is no spacing between the rows.

wsButtonBorder (class **WsButtonBorder**)
This resource sets the border width for the buttons in the workspace switch--the margin between the rows of buttons and the frame of the workspace switch. The default value of this resource is 10.

This resource also controls the vertical inter-button spacing of the workspace switch--the spacing between workspace buttons rows is 2 less than the **wsButtonBorder** (but always nonnegative).

wsButtonShadowThickness (class **WsButtonShadowThickness**)
This resource sets the shadow thickness (or beveling) for the buttons in the workspace switch. The default value is 2.

wsButtonSpacing (class **WsButtonSpacing**)
This resource sets the horizontal spacing between the buttons in the workspace switch. The default value is 10.

From these resource values, **vuewm** computes the sizes and layout of the buttons themselves in the workspace switch.

waitingBlinkRate (class **WaitingBlinkRate**)
This resource sets the rate of the busy light. The time set by this resource measures a half cycle of the busy light blink frequency. The light is on for this amount of time before being turned off for the same amount of time. The default value is 500 milliseconds.

FRONT PANEL ROW RESOURCES

The front panel is organized into one or more *rows* of controls. Each row can have a *name* specified in the resource description file. You can specify color sets and background patterns on a per-row basis. The syntax for specifying front panel row resources is:

Vuewm[screen_name*]**row_name*resource_id***

For example, Vuewm*Top*rowBackgroundPixmap: 75_foreground causes vuewm to create a background for the row named *Top* using a *75_foreground* pattern.

The following resources are used to configure individual rows of the front panel.

Front Panel Row Resources

Name	Class	Value Type	Default
colorSetId	ColorSetId	integer, *1* to *8*	dynamic
rowBackgroundPixmap	RowBackgroundPixmap	image_name	NULL

colorSetId (class **colorSetId**)
This resource sets the color set (with background, foreground, topshadow, and bottomshadow colors) for the specified row. The default value for this resource is determined dynamically, based on the color type of the screen.

rowBackgroundPixmap (class **RowBackgroundPixmap**)
This resource sets the background pattern to use behind the controls in a given row. The default value is NULL, or no background pixmap.

Note that a background pixmap will underlie all the controls in the specified row(s).

VUEWMDATE RESOURCES

Vuewmdate is a psuedo-client that shows you the current date, like a desk-top calendar. It is built into vuewm. The format of the date display can include multiple lines and multiple fonts. The date format is configurable through the use of **strftime (3)**.

The syntax for specifying resources for vuewmdate is:

Vuewm*vuewmdate*resource_id

For example, **Vuewm*vuewmdate*fontList2** is used to specify a fontList for the first line of the vuewmdate display.

The following resources are used to configure vuewmdate.

Vuewmdate Resources

Name	Class	Value Type	Default
fontList	FontList	string	"fixed"
fontList1	FontList	string	fontList
fontList2	FontList	string	fontList
fontList3	FontList	string	fontList
fontList4	FontList	string	fontList
fontList5	FontList	string	fontList
format	Format	string	"%b%n%.1%n%a"

fontList (class **FontList**)
This resource describes the font to use for the text in the date display. The default value is "fixed".

fontList1 (class **FontList**)
This resource describes the font to use in the first line of the date display. The default value of this resource is the value of vuewmdate's fontList resource.

fontList2 (class **FontList**)
This resource describes the font to use in the second line of the date display. The default value of this resource is the value of vuewmdate's fontList resource.

fontList3 (class **FontList**)
This resource describes the font to use in the third line of the date display. The default value of this resource is the value of vuewmdate's fontList resource.

fontList4 (class **FontList**)
This resource describes the font to use in the fourth line of the date display. The default value of this resource is the value of vuewmdate's fontList resource.

fontList5 (class **FontList**)
This resource describes the font to use in the fifth line of the date display. The default value of this resource is the value of vuewmdate's fontList resource.

format (class **Format**)
This resource provides the format string for the vuewmdate date. The format string is passed to **strftime(3)** for interpretation. The character string returned from that function is parsed for new lines and displayed appropriately in the vuewmdate area. The default value for this resource is "%b%n%.1d%n%a".

WORKSPACE RESOURCES

The only workspace resource is the **title** resource. The title of a workspace appears in the workspace button in the workspace switch. It is a user-visible name for the workspace whereas the workspace name given in the **workspaceList** resource is used for fetching resources.

The syntax for specifying the workspace title resource is:

Vuewm[**screen_name***]****workspace_name****title**

For example, **Vuewm*0*A*title** is used to set the title of workspace **A** on screen **0**. Note that *workspace_name* must appear elsewhere in the **workspaceList** resource.

The following table describes the workspace title resource.

Workspace Title Resource

Name	Class	Value Type	Default
title	Title	XmString	varies*

*The default title is the workspace name.

title (class **Title**)
This resource specifies the user-visible title of a workspace. The default value is the workspace name (as given in the **workspaceList** resource).

BACKDROP RESOURCES

The backdrop resources are the only resource that can be specified on a per-workspace basis. These resource set the backdrop bitmap image and colors to be used for that workspace.

The syntax for specifying backdrop resources is:

Vuewm[**screen_name***][****workspace_name***]*backdrop****resource_id*

For example, **Vuewm*0*Main*backdrop*image** is used to set the backdrop bitmap image for the **Main** workspace on screen **0**.

The following resources are used to configure vuewmdate.

Backdrop Resources

Name	Class	Value Type	Default
colorSetId	ColorSetId	integer	NULL
image	Image	string	"none"
imageBackground	ImageBackground	pixel	varies*
imageForeground	ImageForeground	pixel	varies*

*The default is chosen based on the display type.

colorSetId (class **ColorSetId**)
This resource specifies a color set to use for the backdrop. This is normally an integer from 1 to 8. The default value of this resource is display dependent.

image (class **Image**)
This resource specifies a bitmap image to use as the backdrop. If "none" is specified, then no backdrop image will be used. The root window will show through. A bitmap file can be specified, provided it is preceded by a '@' character. If the first four letters in the base name of the bitmap file are "deep", then vuewm will create a 3D shaft with the bitmap image at the "bottom". Image files are fetched using the directory search path of **XmGetPixmap()**.

The default value for this resource is none (no image).

imageBackground (class **ImageBackground**)
This resource specifies a color to use as the background color of the backdrop. If not specified, this color will default to a pixel value that can be dynamically customized with the color customizer, providing you are running in the HP VUE. If this resource is specified, a color cell will be allocated outside of any dynamically customizable set. The default value varies based on the screen type.

imageForeground (class **ImageForeground**)
This resource specifies a color to use as the foreground color of the backdrop. If not specified, this color will default to a pixel value that can be dynamically customized with the color customizer, providing you are running in the HP VUE. If this resource is specified, a color cell will be allocated outside of any dynamically customizable set. The default value varies based on the screen type.

DIALOG RESOURCES

The titles of the Workspace Rename and Workspace Presence dialog are settable via resources.

The syntax for setting the Workspace Rename title is:

Vuewm*_screen_name_*workspaceController*title

Similarly, the syntax for setting the Workspace Presence title is:

Vuewm*_screen_name_*workspacePresence*title

The title subresource is described below:

Workspace Dialog Resources

Name	Class	Value Type	Default
title	Title	XmString	varies*

*The default name of the dialog: "Workspace Rename" or "Workspace Presence".

title (class **Title**)
This resource specifies the title of a workspace dialog. The default value varies based on the dialog.

RESOURCE DESCRIPTION FILE

The *vuewm resource description file* is a supplementary resource file that contains resource descriptions that are referred to by entries in the defaults files (.Xdefaults, app-defaults/Vuewm). It contains descriptions of resources that are to be used by vuewm, and that cannot be easily encoded in the defaults files.

The following types of resources can be described in the *vuewm resource description file*:

Buttons	Window manager functions can be bound (associated) with button events.
Front Panel	The vuewm front panel can be configured for a custom content or layout.
Keys	Window manager functions can be bound (associated) with key press events.
Menus	Menu panes can be used for the window menu and other menus posted with key bindings and button bindings.

Use **vuewm** resources (such as **vuewm*windowMenu** and **vuewm*frontPanelName**) described on the preceding pages to select which corresponding description to use from the resource description file.

ENVIRONMENT

Vuewm uses the environment variable **$HOME** specifying the user's home directory.

Vuewm uses the environment variable **$MWMSHELL** (first) or **$SHELL** (second) for specifying the user's shell in carrying out f.exec functions. (If neither variable is set, **vuewm** uses **/bin/sh**.)

Vuewm uses the environment variable **$LANG** to specify the location of the resource description file and its localized message file.

FILES

$HOME/.vue/$LANG/vuewmrc
$HOME/.vue/vuewmrc
/usr/lib/X11/vue/Vuewm/$LANG/sys.vuewmrc
/usr/lib/X11/vue/Vuewm/sys.vuewmrc
/usr/lib/X11/bitmaps/Vuewm/*
/usr/lib/nls/$LANG/vuewm.cat
$HOME/.Xdefaults
/usr/lib/X11/app-defaults/Vuewm

ORIGIN

Hewlett-Packard Company ITO.

SEE ALSO

vuesession(1X), **vuestyle(1X)**, **X(1)**, **VendorShell(3X)**, **strftime(3)**, **XmInstallImage(3X)**, **XmGetPixmap(3X)**, and **vuewmrc(4X)**.

NAME

vuewmrc - The HP VUE Workspace Manager Resource Description File.

DESCRIPTION

The *vuewm resource description file* is a supplementary resource file that controls much of the operation of the HP Visual User Environment Window Manager (*vuewm*). It contains descriptions of resources that are to be used by vuewm, and that cannot be easily written using standard X Window System, Version 11 resource syntax:

Vuewm* *resource_id*

The resource description file contains entries that are cross-referenced by entries in the resource defaults files (for example, **/usr/lib/X11/app-defaults/Vuewm**). For example, the resource description file enables you to specify different types of window menus; on the other hand, the default resource file enables you to specify *which* of these window menus the manager should use for a particular window.

The specifications of the resource description file supported by the HP VUE workspace manager are a strict superset of the specifications supported by the HP/OSF Motif Window Manager (**mwm 1.0**). In other words, the **/usr/lib/X11/system.mwmrc** or **$HOME/.mwmrc** file that you've used for **mwm** is immediately usable by **vuewm**: Simply open the **vuewm** resource description file with a text editor (make a copy first!), then cut and paste specifications from your current **.mwmrc** file into it. When you restart the workspace manager, your menu, key, and button definitions will take immediate effect.

Location

The workspace manager searches for one of the following *resource description files*, where $LANG is the value of the language environment on a per-user basis:

$HOME/.vue/$LANG/vuewmrc

$HOME/.vue/vuewmrc

/usr/lib/X11/vue/Vuewm/$LANG/sys.vuewmrc

/usr/lib/X11/vue/Vuewm/sys.vuewmrc

The first file found is the first used.

If none is found, then a set of *built-in* specifications is used. For example, the front panel, if used, will consist solely of the workspace switch (a set of buttons for changing workspaces).

A particular *resource description file* can be selected using the **configFile** resource.

The following shows how a different resource description file can be specified from the command line:

/usr/bin/X11/vuewm -xrm "Vuewm*configFile: myvuewmrc"

Resource Types

The following types of resources can be described in the *vuewm resource description file*:

Buttons	Workspace manager functions can be bound (associated) with button events.
Keys	Workspace manager functions can be bound (associated) with key press events.
Menus	Menu panes can be used for the window menu and other menus posted with key bindings and button bindings.
Front Panel	The vuewm front panel can be configured for a custom content or layout.

VUEWM RESOURCE DESCRIPTION FILE SYNTAX

The *vuewm resource description file* is a standard text file that contains items of information separated by blanks, tabs, and new lines characters. Blank lines are ignored. Items or characters can be quoted to avoid special interpretation (e.g., the comment character can be quoted to prevent it from being interpreted as the comment character). A quoted item can be contained in

double quotes (").

