ULTRIX
Worksystem Software

digital

Introduction to the ULTRIX Worksystem Software Environment

Order Number: AA-MA86D-TE

ULTRIX Worksystem Software

Introduction to the ULTRIX Worksystem Software Environment

Order Number: AA-MA86D-TE
June 1990

Product Version:	ULTRIX Worksystem Software Version 4.0
Operating System and Version:	ULTRIX Version 4.0

digital equipment corporation
maynard, massachusetts

Contents

3 The Programming Environment

4 Customizing Your DECwindows Environment

ndex

:xamples

:igures

Tables

About This Guide

This manual introduces the ULTRIX Worksystem Software environment and describes how to customize its components and applications.

Audience

This manual is intended for all ULTRIX Worksystem Software users.

Organization

This manual consists of four chapters:

- Chapter 1 introduces the ULTRIX Worksystem environment.
- Chapter 2 provides an overview of the user environment and DECwindows applications.
- Chapter 3 provides an overview of the programming environment.
- Chapter 4 describes how to personalize DECwindows applications by modifying entries in the .Xdefaults file.

Related Documentation

You should have the following manuals at hand:

- *ULTRIX Worksystem Software Reference Pages*
- *DECwindows User's Guide*
- *DECwindows Desktop Applications Guide*
- *The Little Gray Book: An ULTRIX Primer*

You may also wish to have the following manuals at hand:

- *XUI Style Guide*
- *Guide to Writing Applications Using XUI Toolkit Widgets*
- *Guide to XUI Toolkit: C Language Binding*

Conventions

mouse	The term *mouse* refers to any pointing device, such as a mouse, a puck, or a stylus.
bold	Boldface text is used to introduce new terms.
Return	A key name enclosed in a box indicates that you press a key on the keyboard.
Ctrl/x	A sequence such as Ctrl/x indicates that you must hold down the key labeled Ctrl while you press another key or a pointing device button.
PF1 x	A sequence such as PF1 x indicates that you must first press and release the key labeled PF1, then press and release another key or a pointing device button.
italics	In syntax and function descriptions, italic type indicates terms that are variable.
. . .	In examples, a horizontal ellipsis indicates one of the following possibilities: ■ Additional optional arguments in a statement have been omitted. ■ The preceding item or items can be repeated one or more times. ■ Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

1

Overview of the UWS Environment

The ULTRIX Worksystem Software (UWS) environment comprises DECwindows applications, X User Interace (XUI) and X programming libraries, and guidelines for creating applications with XUI-compliant interfaces. This environment gives you the following capabilities:

- The ability to distribute applications throughout the network, letting customers take full advantage of network resources.
- A software environment that supports the advanced graphics capabilities of workstations and personal computers.
- A software development environment that makes it easy to create applications with a consistent look and feel—regardless of the hardware and operating system on which the applications run.
- The ability to use workstations and personal computers to display multiple applications concurrently—accomplished most easily in a windowed environment. Each application has a window on the screen that displays output and accepts input.

This chapter summarizes the features of the worksystem environment and describes the XUI architecture.

XUI and DECwindows Features

XUI is an advanced windowing system that extends and improves the X Window System, Version 11. DECwindows is Digital's implementation of XUI-based software components and applications.

The X Window System was developed at the Massachusetts Institute of Technology (MIT) with cooperation from Digital Equipment Corporation. The X Window System provides a high-performing, high-level, and device-independent graphics programming environment. Since its introduction by MIT, the X Window System has become an industry standard.

XUI incorporates and extends the X Window System to provide a windowing environment with the following features:

- Graphics-oriented interaction with the operating system

  DECwindows handles all your interactions with the system. These interactions include initializing a session, managing your environment and resources, starting applications, and managing windows. You can create windows, move windows, resize windows, and shrink windows to icons.

- Consistent user interfaces

  DECwindows user interfaces consist of graphic objects that look and function the same, regardless of the application you are using. This simplifies learning about and using new applications.

- A library of desktop applications

  DECwindows provides a variety of applications, including the following: Clock, Calculator, Calendar, Cardfiler, CDA Viewer, DECterm terminal emulator, Paint Graphics Editor, Notepad Text Editor, and Mail.

- Powerful libraries of programming routines that simplify the development of graphics-oriented applications

  Some of the functions these routines perform include:

  - Graphics operations—drawing points, lines, rectangles, arcs, and text
  - Display management—open, close, and obtain information about the display
  - Window manipulation—create, destroy, configure, and change window attributes

Applications using these routines run on all supported hardware without modifications. For example, an application designed for the VAXstation II looks and functions the same on a VAXstation 2000.

- Libraries for creating and accessing documents containing text, graphics, and scanned images.
- Libraries for communicating with Digital's implementation of the Display PostScript ® system.
- Network-transparent application interface

  The XUI architecture lets you run an application on a remote node, while displaying and inputting data on a local workstation. The network functions that make this possible are transparent; the application appears to run locally.

  This network transparency allows workstations to access the power and resources of other systems on the network.
- Compatibility with VMS

  Because ULTRIX and VMS DECwindows use the same set of programming libraries, you can easily port applications to and from VMS DECwindows systems; no modifications to applications are required.
- Extensible architecture

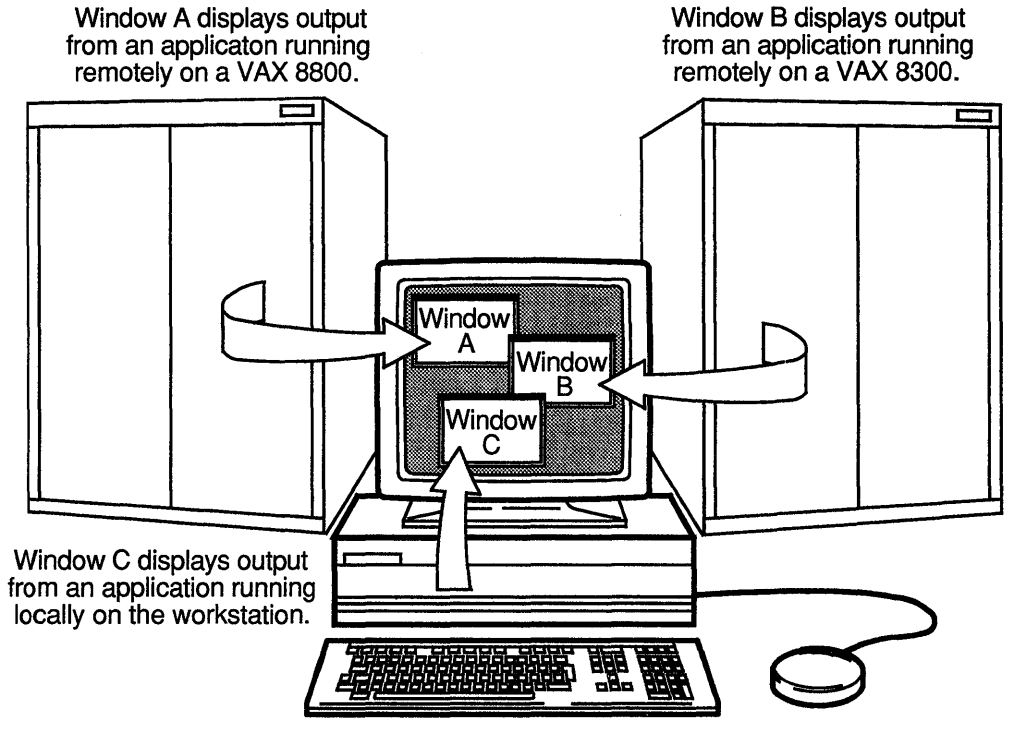
  The XUI architecture has been designed to accommodate new hardware as it becomes available and to add new features.
- Security

  XUI supports the full range of security available to systems running the ULTRIX operating system. In addition to UTLRIX security features, XUI provides the ability to specify which systems can access a server.

Figure 1–1 illustrates several features of XUI. Three applications display a window on a VAXstation. Two of the applications run on remote processors, and yet all the applications appear to run locally.

® PostScript and Display PostScript are registered trademarks of Adobe Systems Incorporated.

Figure 1–1 DECwindows Distributed Applications



ZS–0300–R

XUI Architecture

The XUI architecture is an extension of the X Window System architecture. Components of the architecture include the following:

- DECwindows applications
- XUI Toolkit
- Xlib library
- Network transport
- Server
- Device drivers

In addition to these basic components, XUI includes the Compound Document Architecture (CDA) Toolkit. The CDA Toolkit allows applications to create and access documents containing text, graphics, and scanned images. Figure 1–2 illustrates the relationships among the components of the XUI architecture.

DECwindows Applications

For more information about the user environment and DECwindows applications, see Chapter 2.

DECwindows applications provide a consistent, graphics-oriented user interface that makes it easy to interact with the ULTRIX operating system. In the worksystem environment, applications look and respond similarly.

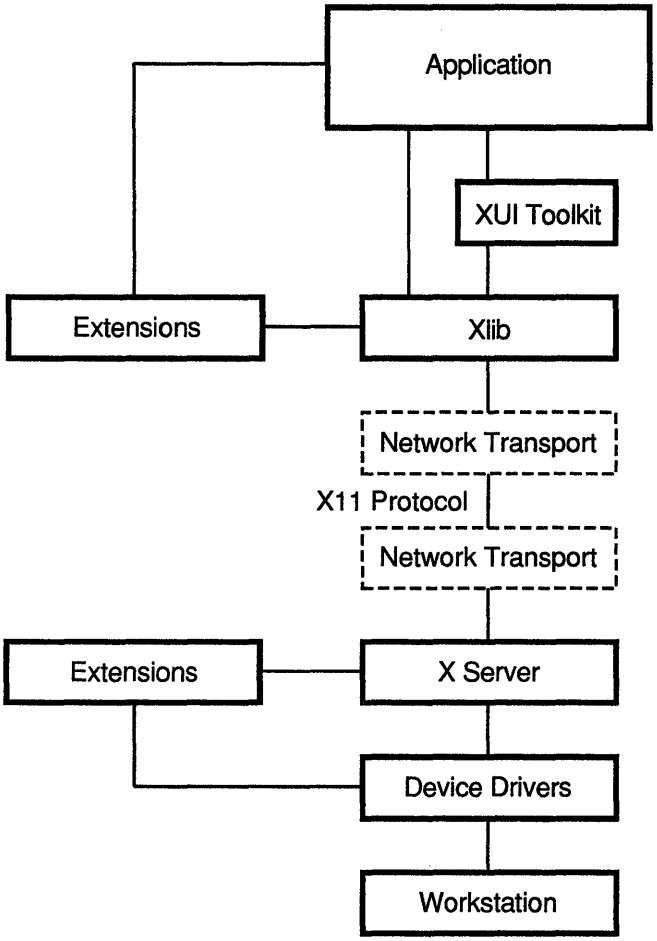
To perform windowing and graphics functions or to get input from the mouse and keyboard, DECwindows applications call routines from the XUI Toolkit and Xlib programming libraries.