Single characters can be quoted by preceding them by the back-slash character (\), except for workspace names, which may contain no back-slash characters.

All text from an unquoted # to the end of the line is regarded as a comment and is not interpreted as part of a resource description. If ! is the first character in a line, the line is regarded as a comment.

Workspace Manager Functions

Workspace manager functions can be accessed with button and key bindings, and with workspace manager menus. Functions are indicated as part of the specifications for button and key binding sets, and menu panes. The function specification has the following syntax:

function = *function_name* [*function_args*]
function_name = *workspace manager function*
function_args = {*quoted_item* | *unquoted_item*}

The following functions are supported. If a function is specified that isn't one of the supported functions then it is interpreted by vuewm as *f.nop*.

f.action This function causes the specified *action* to be invoked by means of the message server.

f.beep This function causes a beep.

f.circle_down [icon | window]
This function causes the window or icon that is on the top of the window stack to be put on the bottom of the window stack (so that it is no longer obscuring any other window or icon). This function affects only those windows and icons that are obscuring other windows and icons, or that are obscured by other windows and icons. Secondary windows (i.e. transient windows) are restacked with their associated primary window. Secondary windows always stay on top of the associated primary window and there can be no other primary windows between the secondary windows and their primary window. If an **icon** function argument is specified, then the function applies only to icons. If a **window** function argument is specified then the function applies only to windows.

f.circle_up [icon | window]
This function raises the window or icon on the bottom of the window stack (so that it is not obscured by any other windows). This function affects only those windows and icons that are obscuring other windows and icons, or that are obscured by other windows and icons. Secondary windows (i.e. transient windows) are restacked with their associated primary window. If an *icon* function argument is specified then the function applies only to icons. If an *window* function argument is specified then the function applies only to windows.

f.exec or **!**
This function causes *command* to be executed (using the value of the *$SHELL* environment variable if it is set, otherwise */bin/sh*). The ! notation can be used in place of the **f.exec** function name.

f.focus_color
This function sets the colormap focus to a client window. If this function is done in a root context, then the default colormap (setup by the *X Window System* for the screen where vuewm is running) is installed and there is no specific client window colormap focus. This function is treated as *f.nop* if *colormapFocusPolicy* is not explicit.

f.focus_key
This function sets the keyboard input focus to a client window or icon. This function is treated as *f.nop* if *keyboardFocusPolicy* is not explicit or the function is executed in a root context.

f.goto_workspace
This function causes the workspace manager to switch to the specified *workspace*. If no workspace exists by the specified name, then no action occurs.

f.kill If the WM_DELETE_WINDOW protocol is set up, the client is sent a client message event indicating that the client window should be deleted. If the WM_SAVE_YOURSELF protocol is set up and the WM_DELETE_WINDOW protocol is not set up, the client is sent a client message event indicating that the client needs to prepare to be terminated. If the client does not have the WM_DELETE_WINDOW or WM_SAVE_YOURSELF protocol set up, this function causes a client's X connection to be terminated (usually resulting in termination of the client). Refer to the description of the *quitTimeout* resource and the *WM_PROTOCOLS* property.

f.lower [*-client*]
This function lowers a client window to the bottom of the window stack (where it obscures no other window). Secondary windows (i.e. transient windows) are restacked with their associated primary window. The *client* argument indicates the name or class of a client to lower. If the *client* argument is not specified then the context that the function was invoked in indicates the window or icon to lower.

f.maximize
This function causes a client window to be displayed with its maximum size.

f.menu This function associates a cascading (pull-right) menu with a menu pane entry or a menu with a button or key binding. The *menu_name* function argument identifies the menu to be used.

f.minimize
This function causes a client window to be minimized (iconified). When a window is minimized with no icon box in use, and if the **lowerOnIconify** resource has the value True (the default), the icon is placed on the bottom of the window stack (such that it obscures no other window). If an icon box is used, then the client's icon changes to its iconified form inside the icon box. Secondary windows (i.e. transient windows) are minimized with their associated primary window. There is only one icon for a primary window and all its secondary windows.

f.move This function allows a client window or icon to be moved interactively.

f.next_cmap
This function installs the next colormap in the list of colormaps for the window with the colormap focus.

f.next_key [icon | window | transient]
This function sets the keyboard input focus to the next window/icon in the set of windows/icons managed by the workspace manager (the ordering of this set is based on the stacking of windows on the screen). This function is treated as *f.nop* if *keyboardFocusPolicy* is not explicit. The keyboard input focus is only moved to windows that do not have an associated secondary window that is application modal. If the **transient** argument is specified, then transient (secondary) windows are traversed (otherwise, if only **window** is specified, traversal is done only to the last focused window in a transient group). If an **icon** function argument is specified, then the function applies only to icons. If a **window** function argument is specified, then the function applies only to windows.

f.next_workspace
This function causes the workspace manager to switch to the next workspace. If you're viewing the last workspace, the switch occurs to the first workspace.

f.nop This function does nothing.

f.normalize
This function causes a client window to be displayed with its normal size. Secondary windows (i.e. transient windows) are placed in their normal state along with

their associated primary window.

f.normalize_and_raise
This function causes a client window to be displayed with its normal size and raised to the top of the window stack. Secondary windows (i.e. transient windows) are placed in their normal state along with their associated primary window.

f.occupy_all
This function causes the associated window to be placed in all workspaces.

f.pack_icons
This function is used to relayout icons (based on the layout policy being used) on the root window or in the icon box. In general this causes icons to be "packed" into the icon grid.

f.pass_keys
This function is used to enable/disable (toggle) processing of key bindings for workspace manager functions. When it disables key binding processing all keys are passed on to the window with the keyboard input focus and no workspace manager functions are invoked. If the *f.pass_keys* function is invoked with a key binding to disable key binding processing the same key binding can be used to enable key binding processing.

f.post_wmenu
This function is used to post the window menu. If a key is used to post the window menu and a window menu button is present, the window menu is automatically placed with its top-left corner at the bottom-left corner of the window menu button for the client window. If no window menu button is present, the window menu is placed at the top-left corner of the client window.

f.prev_cmap
This function installs the previous colormap in the list of colormaps for the window with the colormap focus.

f.prev_key [icon | window | transient]
This function sets the keyboard input focus to the previous window/icon in the set of windows/icons managed by the workspace manager (the ordering of this set is based on the stacking of windows on the screen). This function is treated as *f.nop* if *keyboardFocusPolicy* is not explicit. The keyboard input focus is only moved to windows that do not have an associated secondary window that is application modal. If the *transient* argument is specified, then transient (secondary) windows are traversed (otherwise, if only *window* is specified, traversal is done only to the last focused window in a transient group). If an *icon* function argument is specified then the function applies only to icons. If an *window* function argument is specified then the function applies only to windows.

f.prev_workspace
This function causes the workspace manager to switch to the previous workspace. If you're viewing the first workspace, the switch occurs to the last workspace.

f.quit_vuewm
This function terminates vuewm (but NOT the X window system).

f.raise [*-client*]
This function raises a client window to the top of the window stack (where it is obscured by no other window). Secondary windows (i.e. transient windows) are restacked with their associated primary window. The *client* argument indicates the name or class of a client to raise. If the *client* argument is not specified then the context that the function was invoked in indicates the window or icon to raise.

f.raise_lower
This function raises a client window to the top of the window stack if it is partially obscured by another window, otherwise it lowers the window to the bottom of the

window stack. Secondary windows (i.e. transient windows) are restacked with their associated primary window.

f.refresh
This function causes all windows to be redrawn.

f.refresh_win
This function causes a client window to be redrawn.

f.remove
This function causes a client window to be removed from the current workspace. If the client window exists only in this workspace, no action will occur.

f.resize This function allows a client window to be interactively resized.

f.restart
This function causes vuewm to be restarted (effectively terminated and re-executed).

f.send_msg *message_number*
This function sends a client message of the type _MOTIF_WM_MESSAGES with the *message_type* indicated by the *message_number* function argument. The client message will only be sent if *message_number* is included in the client's _MOTIF_WM_MESSAGES property. A menu item label is grayed out if the menu item is used to do *f.send_msg* of a message that is not included in the client's _MOTIF_WM_MESSAGES property.

f.separator
This function causes a menu separator to be put in the menu pane at the specified location (the label is ignored).

f.set_behavior
This function causes the workspace manager to restart with the default behavior (if a custom behavior is configured) or a custom behavior (if a default behavior is configured).

f.title This function inserts a title in the menu pane at the specified location.

f.toggle_frontpanel
This function causes the front panel component of the workspace manager to be minimized or, alternately, normalized.

f.version
This function causes the workspace manager to display its release version in a dialog box.

f.workspace_presence
This function posts the workspace presence dialog box. This allows you to view and set the workspace in which a particular window resides. The root context is disallowed for this function.

f.workspace_rename
This function posts the workspace rename dialog box. This allows you to rename the current workspace.

Each function may be constrained as to which resource types can specify the function (e.g., menu pane) and also what context the function can be used in (e.g., the function is done to the selected client window). Function contexts are

root	No client window or icon has been selected as an object for the function.
window	A client window has been selected as an object for the function. This includes the window's title bar and frame. Some functions are applied only when the window is in its normalized state (e.g., f.maximize) or its maximized state (e.g., f.normalize).
icon	An icon has been selected as an object for the function.

If a function is specified in a type of resource where it is not supported or is invoked in a context that does not apply then the function is treated as *f.nop*. The following table indicates the resource types and function contexts in which workspace manager functions apply.

Function	Contexts	Resources
f.action	root,icon,window	button,key,menu
f.beep	root,icon,window	button,key,menu
f.circle_down	root,icon,window	button,key,menu
f.circle_up	root,icon,window	button,key,menu
f.exec	root,icon,window	button,key,menu
f.focus_color	root,icon,window	button,key,menu
f.focus_key	root,icon,window	button,key,menu
f.goto_workspace	root,icon,window	button,key,menu
f.kill	icon,window	button,key,menu
f.lower	root,icon,window	button,key,menu
f.maximize	icon,window(normal)	button,key,menu
f.menu	root,icon,window	button,key,menu
f.minimize	window	button,key,menu
f.move	icon,window	button,key,menu
f.next_cmap	root,icon,window	button,key,menu
f.next_key	root,icon,window	button,key,menu
f.next_workspace	root,icon,window	button,key,menu
f.nop	root,icon,window	button,key,menu
f.normalize	icon,window(maximized)	button,key,menu
f.normalize_and_raise	icon,window	button,key,menu
f.occupy_all	icon, window	button,key,menu
f.pack_icons	root,icon,window	button,key,menu
f.pass_keys	root,icon,window	button,key,menu
f.post_wmenu	root,icon,window	button,key
f.prev_cmap	root,icon,window	button,key,menu
f.prev_key	root,icon,window	button,key,menu
f.prev_workspace	root,icon,window	button,key,menu
f.quit_vuewm	root	button,key,menu
f.raise	root,icon,window	button,key,menu
f.raise_lower	icon,window	button,key,menu
f.refresh	root,icon,window	button,key,menu
f.refresh_win	window	button,key,menu
f.remove	root,icon,window	button,key,menu
f.resize	window	button,key,menu
f.restart	root	button,key,menu
f.send_msg	icon,window	button,key,menu
f.separator	root,icon,window	menu
f.set_behavior	root,icon,window	button,key,menu
f.title	root,icon,window	menu
f.toggle_frontpanel	root,icon,window	button,key,menu
f.version	root,icon,window	button,key,menu
f.workspace_presence	window	button,key,menu
f.workspace_rename	root,icon,window	button,key,menu

WORKSPACE MANAGER EVENT SPECIFICATION

Events are indicated as part of the specifications for button and key binding sets, and menu panes.