XUI Toolkit

For more information about the programming environment see Chapter 3.

The XUI Toolkit includes Digital's implementation of the X Toolkit and consists of high-level library routines that allow applications to create and manage a user interface. Using XUI Toolkit routines, applications can create, modify, and control interface objects such as menus, scroll bars, and buttons. In turn, XUI Toolkit routines call Xlib routines to perform fundamental input and output functions.

Figure 1–2 XUI Architecture



Xlib Programming Library

The Xlib programming library is Digital's implementation of the X Window System's graphics programming library. Xlib provides low-level routines for creating windows, managing windows, performing graphics functions, and communicating with the Display PostScript interpreter.

Xlib routines handle application requests by generating and sending X11 protocol requests through the network to another component of the architecture called the **server**. The X11 protocol defines the common language through which an application and workstation communicate.

Transport Layers

The network transport mechanism uses TCP/IP or DECnet to handle communication, in the form of X11 protocol packets, between Xlib and the server.

The implementation of the transport (TCP/IP or DECnet) and the X11 protocol allows the application and the user interface to be physically separated across the network. For example, you can run an application on a remote node while displaying the window locally on a workstation. The window accepts input and displays output for the application. The application appears to run on the workstation.

XUI also provides an optimized, shared-memory transport mechanism. When an application runs locally on the workstation that displays the interface, the optimized transport reduces the time required to transfer a protocol packet to or from the server.

Server

The X server is Digital's implementation of the X Window System server. The server is the component of the architecture that allows application interfaces to interact with all supported systems in the same way. Residing on the displaying system, the server performs the following:

- Handles interprocess communication between client applications and the X server
- Distributes user input to and accepts output request from client applications
- Maintains an access control list that contains processor host names from which client applications can execute.

In essence, the server converts the data that represents the request to a command that can be executed by the appropriate device drivers.

Device Drivers

A device driver library supports the hardware devices that interact with the window system.

The use of a device library allows programmers to write device-independent client applications. This means programmers do not have to rewrite, recompile, or relink an application when a new hardware display is added. Instead, the X server and its device drivers need only be modified or extended.

Device drivers process requests from the server to the display and pass data from the input devices to the server. The device driver architecture consists of the following components:

- Port input drivers
- Class input drivers
- Common driver
- Output driver

Extending the Architecture

A major feature of the XUI architecture is its extensibility. The programming libraries, X11 protocol, and server can all be extended to support new hardware or to implement new features.

For example, you could extend XUI to support three-dimensional applications. This would require the following actions:

- Develop a new library of routines to create and manage three dimensional images
- Add protocol requests that correspond to the new routines
- Extend the server so that it can interpret and execute the requests

The XUI architecture is also designed to support optional libraries that layer on top of Xlib, for example, the GKS programming library. Such libraries do not require extensions to the protocol or server.

2

The User Environment

By providing consistent, graphics-oriented user interfaces to all software components and applications, the DECwindows user environment makes it easy to interact with the operating system and to use applications. This chapter provides an overview of the user environment and describes the DECwindows desktop applications.

Overview of the User Environment

The DECwindows user environment is designed to increase productivity by saving time. DECwindows provides an environment in which applications look and respond similarly. For example, if two different applications display a menu, the menu looks the same for both applications, and the steps required to pick a menu option are the same.

You interact with a DECwindows system by finding an object on the screen that represents the task you want to perform, then use a pointing device (for example, a mouse) to select that object.

Typical interface objects include:

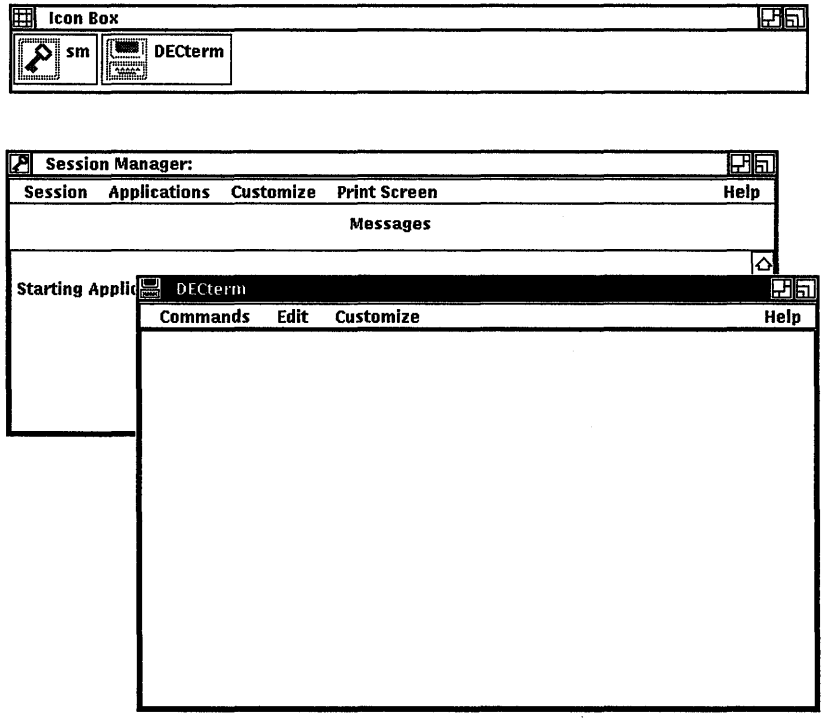
- Menu bars
- Scroll bars
- Pull-down menus
- Pop-up menus
- Buttons

The DECwindows user environment saves time by letting you start a number of applications simultaneously. For example, you can run a program in one window, read mail in a second window, and execute a command in a third.

The DECwindows user environment includes the following components:

- The Session Manager
- The Window Manager
- The User Executive
- DECwindows applications
- Online help

Figure 2–1 illustrates some components of the DECwindows user environment. The following sections describe the components of the user environment.



Starting and Controlling Your DECwindows Session

The Session Manager, your initial interface for DECwindows, acts as a control panel for your DECwindows session. From the Session Manager, you can

- Log on to your workstation and start a DECwindows session
- Create new DECterm terminal emulator and User Executive (UE) windows or start other DECwindows applications
- Customize your DECwindows environment
- Capture window snapshots in a file for printing
- Lock your workstation by pausing your DECwindows session
- End your DECwindows session

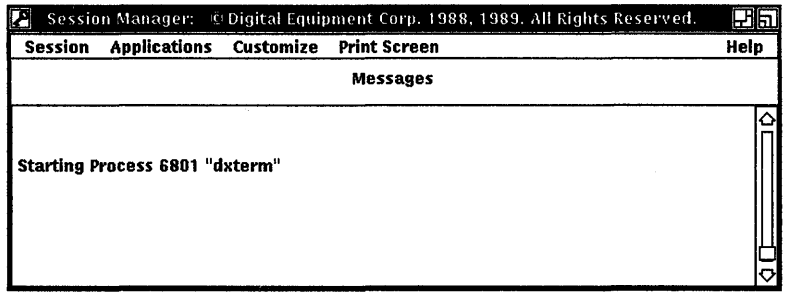
For more information about the Session Manager, see the DECwindows User's Guide.

DECwindows comes with preset options that you can change. You use the Session Manager to change these options, thereby customizing your environment to fit your working style. For example, with Session Manager, you can change options to customize your:

- Keyboard settings
- Windows
- Pointer
- Security settings

You can also set options to have the Session Manager automatically start certain applications, such as the DECterm terminal emulator, each time a new session begins. Figure 2–2 shows the Session Manager window.

Figure 2–2 Session Manager Window



Managing Windows

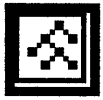
To learn how to manipulate windows and other DECwindows techniques, see the DECwindows User's Guide.

As you start applications and create windows, you might want to change how windows look on your screen. The DECwindows Window Manager lets you:

- Shrink windows to icons
- Change the size of windows
- Move windows around on your screen
- Stack windows

If your screen becomes too crowded, you can shrink windows to icons. When you shrink a window, DECwindows puts an icon representing the window in an area of the screen called the Icon Box. Windows that you shrink remain active but are hidden from view. You can expand the icon to redisplay the entire window by selecting the icon from the icon box.

Accessing the ULTRIX Operating System



The User Executive (UE) is a graphic interface to the ULTRIX operating system. Using UE, you can execute system commands and run DECwindows applications. Because the UE lets you make selections from menus, you do not need to remember the format of system commands.

To learn more about using and customizing UE, see the DECwindows User's Guide.

When you run the UE, the main window displays the names of the files in your current directory. To execute a command, you select the file to which the command applies, then choose the appropriate command. The UE prompts you for parameters, then executes the command.

UE provides the following commands for working with directories and files:

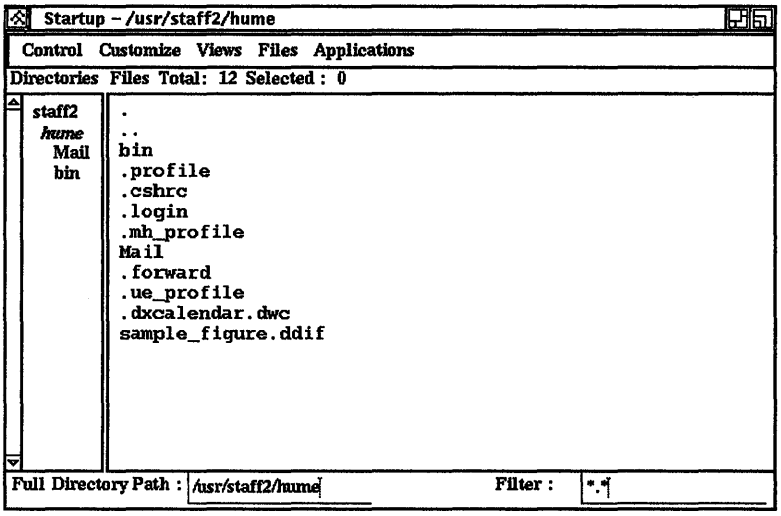
- Create Directory
- Copy
- Display
- Move
- Search
- Remove
- Print

If you are unfamiliar with the ULTRIX operating system, see The Little Gray Book: An ULTRIX Primer.