Button events have the following syntax:

button = [*modifier_list*]<*button_event_name*>

modifier_list = *modifier_name* {*modifier_name*}

All modifiers specified are interpreted as being exclusive (this means that only the specified modifiers can be present when the button event occurs). The following table indicates the values that can be used for *modifier_name*. The [Alt] key is frequently labeled [Extend] or [Meta]. Alt and Meta can be used interchangeably in event specification.

Modifier	Description
Ctrl	Control Key
Shift	Shift Key
Alt	Alt/Meta Key
Meta	Meta/Alt Key
Lock	Lock Key
Mod1	Modifier1
Mod2	Modifier2
Mod3	Modifier3
Mod4	Modifier4
Mod5	Modifier5

The following table indicates the values that can be used for *button_event_name*.

Button	Description
Btn1Down	Button 1 Press
Btn1Up	Button 1 Release
Btn1Click	Button 1 Press and Release
Btn1Click2	Button 1 Double Click
Btn2Down	Button 2 Press
Btn2Up	Button 2 Release
Btn2Click	Button 2 Press and Release
Btn2Click2	Button 2 Double Click
Btn3Down	Button 3 Press
Btn3Up	Button 3 Release
Btn3Click	Button 3 Press and Release
Btn3Click2	Button 3 Double Click
Btn4Down	Button 4 Press
Btn4Up	Button 4 Release
Btn4Click	Button 4 Press and Release
Btn4Click2	Button 4 Double Click
Btn5Down	Button 5 Press
Btn5Up	Button 5 Release
Btn5Click	Button 5 Press and Release
Btn5Click2	Button 5 Double Click

Key events that are used by the workspace manager for menu mnemonics and for binding to workspace manager functions are single key presses; key releases are ignored. Key events have the following syntax:

key = [*modifier_list*]**<Key>***key_name*
modifier_list = *modifier_name* {*modifier_name*}

All modifiers specified are interpreted as being exclusive (this means that only the specified modifiers can be present when the key event occurs). Modifiers for keys are the same as those that apply to buttons. The *key_name* is an X11 keysym name. Keysym names can be found in the keysymdef.h file (remove the *XK_* prefix).

BUTTON BINDINGS

The **buttonBindings** resource value is the name of a set of button bindings that are used to configure workspace manager behavior. A workspace manager function can be done when a button press occurs with the pointer over a framed client window, an icon or the root window. The

context for indicating where the button press applies is also the context for invoking the workspace manager function when the button press is done (significant for functions that are context sensitive).

The button binding syntax is

Buttons *bindings_set_name*
{
  button context function
  button context function
  .
  .
  button context function
}

The syntax for the *context* specification is

context = *object*[|*context*]
object = **root | icon | window | title | frame | border | app**

The context specification indicates where the pointer must be for the button binding to be effective. For example, a context of **window** indicates that the pointer must be over a client window or window management frame for the button binding to be effective. The **frame** context is for the window management frame around a client window (including the border and titlebar), the **border** context is for the border part of the window management frame (not including the titlebar), the **title** context is for the title area of the window management frame, and the **app** context is for the application window (not including the window management frame).

If an *f.nop* function is specified for a button binding, the button binding will not be done.

KEY BINDINGS

The **keyBindings** resource value is the name of a set of key bindings that are used to configure workspace manager behavior. A window manager function can be done when a particular key is pressed. The context in which the key binding applies is indicated in the key binding specification. The valid contexts are the same as those that apply to button bindings.

The key binding syntax is

Keys *bindings_set_name*
{
  key context function
  key context function
  .
  .
  key context function
}

If an *f.nop* function is specified for a key binding, the key binding will not be done. If an *f.post_wmenu* or *f.menu* function is bound to a key, vuewm will automatically use the same key for removing the menu from the screen after it has been popped up.

The *context* specification syntax is the same as for button bindings. For key bindings, the **frame**, **title**, **border**, and **app** contexts are equivalent to the **window** context. The context for a key event is the window or icon that has the keyboard input focus (**root** if no window or icon has the keyboard input focus).

MENU PANES

Menus can be popped up using the *f.post_wmenu* and *f.menu* workspace manager functions. The context for workspace manager functions that are done from a menu is *root*, *icon* or *window*

depending on how the menu was popped up. In the case of the *window* menu or menus popped up with a key binding, the location of the keyboard input focus indicates the context. For menus popped up using a button binding, the context of the button binding is the context of the menu.

The menu pane specification syntax is

```
Menu menu_name
{
  label [mnemonic] [accelerator] function
  label [mnemonic] [accelerator] function
        .
        .
  label [mnemonic] [accelerator] function
}
```

Each line in the *Menu* specification identifies the label for a menu item and the function to be done if the menu item is selected. Optionally a menu button mnemonic and a menu button keyboard accelerator may be specified. Mnemonics are functional only when the menu is posted and keyboard traversal applies.

The *label* may be a string or a bitmap file. The label specification has the following syntax:

```
label =       text | bitmap_file
bitmap_file = @file_name
text =        quoted_item | unquoted_item
```

The string encoding for labels must be compatible with the menu font that is used. Labels are greyed out for menu items that do the *f.nop* function or an invalid function or a function that doesn't apply in the current context.

A *mnemonic* specification has the following syntax

```
mnemonic =  _character
```

The first matching *character* in the label is underlined. If there is no matching *character* in the label, no mnemonic is registered with the workspace manager for that label. Although the *character* must exactly match a character in the label, the mnemonic will not execute if any modifier (such as Shift) is pressed with the character key.

The *accelerator* specification is a key event specification with the same syntax as is used for key bindings to workspace manager functions.

FRONT PANEL CONTENT AND LAYOUT

The *resource description file* enables you to specify the content and layout of the front panel, including the number of controls, their size and arrangment within the front panel, and their operation.

The layout is arranged horizontally, in terms of *rows* of controls.

The front panel specification syntax is demonstrated in the **sys.vuewmrc** system file included in the **/usr/lib/X11/vue/Vuewm** directory.

ERRORS

Errors that occur as a result of parsing or trying to use the resource description file will be reported to a log file under your home directory: **$HOME/.vue/errorlog**. Be sure to check this text file if the appearance or behavior of **vuewm** is not what you expect.

FILES

$HOME/.vue/$LANG/vuewmrc

$HOME/.vue/vuewmrc
/usr/lib/X11/vue/Vuewm/$LANG/sys.vuewmrc
/usr/lib/X11/vue/Vuewm/sys.vuewmrc
$HOME/.vue/errorlog

COPYRIGHT

AUTHOR

The OSF/HP Motif Window Manager and the Hewlett-Packard Visual User Environment Window Manager were developed by HP.

ORIGIN

Hewlett-Packard Company ITO

SEE ALSO

vuewm(1X) and X(1).

NAME

xload - load average display for X

SYNOPSIS

xload [*-toolkitoption ...*] [-scale *integer*] [-update *seconds*] [-hl *color*] [-highlight *color*] [-jumpscroll *pixels*] [-label *string*] [-nolabel]

DESCRIPTION

The *xload* program displays a periodically updating histogram of the system load average.

OPTIONS

Xload accepts all of the standard X Toolkit command line options (see *X(1)*). The order of the options in unimportant. *Xload* also accepts the following additional options:

-hl *color* or **-highlight** *color*
This option specifies the color of the scale lines.

-jumpscroll *number of pixels*
The number of pixels to shift the graph to the left when the graph reaches the right edge of the window. The default value is 1/2 the width of the current window. Smooth scrolling can be achieved by setting it to 1.

-label *string*
The string to put into the label above the load average.

-nolabel If this command line option is specified then no label will be displayed above the load graph.

-scale *integer*
This option specifies the minimum number of tick marks in the histogram, where one division represents one load average point. If the load goes above this number, *xload* will create more divisions, but it will never use fewer than this number. The default is 1.

-update *seconds*
This option specifies the frequency in seconds at which *xload* updates its display. The minimum amount of time allowed between updates is 1 second (the default is 5 seconds).

RESOURCES

In addition to the resources available to each of the widgets used by *xload* there is one resource defined by the application itself.

showLabel (class **Boolean**)
If False then no label will be displayed.

WIDGETS

In order to specify resources, it is useful to know the hierarchy of the widgets which compose *xload*. In the notation below, indentation indicates hierarchical structure. The widget class name is given first, followed by the widget instance name.

```
XLoad xload
        Paned paned
                Label label
                StripChart load
```

ENVIRONMENT

DISPLAY
The default host and display number.

XENVIRONMENT
The name of a resource file that overrides the global resources stored in the RESOURCE_MANAGER property.

FILES

/usr/lib/X11/app-defaults/XLoad - specifies required resources

SEE ALSO

X(1), xrdb(1), mem(4), Athena StripChart Widget.

BUGS

This program requires the ability to open and read the special system file */dev/kmem*. Sites that do not allow general access to this file should make *xload* belong to the same group as */dev/kmem* and turn on the *set group id* permission flag.

Reading /dev/kmem is inherently non-portable. Therefore, the routine used to read it (get_load.c) must be ported to each new operating system.

COPYRIGHT

AUTHORS

K. Shane Hartman (MIT-LCS) and Stuart A. Malone (MIT-LCS);
with features added by Jim Gettys (MIT-Athena), Bob Scheifler (MIT-LCS), Tony Della Fera (MIT-Athena), and Chris Peterson (MIT-LCS).

NAME

xterm - terminal emulator for X

SYNOPSIS

xterm [-toolkitoption] [-option]

DESCRIPTION

The *xterm* program is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for programs that can't use the window system directly. If the underlying operating system supports terminal resizing capabilities (for example, the SIGWINCH signal in systems derived from 4.3bsd), *xterm* will use the facilities to notify programs running in the window whenever it is resized.

The VT102 and Tektronix 4014 terminals each have their own window so that you can edit text in one and look at graphics in the other at the same time. To maintain the correct aspect ratio (height/width), Tektronix graphics will be restricted to the largest box with a 4014's aspect ratio that will fit in the window. This box is located in the upper left area of the window.

Although both windows may be displayed at the same time, one of them is considered the "active" window for receiving keyboard input and terminal output. This is the window that contains the text cursor and whose border highlights whenever the pointer is in either window. The active window can be chosen through escape sequences, the "Modes" menu in the VT102 window, and the "Tektronix" menu in the 4014 window.

OPTIONS

The *xterm* terminal emulator accepts all of the standard X Toolkit command line options as well as the following (if the option begins with a '+' instead of a '-', the option is restored to its default value):

-help This causes *xterm* to print out a verbose message describing its options.

-132 Normally, the VT102 DECCOLM escape sequence that switches between 80 and 132 column mode is ignored. This option causes the DECCOLM escape sequence to be recognized, and the *xterm* window will resize appropriately.

-ah This option indicates that *xterm* should always highlight the text cursor and borders. By default, *xterm* will display a hollow text cursor whenever the focus is lost or the pointer leaves the window.

+ah This option indicates that *xterm* should do text cursor highlighting.

-b *number*
This option specifies the size of the inner border (the distance between the outer edge of the characters and the window border) in pixels. The default is 2.

-cc *characterclassrange:value*[,...]
This sets classes indicated by the given ranges for using in selecting by words. See the section specifying character classes.

-cn This option indicates that newlines should not be cut in line-mode selections.

+cn This option indicates that newlines should be cut in line-mode selections.

-cr *color* This option specifies the color to use for text cursor. The default is to use the same foreground color that is used for text.

-cu This option indicates that *xterm* should work around a bug in the *curses*(3x) cursor motion package that causes the *more*(1) program to display lines that are exactly the width of the window and are followed by a line beginning with a tab to be displayed incorrectly (the leading tabs are not displayed).

+cu This option indicates that that *xterm* should not work around the *curses*(3x) bug mentioned above.