You can start many DECwindows applications from the UE window. When you choose an item from the UE Applications menu, DECwindows starts the application in a new window. The UE window remains available to start additional applications or to perform other tasks.

You can customize UE menus to suit your work environment. For example, you can add menu options that perform additional file operations or add new menu names and build accompanying pull-down menus. Figure 2–3 shows the User Executive window.

Figure 2–3 UE Window



Getting Online Help

DECwindows provides online help for the user environment. For most applications, you get help from a pull-down menu. DECwindows displays a general help screen from which you can request more specific help. Some applications also provide context-sensitive help. For these applications, DECwindows displays help that is relevant to the window you are using.

Basic information about using DECwindows, such as how to manage windows and use dialog boxes and scroll bars, is available from the Session Manager online help. In addition, many DECwindows terms are explained in the Session Manager help.

Help is designed to let you request general information on an application and quickly narrow the focus of your inquiry. With DECwindows Help you can:

- Navigate quickly through help topics
- Record a history of your help session
- Search help for a keyword or topic supplied by the application
- Save the help text in a file for printing.

The Help menu contains the Overview and About menu items. To see an overview of your application, choose the Overview menu item. To get product information about your application, such as the software version number, choose the About menu item. Some applications have additional Help menu items, for example, the Glossary menu item, which lets you look up terms used in your application.

Choosing any menu item from the Help menu displays a Help window.

DECwindows Desktop Applications

For more information about using individual applications, see the DECwindows Desktop Applications Guide.

DECwindows includes a variety of applications that are designed to increase productivity by automating basic tasks. For example, Mail automates sending and receiving interoffice mail; the Cardfiler automates creating and filing index cards.

Because DECwindows is based on the industry-standard X Window System, you will be able to choose from an extensive offering of applications from Digital and other vendors.

DECwindows provides the following applications:

- Calculator
- Calendar
- Cardfiler
- Clock
- CDA Viewer
- DECterm terminal emulator
- Mail
- Notepad
- Paint
- PostScript Previewer
- Puzzle

For information about basic DECwindows terms and techniques, refer to the DECwindows User's Guide.

You use the same basic window techniques with every DECwindows Application. This simplifies learning about and using new applications. Some of these basic techniques include:

- Using a mouse to point to and select screen objects
- Choosing items from menus
- Using scroll bars
- Supplying information in dialog boxes
- Copying or moving text within a window or between applications

The following sections introduce the DECwindows applications.

Calculator

You use the Calculator to perform basic arithmetic functions: addition, subtraction, multiplication, division, percentages, and square roots. You can also use the Calculator to perform advanced operations, such as trigonometric and inverse trigonometric functions.

You enter data in the Calculator by using the mouse to click buttons on the screen, or by using the keyboard. The Calculator has two displays. One display shows the current operation; the other display shows the contents of memory.

Calendar

The Calendar combines the functions of a desktop calendar and an appointment book to help you keep track of your meetings and plan your time. The Calendar lets you look at a year or month display, or to review your scheduled appointments a day at a time.

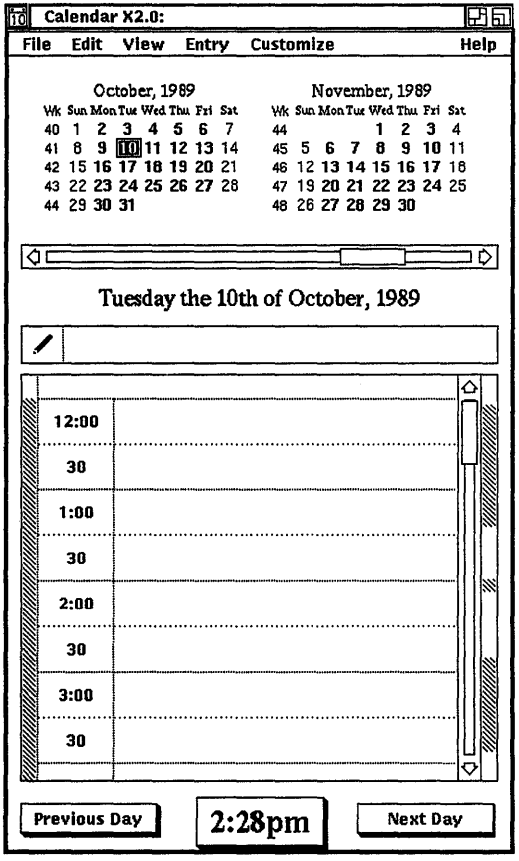
You can use several calendars for different purposes. For example, a group might share a calendar for making conference room reservations.

Calendar also lets you:

- Create calendar entries that look like appointment pages from a desktop calendar or diary
- Mark special days, such as birthdays or holidays
- Add icons to calendar entries that you can use to associate with the entry
- Send and receive calendar entries through electronic mail

Figure 2–4 shows the Calendar Day view window.

Figure 2–4 Calendar Window

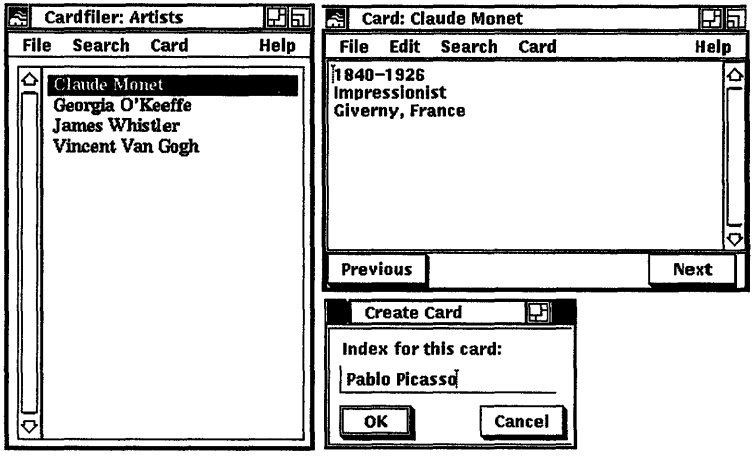


Cardfiler

The Cardfiler provides a system for organizing information; this system is similar to using index cards. Using the Cardfiler, you create cards that contain information, such as names, addresses, and phone numbers. You can also create files that group cards with similar information. For example, you might create a file for business cards and a file for personal cards.

Figure 2–5 shows the cardfiler windows.

Figure 2–5 Cardfiler Windows



In addition to text, you can add graphics, such as those you create with the Paint program, to Cardfiler cards.

Clock

The Clock displays the date and time in analog, digital, and military format. You can customize the display to produce different formats. Clock provides an alarm to remind you of important events.

CDA Viewer

The CDA Viewer displays a Digital Document Interchange Format (DDIF) file and lets you page through its contents. DDIF is a standard format for the storage and interchange of compound documents. Compound documents are documents that contain multiple elements (for example, text and graphics).

DECterm

The DECterm terminal emulator displays a window that looks and functions like a VT300-series terminal. Applications written for VT52, VT100, VT220, or ReGIS terminals run without modification in the DECterm window. Within a DECterm window, you can set terminal characteristics and customize DECterm features such as the window size and screen fonts, the display background and cursor shape, and special keyboards and character sets.

DECterm also lets you

- Copy information within or between terminal windows
- Clear and resize terminal windows
- Compose special characters

See Figure 2–1 for an illustration of a DECterm window.

Mail

The Mail application provides a way to communicate with other users on the system or network. With this application you can:

- Send messages to any user on the system or network
- Read messages from other users and reply to those users
- Print messages
- File and delete messages
- Forward messages
- Send files
- Extract messages you receive to files

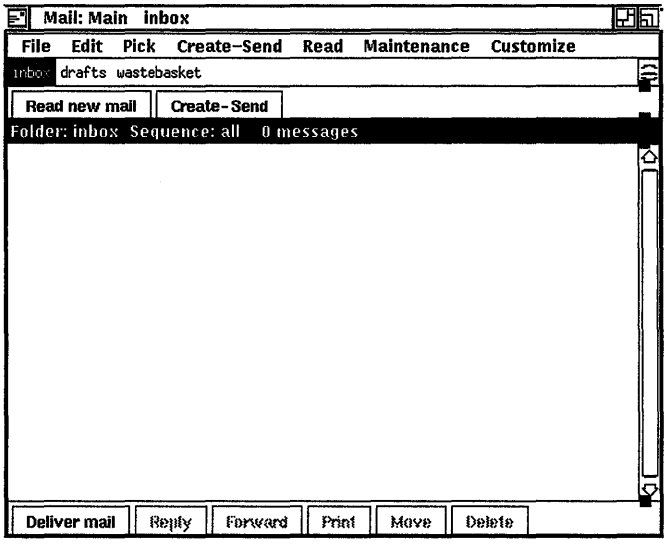
Figure 2–6 shows the main Mail window.

Notepad

The Notepad is an editing program that lets you compose and manipulate text. You can use the Notepad editor for any writing task. You can copy text from other windows and paste it into the Notepad. Notepad also lets you:

- Search for text strings
- Include other files
- Sort text
- Use ULTRIX text filters
- Split your Notepad window into a number of edit windows
- Change the size and style of fonts used to display text
- Recover lost edits

Figure 2–6 Main Mail Window



Paint

Paint is an application that you use to create and save screen images. Using Paint is much like painting or drawing. You select a brush or pencil and trace designs. However, Paint makes the process easier.