-e *program* [*arguments* ...]
This option specifies the program (and its command line arguments) to be run in the *xterm* window. It also sets the window title and icon name to be the basename of the

program being executed if neither *-T* nor *-n* are given on the command line. **This must be the last option on the command line.**

-fb *font* This option specifies a font to be used when displaying bold text. This font must be the same height and width as the normal font. If only one of the normal or bold fonts is specified, it will be used as the normal font and the bold font will be produced by overstriking this font. The default is to do overstriking of the normal font.

-j This option indicates that *xterm* should do jump scrolling. Normally, text is scrolled one line at a time; this option allows *xterm* to move multiple lines at a time so that it doesn't fall as far behind. Its use is strongly recommended since it make *xterm* much faster when scanning through large amounts of text. The VT100 escape sequences for enabling and disabling smooth scroll as well as the "Modes" menu can be used to turn this feature on or off.

+j This option indicates that *xterm* should not do jump scrolling.

-l This option indicates that *xterm* should send all terminal output to a log file as well as to the screen. This option can be enabled or disabled using the "xterm X11" menu.

+l This option indicates that *xterm* should not do logging.

-lf *filename*
This option specifies the name of the file to which the output log described above is written. If *file* begins with a pipe symbol (|), the rest of the string is assumed to be a command to be used as the endpoint of a pipe. The default filename is "**XtermLog.***XXXXX*" (where *XXXXX* is the process id of *xterm*) and is created in the directory from which *xterm* was started (or the user's home directory in the case of a login window).

-ls This option indicates that the shell that is started in the *xterm* window be a login shell (i.e. the first character of argv[0] will be a dash, indicating to the shell that it should read the user's .login or .profile).

+ls This option indicates that the shell that is started should not be a login shell (i.e. it will be a normal "subshell").

-mb This option indicates that *xterm* should ring a margin bell when the user types near the right end of a line. This option can be turned on and off from the "Modes" menu.

+mb This option indicates that margin bell should not be rung.

-mc milliseconds
This option specifies the maximum time between multi-click selections.

-ms *color*
This option specifies the color to be used for the pointer cursor. The default is to use the foreground color.

-nb *number*
This option specifies the number of characters from the right end of a line at which the margin bell, if enabled, will ring. The default is 10.

-rw This option indicates that reverse-wraparound should be allowed. This allows the cursor to back up from the leftmost column of one line to the rightmost column of the previous line. This is very useful for editing long shell command lines and is encouraged. This option can be turned on and off from the "Modes" menu.

+rw This option indicates that reverse-wraparound should not be allowed.

-aw This option indicates that auto-wraparound should be allowed. This allows the cursor to automatically wrap to the beginning of the next line when when it is at the rightmost position of a line and text is output.

+aw This option indicates that auto-wraparound should not be allowed.

-s This option indicates that *xterm* may scroll asynchronously, meaning that the screen does not have to be kept completely up to date while scrolling. This allows *xterm* to run faster when network latencies are very high and is typically useful when running across a very large internet or many gateways.

+s This option indicates that *xterm* should scroll synchronously.

-sb This option indicates that some number of lines that are scrolled off the top of the window should be saved and that a scrollbar should be displayed so that those lines can be viewed. This option may be turned on and off from the "Modes" menu.

+sb This option indicates that a scrollbar should not be displayed.

-sf This option indicates that Sun Function Key escape codes should be generated for function keys.

+sf This option indicates that the standard escape codes should be generated for function keys.

-si This option indicates that output to a window should not automatically reposition the screen to the bottom of the scrolling region. This option can be turned on and off from the "Modes" menu.

+si This option indicates that output to a window should cause it to scroll to the bottom.

-sk This option indicates that pressing a key while using the scrollbar to review previous lines of text should cause the window to be repositioned automatically in the normal position at the bottom of the scroll region.

+sk This option indicates that pressing a key while using the scrollbar should not cause the window to be repositioned.

-sl *number*
This option specifies the number of lines to save that have been scrolled off the top of the screen. The default is 64.

-t This option indicates that *xterm* should start in Tektronix mode, rather than in VT102 mode. Switching between the two windows is done using the "Modes" menus.

+t This option indicates that *xterm* should start in VT102 mode.

-tm *string*
This option specifies a series of terminal setting keywords followed by the characters that should be bound to those functions, similar to the *stty* program. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate delete.

-tn *name*
This option specifies the name of the terminal type to be set in the TERM environment variable. This terminal type must exist in the *termcap(5)* database and should have *li#* and *co#* entries.

-ut This option indicates that *xterm* shouldn't write a record into the the system log file */etc/utmp*.

+ut This option indicates that *xterm* should write a record into the system log file */etc/utmp*.

-vb This option indicates that a visual bell is preferred over an audible one. Instead of ringing the terminal bell whenever a Control-G is received, the window will be flashed.

+vb This option indicates that a visual bell should not be used.

-wf This option indicates that *xterm* should wait for the window to be mapped the first time before starting the subprocess so that the initial terminal size settings and environment variables are correct. It is the application's responsibility to catch subsequent terminal size changes.

+wf This option indicates that *xterm* show not wait before starting the subprocess.

-C This option indicates that this window should receive console output. This is not supported on all systems.

-S*ccn* This option specifies the last two letters of the name of a pseudoterminal to use in slave mode, plus the number of the inherited file descriptor. The option is parsed "%c%c%d".

This allows *xterm* to be used as an input and output channel for an existing program and is sometimes used in specialized applications.

The following command line arguments are provided for compatibility with older versions. They may not be supported in the next release as the X Toolkit provides standard options that accomplish the same task.

%geom This option specifies the preferred size and position of the Tektronix window. It is shorthand for specifying the "**tekGeometry*" resource.

#geom This option specifies the preferred position of the icon window. It is shorthand for specifying the "**iconGeometry*" resource.

-T *string* This option specifies the title for *xterm*'s windows. It is equivalent to **-title**.

-n *string* This option specifies the icon name for *xterm*'s windows. It is shorthand for specifying the "**iconName*" resource. Note that this is not the same as the toolkit option **-name** (see below). The default icon name is the application name.

-r This option indicates that reverse video should be simulated by swapping the foreground and background colors. It is equivalent to **-reversevideo** or **-rv**.

-w *number*
This option specifies the width in pixels of the border surrounding the window. It is equivalent to **-borderwidth** or **-bw**.

The following standard X Toolkit command line arguments are commonly used with *xterm*:

-bg *color*
This option specifies the color to use for the background of the window. The default is "white."

-bd *color*
This option specifies the color to use for the border of the window. The default is "black."

-bw *number*
This option specifies the width in pixels of the border surrounding the window.

-fg *color* This option specifies the color to use for displaying text. The default is "black".

-fn *font* This option specifies the font to be used for displaying normal text. The default is *fixed*.

-name *name*
This option specifies the application name under which resources are to be obtained, rather than the default executable file name. *Name* should not contain "." or "*" characters.

-title *string*
This option specifies the window title string, which may be displayed by window managers if the user so chooses. The default title is the command line specified after the **-e** option, if any, otherwise the application name.

-rv This option indicates that reverse video should be simulated by swapping the foreground and background colors.

-geometry *geometry*
This option specifies the preferred size and position of the VT102 window; see *X(1)*.

-display *display*
This option specifies the X server to contact; see *X(1)*.

-xrm *resourcestring*
This option specifies a resource string to be used. This is especially useful for setting resources that do not have separate command line options.

-iconic This option indicates that *xterm* should ask the window manager to start it as an icon rather than as the normal window.

RESOURCES

The program understands all of the core X Toolkit resource names and classes as well as:

iconGeometry (class **IconGeometry**)
Specifies the preferred size and position of the application when iconified. It is not necessarily obeyed by all window managers.

termName (class **TermName**)
Specifies the terminal type name to be set in the TERM environment variable.

title (class **Title**)
Specifies a string that may be used by the window manager when displaying this application.

ttyModes (class **TtyModes**)
Specifies a string containing terminal setting keywords and the characters to which they may be bound. Allowable keywords include: intr, quit, erase, kill, eof, eol, swtch, start, stop, brk, susp, dsusp, rprnt, flush, weras, and lnext. Control characters may be specified as ^char (e.g. ^c or ^u) and ^? may be used to indicate delete. This is very useful for overriding the default terminal settings without having to do an *stty* every time an *xterm* is started.

utmpInhibit (class **UtmpInhibit**)
Specifies whether or not *xterm* should try to record the user's terminal in */etc/utmp*.

sunFunctionKeys (class **SunFunctionKeys**)
Specifies whether or not Sun Function Key escape codes should be generated for function keys instead of standard escape sequences.

The following resources are specified as part of the *vt100* widget (class *VT100*):

allowSendEvents (class **AllowSendEvents**)
Specifies whether or not synthetic key and button events (generated using the X protocol SendEvent request) should be interpreted or discarded. The default is "false" meaning they are discarded. Note that allowing such events creates a very large security hole.

alwaysHighlight (class **AlwaysHighlight**)
Specifies whether or not *xterm* should always display a highlighted text cursor. By default, a hollow text cursor is displayed whenever the pointer moves out of the window or the window loses the input focus.

boldFont (class **Font**)
Specifies the name of the bold font to use instead of overstriking.

c132 (class **C132**)
Specifies whether or not the VT102 DECCOLM escape sequence should be honored. The default is "false."

charClass (class **CharClass**)
Specifies comma-separated lists of character class bindings of the form *[low-]high:value*. These are used in determining which sets of characters should be treated the same when doing cut and paste. See the section on specifying character classes.

curses (class **Curses**)
Specifies whether or not the last column bug in *curses*(3x) should be worked around. The default is "false."

background (class **Background**)
Specifies the color to use for the background of the window. The default is "white."

foreground (class **Foreground**)
Specifies the color to use for displaying text in the window. Setting the class name instead of the instance name is an easy way to have everything that would normally appear in the "text" color change color. The default is "black."

cursorColor (class **Foreground**)
Specifies the color to use for the text cursor. The default is "black."

eightBitInput (class **EightBitInput**)
Specifies how a character input from the keyboard via the action, **insert-eight-bit()** is modified (**insert-eight-bit()** is normally bound to Meta<KeyPress>). If "true", the character always has its high bit set on. If "false", ESC is prepended to the character without modifying it. The default is "true."

eightBitOutput (class **EightBitOutput**)
Specifies whether or not eight-bit characters sent from the host should be accepted as is or stripped when printed. The default is "true."

font (class **Font**)
Specifies the name of the normal font. The default is "fixed."

font1 (class **Font1**)
Specifies the name of the first alternate font.

font2 (class **Font2**)
Specifies the name of the second alternate font.

font3 (class **Font3**)
Specifies the name of the third alternate font.

font4 (class **Font4**)
Specifies the name of the fourth alternate font.

geometry (class **Geometry**)
Specifies the preferred size and position of the VT102 window.

internalBorder (class **BorderWidth**)
Specifies the number of pixels between the characters and the window border. The default is 2.

jumpScroll (class **JumpScroll**)
Specifies whether or not jump scroll should be used. The default is "true".

logFile (class **Logfile**)
Specifies the name of the file to which a terminal session is logged. The default is "**XtermLog.***XXXXX*" (where *XXXXX* is the process id of *xterm*).

logging (class **Logging**)
Specifies whether or not a terminal session should be logged. The default is "false."

logInhibit (class **LogInhibit**)
Specifies whether or not terminal session logging should be inhibited. The default is "false."

loginShell (class **LoginShell**)
Specifies whether or not the shell to be run in the window should be started as a login shell. The default is "false."

marginBell (class **MarginBell**)
Specifies whether or not the bell should be run when the user types near the right margin. The default is "false."

multiScroll (class **MultiScroll**)
Specifies whether or not asynchronous scrolling is allowed. The default is "false."

multiClickTime (class **MultiClickTime**)
Specifies the maximum time in milliseconds between multi-clock select events. The default is 250 milliseconds.

multiScroll (class **MultiScroll**)
Specifies whether or not scrolling should be done asynchronously. The default is "false."