With Paint you pick shapes from a menu and let paint draw the shape; you can also select fill patterns to shade (color) your picture. To provide greater detail for part of your picture, you can use the zoom feature to magnify a specified area. If you do not like your picture, you can erase it and start over; if you like it, you can print a copy.

Paint stores images as DDIF files or X11 Bitmap files. Applications that display graphics, such as Cardfiler, can use files that you create using Paint.

PostScript Previewer

The PostScript Previewer lets you display PostScript files on your workstation screen and page through the contents. Because the screen fonts correspond to fonts used by PostScript printers, you can see what a PostScript file looks like before you print it.

Puzzle

The Puzzle game lets you solve a classic sliding-tile puzzle on your workstation display.

Other Worksystem Applications

Your worksystem environment also includes tools for the programmer and a tool to compare differences between two files.

Debugger

For more information about using the Debugger see the Guide to the dxdb Debugger.

The Debugger is a tool that helps you locate run-time or logic programming errors in compiled C language source files while in the DECwindows environment. With the Debugger, you can display your source code in one window while you set a breakpoint or examine variables in other windows.

Visual Differences

For more information about the Visual Differences program see the Guide to the dxdiff Visual Differences Program.

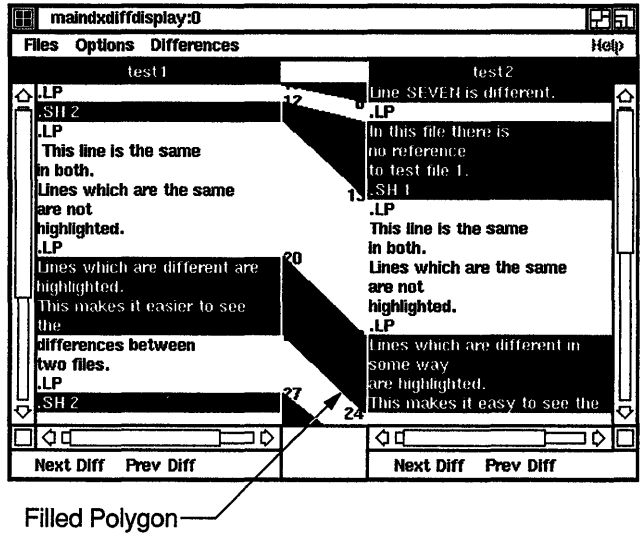
The Visual Differences application lets you compare ASCII text files on a line-by-line basis, then provides a graphical display of the differences between the two files.

The files you want to compare are displayed in two windows on your screen. Visual Differences highlights the differences it finds and shows the location of the differences in the two files by connecting them with lines. You can use scroll bars or control buttons to move forward or backward through the text displayed on your screen.

The Options menu lets you change the way differences are displayed. For example, you can display line numbers or scroll one file independently of the other.

Figure 2–7 shows how Visual Differences displays the differences between two files.

Figure 2–7 The Visual Differences Display



ZS–0304–R

3

The Programming Environment

The XUI offers a rich environment for developing applications for graphic workstations. This chapter provides an overview of the programming environment and describes the libraries and tools available to the DECwindows application programmer.

Overview of the Programming Environment

The XUI programming environment allows you to develop applications that have simple, consistent, graphics-oriented interfaces. This consistency is important: it allows the user to transfer knowledge gained from using one application to another, reducing the amount of learning and subsequent recall.

The core of the graphics programming environment consists of the Xlib and XUI Toolkit programming libraries. Xlib supplies low-level routines for performing basic graphic and windowing functions. The XUI Toolkit supplies high-level routines for creating and managing user interface objects like menus, scroll bars, and buttons. Applications can call routines from both libraries.

The XUI style is fully defined and illustrated in the XUI Style Guide.

XUI Toolkit routines simplify the task of creating a user interface. For example, you can create a menu with one call to an XUI Toolkit routine, whereas creating the same menu using Xlib routines would require many more calls and program lines. Using XUI Toolkit routines also ensures that an application interface conforms to the XUI style, which makes the application easy to learn and use.

In addition to Toolkit routines, the XUI Toolkit includes the XUI User Interface Language (UIL) compiler and XUI Resource Manager (DRM) routines, which, in essence, let you create an entire interface with one library call. The UIL and DRM let you separate form and function in an application, giving you the ability to specify and modify the interface without recompiling the entire application.