nMarginBell (class **Column**)
Specifies the number of characters from the right margin at which the margin bell should

be run, when enabled.

pointerColor (class **Foreground**)
Specifies the foreground color of the pointer. The default is "XtDefaultForeground."

pointerColorBackground (class **Background**)
Specifies the background color of the pointer. The default is "XtDefaultBackground."

pointerShape (class **Cursor**)
Specifies the name of the shape of the pointer. The default is "xterm."

reverseVideo (class **ReverseVideo**)
Specifies whether or not reverse video should be simulated. The default is "false."

reverseWrap (class **ReverseWrap**)
Specifies whether or not reverse-wraparound should be enabled. The default is "false."

autoWrap (class **AutoWrap**)
Specifies whether or not auto-wraparound should be enabled. The default is "true."

saveLines (class **SaveLines**)
Specifies the number of lines to save beyond the top of the screen when a scrollbar is turned on. The default is 64.

scrollBar (class **ScrollBar**)
Specifies whether or not the scrollbar should be displayed. The default is "false."

scrollTtyOutput (class **ScrollCond**)
Specifies whether or not output to the terminal should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "true."

scrollKey (class **ScrollCond**)
Specifies whether or not pressing a key should automatically cause the scrollbar to go to the bottom of the scrolling region. The default is "false."

scrollLines (class **ScrollLines**)
Specifies the number of lines that the *scroll-back* and *scroll-forw* actions should use as a default. The default value is 1.

signalInhibit (class **SignalInhibit**)
Specifies whether or not the entries in the "xterm X11" menu for sending signals to *xterm* should be disallowed. The default is "false."

tekGeometry (class **Geometry**)
Specifies the preferred size and position of the Tektronix window.

tekInhibit (class **TekInhibit**)
Specifies whether or not Tektronix mode should be disallowed. The default is "false."

tekSmall (class **TekSmall**)
Specifies whether or not the Tektronix mode window should start in its smallest size if no explicit geometry is given. This is useful when running *xterm* on displays with small screens. The default is "false."

tekStartup (class **TekStartup**)
Specifies whether or not *xterm* should start up in Tektronix mode. The default is "false."

titeInhibit (class **TiteInhibit**)
Specifies whether or not *xterm* should remove remove *ti* or *te* termcap entries (used to switch between alternate screens on startup of many screen-oriented programs) from the TERMCAP string.

translations (class **Translations**)
Specifies the key and button bindings for menus, selections, "programmed strings", etc. See ACTIONS below.

visualBell (class **VisualBell**)
Specifies whether or not a visible bell (i.e. flashing) should be used instead of an audible bell when Control-G is received. The default is "false."

waitForMap (class **WaitForMap**)
Specifies whether or not *xterm* should wait for the initial window map before starting the subprocess. The default is "false."

The following resources are specified as part of the *tek4014* widget (class *Tek4014*):

width (class **Width**)
Specifies the width of the Tektronix window in pixels.

height (class **Height**)
Specifies the height of the Tektronix window in pixels.

fontLarge (class **Font**)
Specifies the large font to use in the Tektronix window.

font2 (class **Font**)
Specifies font number 2 to use in the Tektronix window.

font3 (class **Font**)
Specifies font number 2 font to use in the Tektronix window.

fontSmall (class **Font**)
Specifies the small font to use in the Tektronix window.

The resources that may be specified for the various menus are described in the documentation for the Athena **SimpleMenu** widget. The name and classes of the entries in each of the menus are listed below.

The *mainMenu* has the following entries:

securekbd (class **SmeBSB**)
This entry invokes the **secure()** action.

allowsends (class **SmeBSB**)
This entry invokes the **allow-send-events(toggle)** action.

logging (class **SmeBSB**)
This entry invokes the **set-logging(toggle)** action.

redraw (class **SmeBSB**)
This entry invokes the **redraw()** action.

line1 (class **SmeLine**)
This is a separator.

suspend (class **SmeBSB**)
This entry invokes the **send-signal(suspend)** action on systems that support job control.

continue (class **SmeBSB**)
This entry invokes the **send-signal(cont)** action on systems that support job control.

interrupt (class **SmeBSB**)
This entry invokes the **send-signal(int)** action.

hangup (class **SmeBSB**)
This entry invokes the **send-signal(hup)** action.

terminate (class **SmeBSB**)
This entry invokes the **send-signal(term)** action.

kill (class **SmeBSB**)
This entry invokes the **send-signal(kill)** action.

line2 (class **SmeLine**)
This is a separator.

quit (class **SmeBSB**)
This entry invokes the **quit()** action.

The *vtMenu* has the following entries:

scrollbar (class **SmeBSB**)
This entry invokes the **set-scrollbar(toggle)** action.

jumpscroll (class **SmeBSB**)
This entry invokes the **set-jumpscroll(toggle)** action.

reversevideo (class **SmeBSB**)
This entry invokes the **set-reverse-video(toggle)** action.

autowrap (class **SmeBSB**)
This entry invokes the **set-autowrap(toggle)** action.

reversewrap (class **SmeBSB**)
This entry invokes the **set-reversewrap(toggle)** action.

autolinefeed (class **SmeBSB**)
This entry invokes the **set-autolinefeed(toggle)** action.

appcursor (class **SmeBSB**)
This entry invokes the **set-appcursor(toggle)** action.

appkeypad (class **SmeBSB**)
This entry invokes the **set-appkeypad(toggle)** action.

scrollkey (class **SmeBSB**)
This entry invokes the **set-scroll-on-key(toggle)** action.

scrollttyoutput (class **SmeBSB**)
This entry invokes the **set-scroll-on-tty-output(toggle)** action.

allow132 (class **SmeBSB**)
This entry invokes the **set-allow132(toggle)** action.

cursesemul (class **SmeBSB**)
This entry invokes the **set-cursesemul(toggle)** action.

visualbell (class **SmeBSB**)
This entry invokes the **set-visualbell(toggle)** action.

marginbell (class **SmeBSB**)
This entry invokes the **set-marginbell(toggle)** action.

altscreen (class **SmeBSB**)
This entry is currently disabled.

line1 (class **SmeLine**)
This is a separator.

softreset (class **SmeBSB**)
This entry invokes the **soft-reset()** action.

hardreset (class **SmeBSB**)
This entry invokes the **hard-reset()** action.

line2 (class **SmeLine**)
This is a separator.

tekshow (class **SmeBSB**)
This entry invokes the **set-visibility(tek,toggle)** action.

tekmode (class **SmeBSB**)
This entry invokes the **set-terminal-type(tek)** action.

vthide (class **SmeBSB**)
This entry invokes the **set-visibility(vt,off)** action.

The *fontMenu* has the following entries:

fontdefault (class **SmeBSB**)
This entry invokes the **set-vt-font(d)** action.

font1 (class **SmeBSB**)
This entry invokes the **set-vt-font(1)** action.

font2 (class **SmeBSB**)
This entry invokes the **set-vt-font(2)** action.

font3 (class **SmeBSB**)
This entry invokes the **set-vt-font(3)** action.

font4 (class **SmeBSB**)
This entry invokes the **set-vt-font(4)** action.

fontescape (class **SmeBSB**)
This entry invokes the **set-vt-font(e)** action.

fontsel (class **SmeBSB**)
This entry invokes the **set-vt-font(s)** action.

The *tekMenu* has the following entries:

tektextlarge (class **SmeBSB**)
This entry invokes the **set-tek-text(l)** action.

tektext2 (class **SmeBSB**)
This entry invokes the **set-tek-text(2)** action.

tektext3 (class **SmeBSB**)
This entry invokes the **set-tek-text(3)** action.

tektextsmall (class **SmeBSB**)
This entry invokes the **set-tek-text(s)** action.

line1 (class **SmeLine**)
This is a separator.

tekpage (class **SmeBSB**)
This entry invokes the **tek-page()** action.

tekreset (class **SmeBSB**)
This entry invokes the **tek-reset()** action.

tekcopy (class **SmeBSB**)
This entry invokes the **tek-copy()** action.

line2 (class **SmeLine**)
This is a separator.

vtshow (class **SmeBSB**)
This entry invokes the **set-visibility(vt,toggle)** action.

vtmode (class **SmeBSB**)
This entry invokes the **set-terminal-type(vt)** action.

tekhide (class **SmeBSB**)
This entry invokes the **set-visibility(tek,toggle)** action.

The following resources are useful when specified for the Athena Scrollbar widget:

thickness (class **Thickness**)
Specifies the width in pixels of the scrollbar.

background (class **Background**)
Specifies the color to use for the background of the scrollbar.

foreground (class **Foreground**)
Specifies the color to use for the foreground of the scrollbar. The "thumb" of the scrollbar is a simple checkerboard pattern alternating pixels for foreground and

background color.

EMULATIONS

The VT102 emulation is fairly complete, but does not support the blinking character attribute nor the double-wide and double-size character sets. *Termcap*(5) entries that work with *xterm* include "xterm", "vt102", "vt100" and "ansi", and *xterm* automatically searches the termcap file in this order for these entries and then sets the "TERM" and the "TERMCAP" environment variables.

Many of the special *xterm* features (like logging) may be modified under program control through a set of escape sequences different from the standard VT102 escape sequences. (See the *"Xterm Control Sequences"* document.)

The Tektronix 4014 emulation is also fairly good. Four different font sizes and five different lines types are supported. The Tektronix text and graphics commands are recorded internally by *xterm* and may be written to a file by sending the COPY escape sequence (or through the **Tektronix** menu; see below). The name of the file will be "COPY*yy-MM-dd.hh:mm:ss*", where *yy*, *MM*, *dd*, *hh*, *mm* and *ss* are the year, month, day, hour, minute and second when the COPY was performed (the file is created in the directory *xterm* is started in, or the home directory for a login *xterm*).

POINTER USAGE

Once the VT102 window is created, *xterm* allows you to select text and copy it within the same or other windows.

The selection functions are invoked when the pointer buttons are used with no modifiers, and when they are used with the "shift" key. The assignment of the functions described below to keys and buttons may be changed through the resource database; see **ACTIONS** below.

Pointer button one (usually left) is used to save text into the cut buffer. Move the cursor to beginning of the text, and then hold the button down while moving the cursor to the end of the region and releasing the button. The selected text is highlighted and is saved in the global cut buffer and made the PRIMARY selection when the button is released. Double-clicking selects by words. Triple-clicking selects by lines. Quadruple-clicking goes back to characters, etc. Multiple-click is determined by the time from button up to button down, so you can change the selection unit in the middle of a selection. If the key/button bindings specify that an X selection is to be made, *xterm* will leave the selected text highlighted for as long as it is the selection owner.

Pointer button two (usually middle) 'types' (pastes) the text from the PRIMARY selection, if any, otherwise from the cut buffer, inserting it as keyboard input.

Pointer button three (usually right) extends the current selection. (Without loss of generality, that is you can swap "right" and "left" everywhere in the rest of this paragraph...) If pressed while closer to the right edge of the selection than the left, it extends/contracts the right edge of the selection. If you contract the selection past the left edge of the selection, *xterm* assumes you really meant the left edge, restores the original selection, then extends/contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple-click to cycle through them.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell, for example, or take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a 'file' whose contents you know. The terminal emulator and other text programs should be treating it as if it were a text file, i.e. the text is delimited by new lines.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved. As more text is saved (up to the maximum), the size of the highlighted area decreases.

Clicking button one with the pointer in the scroll region moves the adjacent line to the top of the display window.

Clicking button three moves the top line of the display window down to the pointer position.

Clicking button two moves the display to a position in the saved text that corresponds to the pointer's position in the scrollbar.

Unlike the VT102 window, the Tektronix window dows not allow the copying of text. It does allow Tektronix GIN mode, and in this mode the cursor will change from an arrow to a cross. Pressing any key will send that key and the current coordinate of the cross cursor. Pressing button one, two, or three will return the letters 'l', 'm', and 'r', respectively. If the 'shift' key is pressed when a pointer button is pressed, the corresponding upper case letter is sent. To distinguish a pointer button from a key, the high bit of the character is set (but this is bit is normally stripped unless the terminal mode is RAW; see *tty*(4) for details).

MENUS

Xterm has four menus, named *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*. Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two section, separated by a horizontal line. The top portion contains various modes that can be altered. A check mark appears next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu are command entries; selecting one of these performs the indicated function.

The **xterm** menu pops up when the "control" key and pointer button one are pressed in a window. The *mainMenu* contains items that apply to both the VT102 and Tektronix windows. The **Secure Keyboard** mode is be used when typing in passwords or other sensitive data in an unsecure environment; see **SECURITY** below. Notable entries in the command section of the menu are the **Continue, Suspend, Interrupt, Hangup, Terminate** and **Kill** which sends the SIGCONT, SIGTSTP, SIGINT, SIGHUP, SIGTERM and SIGKILL signals, respectively, to the process group of the process running under *xterm* (usually the shell). The **Continue** function is especially useful if the user has accidentally typed CTRL-Z, suspending the process.

The *vtMenu* sets various modes in the VT102 emulation, and is popped up when the "control" key and pointer button two are pressed in the VT102 window. In the command section of this menu, the soft reset entry will reset scroll regions. This can be convenient when some program has left the scroll regions set incorrectly (often a problem when using VMS or TOPS-20). The full reset entry will clear the screen, reset tabs to every eight columns, and reset the terminal modes (such as wrap and smooth scroll) to their initial states just after *xterm* has finished processing the command line options.

The *fontMenu* sets the font used in the VT102 window.

The *tekMenu* sets various modes in the Tektronix emulation, and is popped up when the "control" key and pointer button two are pressed in the Tektronix window. The current font size is checked in the modes section of the menu. The **PAGE** entry in the command section clears the Tektronix window.

SECURITY

X environments differ in their security consciousness. MIT servers, run under *xdm*, are capable of using a "magic cookie" authorization scheme that can provide a reasonable level of security for many people. If your server is only using a host-based mechanism to control access to the server (see *xhost(1)*), then if you enable access for a host and other users are also permitted to run clients on that same host, there is every possibility that someone can run an application that will use the basic services of the X protocol to snoop on your activities, potentially capturing a transcript of everything you type at the keyboard. This is of particular concern when you want to type in a password or other sensitive data. The best solution to this problem is to use a better authorization mechanism that host-based control, but a simple mechanism exists for protecting keyboard input in *xterm*.

The **xterm** menu (see **MENUS** above) contains a **Secure Keyboard** entry which, when enabled, ensures that all keyboard input is directed *only* to *xterm* (using the GrabKeyboard protocol request). When an application prompts you for a password (or other sensitive data), you can enable **Secure Keyboard** using the menu, type in the data, and then disable **Secure Keyboard** using the menu again. Only one X client at a time can secure the keyboard, so when you attempt to enable **Secure Keyboard** it may fail. In this case, the bell will sound. If the **Secure Keyboard** succeeds, the foreground and background colors will be exchanged (as if you selected the **Reverse Video** entry in the **Modes** menu); they will be exchanged again when you exit secure mode. If the colors do *not* switch, then you should be *very* suspicious that you are being spoofed. If the application you are running displays a prompt before asking for the password, it is safest to enter secure

mode *before* the prompt gets displayed, and to make sure that the prompt gets displayed correctly (in the new colors), to minimize the probability of spoofing. You can also bring up the menu again and make sure that a check mark appears next to the entry.

Secure Keyboard mode will be disabled automatically if your xterm window becomes iconified (or otherwise unmapped), or if you start up a reparenting window manager (that places a title bar or other decoration around the window) while in **Secure Keyboard** mode. (This is a feature of the X protocol not easily overcome.) When this happens, the foreground and background colors will be switched back and the bell will sound in warning.

CHARACTER CLASSES

Clicking the middle mouse button twice in rapid succession will cause all characters of the same class (e.g. letters, white space, punctuation) to be selected. Since different people have different preferences for what should be selected (for example, should filenames be selected as a whole or only the separate subnames), the default mapping can be overridden through the use of the *charClass* (class *CharClass*) resource.

This resource is simply a list of *range:value* pairs where the range is either a single number or *low-high* in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The *value* is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is:

```
static int charClass[128] = {
/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */
   32,   1,   1,   1,   1,   1,   1,   1,
/*  BS   HT   NL   VT   NP   CR   SO   SI */
    1,  32,   1,   1,   1,   1,   1,   1,
/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */
    1,   1,   1,   1,   1,   1,   1,   1,
/* CAN   EM  SUB  ESC   FS   GS   RS   US */
    1,   1,   1,   1,   1,   1,   1,   1,
/*  SP    !    "    #    $    %    &    ' */
   32,  33,  34,  35,  36,  37,  38,  39,
/*   (    )    *    +    ,    -    .    / */
   40,  41,  42,  43,  44,  45,  46,  47,
/*   0    1    2    3    4    5    6    7 */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   8    9    :    ;    <    =    >    ? */
   48,  48,  58,  59,  60,  61,  62,  63,
/*   @    A    B    C    D    E    F    G */
   64,  48,  48,  48,  48,  48,  48,  48,
/*   H    I    J    K    L    M    N    O */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   P    Q    R    S    T    U    V    W */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   X    Y    Z    [    \    ]    ^    _ */
   48,  48,  48,  91,  92,  93,  94,  48,
/*   `    a    b    c    d    e    f    g */
   96,  48,  48,  48,  48,  48,  48,  48,
/*   h    i    j    k    l    m    n    o */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   p    q    r    s    t    u    v    w */
   48,  48,  48,  48,  48,  48,  48,  48,
/*   x    y    z    {    |    }    ~  DEL */
   48,  48,  48, 123, 124, 125, 126,   1};
```

For example, the string "33:48,37:48,45-47:48,64:48" indicates that the exclamation mark, percent sign, dash, period, slash, and ampersand characters should be treated the same way as characters

and numbers. This is very useful for cutting and pasting electronic mailing addresses and filenames.

ACTIONS

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input, by changing the translations for the vt100 or tek4014 widgets. Changing the translations for events other than key and button events is not expected, and will cause unpredictable behavior. The following actions are provided for using within the *vt100* or *tek4014* **translations** resources:

bell([*percent*]**)**
This action rings the keyboard bell at the specified percentage above or below the base volume.

ignore() This action ignores the event but checks for special pointer position escape sequences.

insert() This action is a synonym for **insert-seven-bit()**

insert-seven-bit()
This action inserts the character or string associated with the keysym that was pressed. The character or string is obtained using the standard interpretation of Shift, Lock and group modifiers as defined by the X Protocol Specification.

insert-eight-bit()
This action is the same as **insert-seven-bit()** except that single characters obtained using the standard interpretation are modified according to the resource, **eightBitInput**.

insert-selection(*sourcename* [, ...]**)**
This action inserts the string found in the selection or cutbuffer indicated by *sourcename*. Sources are checked in the order given (case is significant) until one is found. Commonly-used selections include: *PRIMARY*, *SECONDARY*, and *CLIPBOARD*. Cut buffers are typically named *CUT_BUFFER0* through *CUT_BUFFER7*.

keymap(*name***)**
This action dynamically defines a new translation table whose resource name is *name* with the suffix *Keymap* (case is significant). The name *None* restores the original translation table.

popup-menu(*menuname***)**
This action displays the specified popup menu. Valid names (case is significant) include: *mainMenu*, *vtMenu*, *fontMenu*, and *tekMenu*.

secure() This action toggles the *Secure Keyboard* mode described in the section named SECURITY, and is invoked from the **securekbd** entry in *mainMenu*.

select-start()
This action begins text selection at the current pointer location. See the section on POINTER USAGE for information on making selections.

select-extend()
This action tracks the pointer and extends the selection. It should only be bound to Motion events.

select-end(*destname* [, ...]**)**
This action puts the currently selected text into all of the selections or cutbuffers specified by *destname*.

select-cursor-start()
This action is similar to **select-start** except that it begins the selection at the current text cursor position.

select-cursor-end(*destname* [, ...]**)**
This action is similar to **select-end** except that it should be used with **select-cursor-start.**

set-vt-font(*d/1/2/3/4/e/s* [*normalfont* [, *boldfont*]]**)**
This action sets the font or fonts currently being used in the VT102 window. The first argument is a single character that specifies the font to be used: *d* or *D* indicate the default font (the font initially used when *xterm* was started), *1* through *4* indicate the

fonts specified by the *font1* through *font4* resources, *e* or *E* indicate the normal and bold fonts that may be set through escape codes (or specified as the second and third action arguments, respectively), and *i* or *I* indicate the font selection (as made by programs such as *xfontsel(1)*) indicated by the second action argument.

start-extend()
This action is similar to **select-start** except that the selection is extended to the current pointer location.

start-cursor-extend()
This action is similar to **select-extend** except that the selection is extended to the current text cursor position.

string(*string*)
This action inserts the specified text string as if it had been typed. Quotation is necessary if the string contains whitespace or non-alphanumeric characters. If the string argument begins with the characters "0x", it is interpreted as a hex character constant.

scroll-back(*count* [,*units*])
This action scrolls the text window backward so that text that had previously scrolled off the top of the screen is now visible. The *count* argument indicates the number of *units* (which may be *page*, *halfpage*, *pixel*, or *line*) by which to scroll.

scroll-forw(*count* [,*units*])
This action scrolls is similar to **scroll-back** except that it scrolls the other direction.

allow-send-events(*on/off/toggle*)
This action set or toggles the **allowSendEvents** resource and is also invoked by the **allowsends** entry in *mainMenu*.

set-logging(*on/off/toggle*)
This action toggles the **logging** resource and is also invoked by the **logging** entry in *mainMenu*.

redraw()
This action redraws the window and is also invoked by the *redraw* entry in *mainMenu*.

send-signal(*signame*)
This action sends the signal named by *signame* (which may also be a number) to the *xterm* subprocess (the shell or program specified with the *-e* command line option) and is also invoked by the **suspend**, **continue**, **interrupt**, **hangup**, **terminate**, and *kill* entries in *mainMenu*. Allowable signal names are (case is not significant): *suspend*, *tstp* (if supported by the operating system), *cont* (if supported by the operating system), *int*, *hup*, *term*, and *kill*.

quit() This action sends a SIGHUP to the subprogram and exits. It is also invoked by the **quit** entry in *mainMenu*.

set-scrollbar(*on/off/toggle*)
This action toggles the **scrollbar** resource and is also invoked by the **scrollbar** entry in *vtMenu*.

set-jumpscroll(*on/off/toggle*)
This action toggles the **jumpscroll** resource and is also invoked by the **jumpscroll** entry in *vtMenu*.

set-reverse-video(*on/off/toggle*)
This action toggles the *reverseVideo* resource and is also invoked by the **reversevideo** entry in *vtMenu*.

set-autowrap(*on/off/toggle*)
This action toggles automatic wrapping of long lines and is also invoked by the **autowrap** entry in *vtMenu*.

set-reversewrap(*on/off/toggle*)
This action toggles the **reverseWrap** resource and is also invoked by the **reversewrap** entry in *vtMenu*.

set-autolinefeed(*on/off/toggle*)
This action toggles automatic insertion of linefeeds and is also invoked by the **autolinefeed** entry in *vtMenu*.

set-appcursor(*on/off/toggle*)
This action toggles the handling Application Cursor Key mode and is also invoked by the Bappcursor entry in *vtMenu*.

set-appkeypad(*on/off/toggle*)
This action toggles the handling of Application Keypad mode and is also invoked by the **appkeypad** entry in *vtMenu*.

set-scroll-on-key(*on/off/toggle*)
This action toggles the **scrollKey** resource and is also invoked from the **scrollkey** entry in *vtMenu*.

set-scroll-on-tty-output(*on/off/toggle*)
This action toggles the **scrollTtyOutput** resource and is also invoked from the **scrolltyoutput** entry in *vtMenu*.

set-allow132(*on/off/toggle*)
This action toggles the **c132** resource and is also invoked from the **allow132** entry in *vtMenu*.

set-cursesemul(*on/off/toggle*)
This action toggles the **curses** resource and is also invoked from the **cursesemul** entry in *vtMenu*.

set-visual-bell(*on/off/toggle*)
This action toggles the **visualBell** resource and is also invoked by the **visualbell** entry in *vtMenu*.

set-marginbell(*on/off/toggle*)
This action toggles the **marginBell** resource and is also invoked from the **marginbell** entry in *vtMenu*.

set-altscreen(*on/off/toggle*)
This action toggles between the alternative and current screens.

soft-reset()
This action resets the scrolling region and is also invoked from the **softreset** entry in *vtMenu*.

hard-reset()
This action resets the scrolling region, tabs, window size, and cursor keys and clears the screen. It is also invoked from the **hardreset** entry in *vtMenu*.

set-terminal-type(*type*)
This action directs output to either the *vt* or *tek* windows, according to the *type* string. It is also invoked by the **tekmode** entry in *vtMenu* and the **vtmode** entry in *tekMenu*.

set-visibility(*vt/tek,on/off/toggle*)
This action controls whether or not the *vt* or *tek* windows are visible. It is also invoked from the **tekshow** and **vthide** entries in *vtMenu* and the **vtshow** and **tekhide** entries in *tekMenu*.

set-tek-text(*large/2/3/small*)
This action sets font used in the Tektronix window to the value of the resources **tektextlarge, tektext2, tektext3**, and **tektextsmall** according to the argument. It is also by the entries of the same names as the resources in *tekMenu*.

tek-page()
This action clears the Tektronix window and is also invoked by the **tekpage** entry in *tekMenu*.

tek-reset()
This action resets the Tektronix window and is also invoked by the *tekreset* entry in

tekMenu.

tek-copy()
This action copies the escape codes used to generate the current window contents to a file in the current directory beginning with the name COPY. It is also invoked from the *tekcopy* entry in *tekMenu*.

The Tektronix window also has the following action:

gin-press(*l*/*L*/*m*/*M*/*r*/*R*)
This action send the indicated graphics input code.

The default bindings in the VT102 window are:

```
Shift <KeyPress> Prior:          scroll-back(1,halfpage) \n\
 Shift <KeyPress> Next:          scroll-forw(1,halfpage) \n\
Shift <KeyPress> Select:         select-cursor-start() \
                                 select-cursor-end(PRIMARY, CUT_BUFFER0)

Shift <KeyPress> Insert:         insert-selection(PRIMARY, CUT_BUFFER0) \
     ~Meta<KeyPress>:            insert-seven-bit() \n\
      Meta<KeyPress>:            insert-eight-bit() \n\
  Ctrl ~Meta<Btn1Down>:          popup-menu(mainMenu) \n\
    ~Meta <Btn1Down>:            select-start() \n\
   ~Meta <Btn1Motion>:           select-extend() \n\
 Ctrl ~Meta <Btn2Down>:          popup-menu(vtMenu) \n\
~Ctrl ~Meta <Btn2Down>:          ignore() \n\
  ~Ctrl ~Meta <Btn2Up>:          insert-selection(PRIMARY, CUT_BUFFER0) \
 Ctrl ~Meta <Btn3Down>:          popup-menu(fontMenu) \n\
~Ctrl ~Meta <Btn3Down>:          start-extend() \n\
   ~Meta <Btn3Motion>:           select-extend() \n\
  ~Ctrl ~Meta <BtnUp>:           select-end(PRIMARY, CUT_BUFFER0) \n\
        <BtnDown>:               bell(0)
```

The default bindings in the Tektronix window are:

```
     ~Meta<KeyPress>:            insert-seven-bit() \n\
      Meta<KeyPress>:            insert-eight-bit()\n\
 Ctrl ~Meta<Btn1Down>:           popup-menu(mainMenu) \n\
Ctrl ~Meta <Btn2Down>:           popup-menu(tekMenu) \n\
Shift ~Meta<Btn1Down>:           gin-press(L) \n\
    ~Meta<Btn1Down>:             gin-press(l) \n\
Shift ~Meta<Btn2Down>:           gin-press(M) \n\
    ~Meta<Btn2Down>:             gin-press(m) \n\
Shift ~Meta<Btn3Down>:           gin-press(R) \n\
    ~Meta<Btn3Down>:             gin-press(r)
```

Below is a sample how of the **keymap()** action is used to add special keys for entering commonly-typed works:

```
*VT100.Translations: #override <Key>F13: keymap(dbx)
*VT100.dbxKeymap.translations: \
        <Key>F14:       keymap(None) \n\
        <Key>F17:       string("next") string(0x0d) \n\
        <Key>F18:       string("step") string(0x0d) \n\
        <Key>F19:       string("continue") string(0x0d) \n\
        <Key>F20:       string("print ") insert-selection(PRIMARY, CUT_BUFFER0)
```

OTHER FEATURES
Xterm automatically highlights the window border and text cursor when the pointer enters the

window (selected) and unhighlights them when the pointer leaves the window (unselected). If the window is the focus window, then the window is highlighted no matter where the pointer is.

In VT102 mode, there are escape sequences to activate and deactivate an alternate screen buffer, which is the same size as the display area of the window. When activated, the current screen is saved and replace with the alternate screen. Saving of lines scrolled off the top of the window is disabled until the normal screen is restored. The *termcap*(5) entry for *xterm* allows the visual editor *vi*(1) to switch to the alternate screen for editing, and restore the screen on exit.

In either VT102 or Tektronix mode, there are escape sequences to change the name of the windows and to specify a new log file name.

ENVIRONMENT

Xterm sets the environment variables "TERM" and "TERMCAP" properly for the size window you have created. It also uses and sets the environment variable "DISPLAY" to specify which bit map display terminal to use. The environment variable "WINDOWID" is set to the X window id number of the *xterm* window.

SEE ALSO

resize(1), X(1), pty(4), tty(4)
Xterm Control Sequences

BUGS

The *Xterm Control Sequences* document has yet to be converted from X10. The old version, along with a first stab at an update, are available in the sources.

The class name is *XTerm* instead of *Xterm*.

Xterm will hang forever if you try to paste too much text at one time. It is both producer and consumer for the pty and can deadlock.

Variable-width fonts are not handled.

This program still needs to be rewritten. It should be split into very modular sections, with the various emulators being completely separate widgets that don't know about each other. Ideally, you'd like to be able to pick and choose emulator widgets and stick them into a single control widget.

The focus is considered lost if some other client (e.g., the window manager) grabs the pointer; it is difficult to do better without an addition to the protocol.

There needs to be a dialog box to allow entry of log file name and the COPY file name.

Many of the options are not resettable after *xterm* starts.

The Tek widget does not support key/button re-binding.

COPYRIGHT

AUTHORS

Far too many people, including:

Loretta Guarino Reid (DEC-UEG-WSL), Joel McCormack (DEC-UEG-WSL), Terry Weissman (DEC-UEG-WSL), Edward Moy (Berkeley), Ralph R. Swick (MIT-Athena), Mark Vandevoorde (MIT-Athena), Bob McNamara (DEC-MAD), Jim Gettys (MIT-Athena), Bob Scheifler (MIT X Consortium), Doug Mink (SAO), Steve Pitschke (Stellar), Ron Newman (MIT-Athena), Jim Fulton (MIT X Consortium), Dave Serisky (HP)

Glossary

accelerator
A key or sequence of keys (typically a modifier key and some other key) that provides a "shortcut," for accessing functionality.

action definition database
The files that define the actions known to the system. The actions provide the ability to represent commands graphically. They also provide the ability associate default commands with particular file types, and to execute commands from the file manager Actions menu.

active window
The terminal window where what you type appears. If there is no active window, what you type is lost. Only one terminal window can be active at a time.

ampersand (&)
The character placed at the end of a command to specify that the client started by the command should be started as a background process. The command can be typed after the command-line prompt or included in a file such as `.vueprofile`.

application program
A computer program that performs some useful function, such as word processing or data base management.

application server
A computer used solely to provide processing power for application programs.

backdrop
The pixmap used as the visual background for a workspace.

background process
A process that doesn't require the total attention of the computer for operation. Background processing enables the operating system to execute more than one program or command at a time. As a general rule, all clients should be run as background processes.

bitmap
Generally speaking, an array of data bits used for graphic images. Strictly speaking, a pixmap of depth one (capable of 2-color images).

bitmap device
An output device that displays bitmaps. The CRT monitor of your system is a bitmap device.

Broadcast Message Server
A component of HP VUE that provides communication between HP VUE clients.

buffer
An area used for storage.

button
A button on a mouse pointing device. Mouse buttons can be mapped to the keyboard.

button binding
Association of a mouse button operation with a window manager function. For example, pressing button 3 on a window frame displays the system menu.

button mapping
Association of a button number with a physical mouse button.

click
To press *and release* a mouse button. The term comes from the fact that pressing and releasing the buttons of most mice makes a clicking sound.

click
To press *and release* a mouse button. The term comes from the fact that pressing and releasing the buttons of most mice makes a clicking sound.

client
A program written specifically for the X Window System. Some clients make their own windows. Other clients are utility programs.

color palette
A collection of colorsets.

color set
A collection of colors that define the foreground, background, and shadow colors of a particular display element. A color palette consists of a collection of color sets.

combined mode
A combination of image and overlay planes in which a single display has a single screen that is a combination of the image and overlay planes.

command-line prompt
A command-line prompt shows that the computer is ready to accept your commands. Each terminal emulation window has a command-line prompt that acts just like the command-line prompt you see on the screen immediately after login. Usually the command-line prompt is either a **$** (for Bourne and Korn shells) or a **%** (for C shells), but it can be modified. One popular modification is to print the current working directory and the history stack number before the **$** or **%**. You can find the command-line prompt by pressing Return several times. Everytime you press Return, HP-UX prints the prompt.

cluster
A network of computers in which only one computer has file-system disk drives attached to it.

current session
The session to which you are logged in at a particular time.

cut buffer
A buffer (memory area) that holds text that has been deleted from a file.

default
An appearance or behavior that occurs if no other option is explicitly specified.

depth
The number of planes in a set of planes. For example, a set of 12 image planes would have a depth of 12.

dialog box
Sometimes called a secondary window, the dialog box is called by the user from the application's main window. A dialog box contains controls or settings, and sometimes prompts for text entry.

diskless cluster
The networking of several systems (SPUs) together to share a common hard disk for storage of data and programs.

display
Strictly speaking, the combination of a keyboard, mouse, and one or more screens that provide input and output services to a system. While "display" is sometimes used to mean just the CRT screen, a display, as defined here, can actually include more than one physical screen.

display server
In the X Window System, the display server is the software that controls the communication between client programs and the display (keyboard, mouse, and screen combination).

double buffering
A term describing the method used by Starbase wherein half of the color planes on a monitor are used to display to the screen and the other half are used to compute and draw the next screen display. This provides smooth motion for animation and it is faster. However, it does reduce the number of colors that are available for display on the screen at one time.

double-click
To press *and release* a mouse button twice in rapid succession.

drag
To press *and hold down* a mouse button while moving the mouse on the desktop (and the pointer on the screen). Typically, dragging is used with menu selecting, moving, and resizing operations.

drop
Releasing an icon that has been "dragged" to a new position. To drop the icon, release the mouse button.

environment variable
An element of your workstation software that can be set to a value, then read by applications to influence their behavior.

file manager
The HP VUE application that allows you to manage your files and directories, and set viewing preferences.

file server
A computer whose primary task is to control the storage and retrieval of data from hard disks. Any number of other computers can be linked to the file server in order to use it to access data. This means that less storage space is required on the individual computer.

file type database
The files that defines the file types recognized by the system. Different file types can have different file manager icons and default actions.

fonts
A font is a style of printed text characters. Times Roman is the font used for most newspaper text; Helvetica is the font used for most newspaper headlines.

foreground process
A process that has the terminal window's attention. When a program is run in a window as a foreground process (as opposed to a background process), the terminal window cannot be used for other operations until the process is terminated.

graphical user interface
A form of communication between people and computers that uses graphics-oriented software such as windows, menus, and icons, to ease the burden of the interaction.

$HOME
The value of the environment variable representing the home directory.

home directory
The directory in which you are placed after you log in. Typically, this is `/users/`*username*, where *username* is your login name. The home directory is where you keep all "your" files.

home session
A session stored explicitly using the Style Manager Startup dialog box, to which you can always return.

hotspot
The area of a graphical image used as a pointer or cursor that is defined as the "point" of the pointer or cursor.

hpterm
A type of terminal window, sometimes called a "terminal emulator program" that emulates HP2622 terminals, complete with softkeys. `hpterm` is the default terminal emulator for HP-UX and OSF/1 systems running HP VUE.

icon
A small, graphic representation of an object on the root window (typically a terminal window). Objects can be "iconified" (turned into icons) to clear a cluttered workspace and "restored" (returned to their original appearance) as needed. Processes executing in an object continue to execute when the object is iconified.

iconify
The act of turning a window into an icon.

image mode
The default screen mode using multiple image planes for a single screen.

The number of image planes determines the variety of colors that are available to the screen.

image planes
The primary display planes on a device that supports two sets of planes. The other set of display planes is known as the overlay planes. These two sets of planes are treated as two separate screens in stacked mode and one screen in combined mode.

indicator
A control in the workspace manage that indicates the status of a file.

input device
Any of several pieces of equipment used to give information to the system. Examples are the keyboard, a mouse, or a digitizer tablet.

input focus
When a program has input focus, it receives input from the keyboard and mouse.

keyboard binding
Association of a special key press with a window manager function. For example, pressing the special keys Shift Esc displays the system menu of the active window.

label
The text part of an icon.

local access
The ability to run a program on the computer you are currently operating. This is different from remote access, where you run a program on a computer that is physically removed from the one you are operating.

local client
A local client is a program that is running on your `local` computer, the same system that is running your X server.

login
The user's login name.

login manager
The program that controls the initial startup of HP VUE, accepts the user's login and password, and then starts the session manager.

mask
A graphical image used in conjunction with another graphical element to hide unwanted graphical effects.

matte
A border located just inside the window between the client area and the frame. It is used to create a three-dimensional effect for the frame and window.

maximize
To enlarge a window to its maximum size.

maximize button
A window-frame push button used to enlarge the window to its maximum size.

menu
A list of selections from which to make a choice. In a graphical user interface such as the X Window System, menus enable you to control the operation of the system.

menu bar
A area at the top of a window that contains the titles of the pull down menus for that application.

minimize
To turn a window into an icon. The terms minimize and iconify are interchangeable.

modifier key
A key that, when pressed and held along with another key, changes the meaning of the other key. CTRL, Extend char, and Shift are examples of a modifier key.

mouseless operation
Although a mouse makes it easy to use the X Window System, the mouse is not absolutely necessary. The system can be configured to run from the keyboard alone.

multi-tasking
The ability to execute several programs (tasks) simultaneously on the same computer.

node
An address used by the system. For example, each device on the system has its own node. The system looks there whenever it needs to access the device. A node can also be an address on a network, the location of a system.

non-client
A program that is written to run on a terminal and so must be "fooled" by a terminal emulation window into running in the window environment.

normalize
To change an icon back into its "normal" (original) appearance—same as "restore". The opposite of iconify.

overlay planes
The secondary set of display planes on a device that supports two sets of planes. The other set of display planes is known as the image planes. These two sets of planes are treated as two separate screens.

palette
A collection of color sets that creates an attractive color scheme for the display. Each palette has a unique name.

parent window
A window that causes another window to appear. A window that "owns" other windows.

paste
Inserting data into an area.

pixel
Short for "picture element." The individual dots, or components, of a screen. They are arranged in rows and columns and form the images that are displayed on the screen.

pixmap
An array of data bits used for graphics images. Each pixel (picture element) in the map can be several bits deep, resulting in multi-color graphics images.

pointer
Sometimes called the "mouse cursor," the pointer shows the location of the mouse. The pointer's shape depends on its location. In the root window, the pointer is an ×. On a window frame, the pointer is an arrowhead. Inside the frame, the pointer can be an arrowhead (as when it is inside a clock or load histogram frame) or an I-beam (as when it is inside a terminal window).

pointing device
A device that moves the pointer on the display screen. A pointing device is used with a graphical user interface, like HP VUE, that allows you to select and manipulate a graphical object by "pointing" to it. The mouse is probably the most familiar pointing device.

pop-up menu
A menu that provides no visual cue to its presence, but simply pops up when people perform a particular action. Pop-up menus are associated with a particular area of the workspace.

press
Strictly speaking, to hold down a mouse button or a key. Note that to hold down a mouse button *and move* the mouse is called "dragging."

print server
A computer that controls spooling and other printer operations. This permits a large number of individuals to efficiently share printer resources.

pull down menu
A menu that is pulled down from a client application's title bar.

push button
A graphic control that simulates a real-life push button. Use the pointer and mouse to push the button and immediately start an action.

remote access
The ability to run a program on a computer that is physically removed from the one you are currently operating. This is different from local access, where you run a program on the computer that you are operating.

remote client
An X program that is running on a remote system, but the output of the program can be viewed on your terminal.

remote host
A computer physically removed from your own that you can log in to. See chapter 4 for prerequisites for establishing a remote host.

resize handles
The frame part that is used to change the height or width of the window.

resource
That which controls an element of appearance or behavior. Resources are usually named for the elements they control.

resource database
The accumulation of resources for an application. The resources may be obtained from a variety of places. They are managed by the resource manager for the application.

resource files
Files containing resource specifications

resource manager
The component of the X Window System that creates the application's resource database.

RESOURCE_MANAGER property
A property on the root window that is accessed by the resource manager. It is loaded and managed using the `xrdb` client.

restoring
The act of changing an minimized (iconified) or maximized window back to its regular size. The terms restoring and normalizing are usually interchangeable.

.rhosts
A special file used in network environments that enables a remote user to log into your local system without using a password. Obviously, this has a considerable impact on the security of your system.

root menu
The menu associated with the root window. The root menu enables you to control the behavior of your environment.

root window
The root window is what the "screen" (the flat viewing surface of the terminal) becomes when you start X. To a certain extent, you can think of the root as the screen. The root window is the backdrop of your X environment. Although you can hide the root window under terminal windows or other graphic objects, you can never position anything behind the root window. All windows and graphic objects appear "stacked" on the root window.

screen
The physical CRT (Cathode Ray Tube) that displays information from the computer.

screen dump
An operation that captures an image from your screen, saves it in a file, and enables you to send that file to a printer for hardcopy reproduction.

scroll bar
A graphical device used to change your viewpoint of data. A scroll bar consists of a slider, scroll area, and scroll arrows. A person changes the view by sliding the slider up or down in the scroll area or by pressing one of the scroll arrows.

server
A program that controls all access to input devices (typically a mouse and a keyboard) and all access to output devices (typically a display screen). It is

an interface between application programs you run on your system and the system input and output devices.

session
The time between when you log in and when you log out.

session manager
The program that provides the ability to restore the previous or home session the next time the user logs in. When a user logs in, the session manager starts the workspace manager and other programs that were running during a previous session.

stacked mode
A combination of image and overlay planes in which a single display has two "logical" screens, one the image planes, the other the overlay planes. Typically, the image planes are used to display graphics while the overlay planes are used to display text.

style manager
The HP VUE application that provides the ability to customize various aspects of your system, including colors, fonts, the keyboard and mouse, session startup and temination behavior, and access to other machines.

system font
The font used for window titles, menu bar labels, and button labels.

system menu
The menu displayed when you press the system menu button on the HP Window Manager window frame. Every window has a system menu that enables you to control the size, shape, and position of the window.

Term0
An HP level 0 terminal. It is a reference standard that defines basic terminal functions. For more information, see *Term0 Reference* in the HP-UX documentation set.

terminal-based program
A program (non-client) written to be run on a terminal (not in a window). Terminal-based programs must be "fooled" by terminal-emulation clients to run on the X Window System.

terminal emulator
A client program that provides a window within which you can run non-client programs. The non-client program runs just as though it were running from a real terminal rather than a window acting as a terminal.

terminal type
The type of terminal attached to your computer. HP-UX uses the terminal type to set the `TERM` environment variable so that it can communicate with the terminal correctly. The terminal type is usually set at login, but can be set afterward.

terminal window
A terminal window is a window that emulates a complete display terminal. Terminal windows are typically used to "fool" non-client programs into believing they are running in their favorite terminal—not a difficult task in most cases. When not running programs or executing operating system commands, terminal windows display the command-line prompt. Two terminal windows are supplied with X11—`hpterm`, which emulates HP terminals, and `xterm`, which emulates DEC and Tektronix terminals.

text cursor
The line-oriented cursor that appears in a terminal window after the command prompt. The term is used to distinguish the cursor used by a window from the cursor used by the mouse, the pointer.

threshold
The distance the mouse pointer moves before accelerating.

tile
A rectangular area used to cover a surface with a pattern or visual texture. The HP Window Manager supports tiling, enabling users with limited color availability to create new color tiles blended from existing colors.

title bar
The title bar is the rectangular area between the top of the window and the window frame. The title bar contains the title of the window object.

transient window
A window of short duration such as a dialog box. The window is only

displayed for a short time, usually just long enough to get some direction from the user.

user font
The font used for text the user types in.

window
A data structure that represents all or part of the CRT display screen. It contains a two-dimensional array of 16-bit character data words, a cursor, a set of current attributes, and several flags. Visually, a window is represented as a rectangular subset of the display screen.

window-based program
A client or program written for use with the X Window System. The "opposite" of a window-based program is a terminal-based program.

window decoration
The frame and window control buttons that surround windows managed by the HP Window Manager.

window frame
The area surrounding a window. A window frame can consist of a resize border, a window menu button, a title bar, and window control buttons.

window manager
The window manager controls the size, placement, and operation of windows on the root window. The window manager includes the functional window frames that surround each window object as well as a menu for the root window.

window menu
The menu displayed when you press the window menu button on a window frame. It lets you control the size, shape, and position of the window.

window menu button
The control button at the left side of the title bar in the window frame. It is used to display the window menu.

workspace
The workspace is a display-sized windoe onto which application windows are placed. A screen can have multiple workspaces. You use the workspace switch in the workspace manager to switch from one workspace to another.

workspace manager
The portion of HP VUE that displays various controls for manipulating the contents of workspaces.

workspace menu
The menu associated with the workspace window. Same as the "root" menu.

workspace switch
The set of push buttons in the workspace manager that let you switch from one workspace to another.

X0.hosts
A file that tells the X Window System which remote hosts can access the local server and hence the local display.

xclock
An HP VUE client program that displays the time, either analog (hands and dial) or digital (text read out).

xload
An HP VUE client program that displays the work load of the system as a histogram.

xterm
An X11 client program that displays a terminal window that emulates DEC and Tektronix terminals. `xterm` is the default terminal emulator in HP VUE for all operating systems except HP-UX and OSF/1.

Index

D

E

F

L

M

N

O

P

S

T

W

X

Y

Printed in USA 11/91

Reorder No.
B1171-90044

Manufacturing
Part No.
B1171-90044