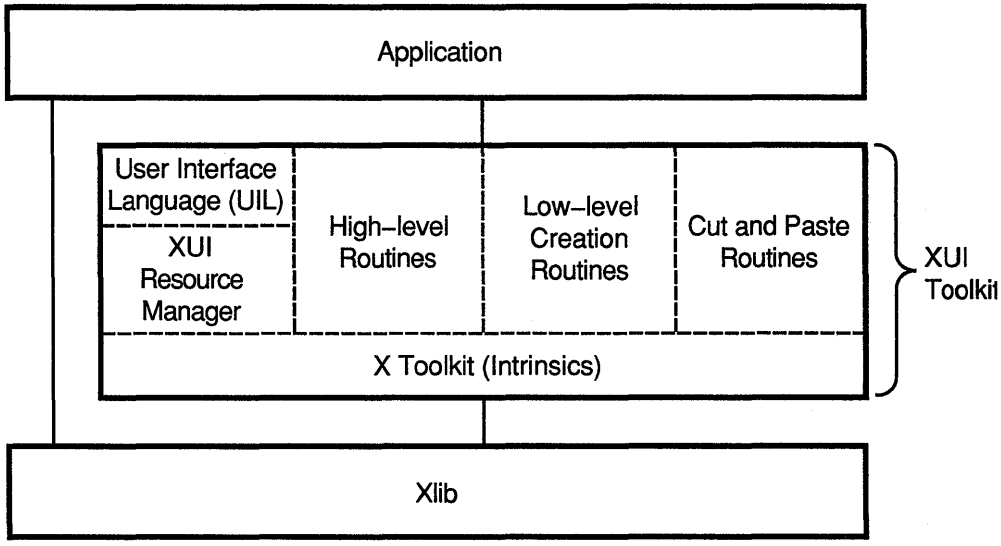
Creating and Managing Windows

Using XUI programming library routines, you can write applications that create and manage windows. A window is an interface object that displays output or accepts input for an application.

From the user's perspective, an application usually consists of a series of windows. You use library routines to construct and manage this hierarchy of windows.

Figure 3–1 illustrates the components of the XUI programming environment. The following sections describe the components.

Figure 3–1 XUI Programming Environment



ZS–0302–R

The Xlib Library

For more information about Xlib, see the Guide to the Xlib Library: C Language Binding.

The Xlib programming library is Digital's implementation of the X Window System's C-language graphics programming library. Xlib is the lowest-level programming interface to the DECwindows environment. Applications call the following Xlib routines to perform basic windowing and graphics functions:

- Display routines
- Window routines
- Management routines
- Event and error handling routines
- Graphics routines
- Text, font, and cursor routines

For more information about the Display PostScript system, see the Guide to Developing Applications for the Display PostScript System.

DIGITAL has extended Xlib to integrate Xlib Library with the ULTRIX operating system and to add DECnet support. In addition, Digital has extended the Xlib library to provide routines for the Display PostScript system. This extension lets a C-Language application display images in an X window by calling functions that send PostScript code.

XUI Toolkit

From a programming perspective, the XUI Toolkit is the key to the DECwindows environment. XUI Toolkit routines provide a straightforward way to create a user interface that conforms to the XUI style. The XUI Toolkit also provides development tools that allow you to separate form and function in an application; you can modify an interface without having to recompile and relink the application.

XUI Toolkit routines allow you to create **widgets**. From a user's perspective, a widget is an interface object that displays output or accepts input. From a programmer's perspective, a widget is an object that is logically connected to application functions.

When a user interacts with a widget (for example, by making a menu selection), information in the widget makes the application respond appropriately. Widgets can be simple interface objects such as labels and buttons, or complex objects such as menus and scroll bars.

For more information about the XUI Toolkit, see the Guide to Writing Applications Using XUI Toolkit Widgets.

You can use the following components of the XUI Toolkit to create and manage widgets:

- High-level widget creation routines
- Low-level widget creation routines
- XUI Intrinsics Routines
- Cut and paste routines
- Convenience routines
- Compound String routines
- XUI User Interface Language (UIL) Compiler
- XUI Resource Manager (DRM)

The following sections describe the components of the XUI Toolkit.

High-Level Widget Creation Routines

High-level routines let you create a widget hierarchy and specify initial attributes of a widget. High-level routines provide some control over widget attributes while ensuring that your application conforms to the DECwindows style.

Low-Level Widget Creation Routines

Low-level routines also provide ways to create a widget hierarchy and specify initial attributes of a widget. Compared to the high-level routines, low-level routines provide more control over widget attributes, and therefore give you more freedom to customize the widgets. However, low-level routines can be more difficult to use. Often, you must specify a lengthy argument list in calls to a routine.

XUI Intrinsics Routines

XUI Intrinsics routines let you manage and manipulate widgets at run time. Using intrinsics routines, you can do the following:

- Control the location of widgets
- Map and unmap widgets to the screen
- Process input from the user of the application

Cut and Paste Routines

Cut and paste routines allow application users to transfer data between windows.

Convenience Routines

Convenience routines provide functions that let your application format and display messages, such as from the operating system, in a message window. Convenience routines also provide functions that let you write upward-compatible applications and widgets.

Compound String Routines

Compound string routines let applications create and manipulate compound strings and font lists. Using compound string routines, you can

- Create a font list and font list entries
- Create compound strings
- Compare and manipulate compound strings

XUI User Interface Language Compiler

For more information about UIL, see the Guide to the XUI User Interface Language Compiler.

The XUI User Interface Language (UIL) is a specification language that simplifies the creation and customization of an application's user interface. Using UIL, you create a specification file that describes the objects (menus, dialog boxes, labels, push buttons, and so on) that make up the user interface for your application. This UIL file also specifies the routines to be called when the interface changes state (as a result of user interaction.)

Using UIL, you can specify the following:

- The widgets that make up the interface
- The characteristics of the specified widgets
- The callback routines for each widget
- The hierarchy of widgets in the application

The UIL compiler provides the following benefits to application developers:

- You can create a variety of interfaces for a single application. For example, if you develop applications for international markets, you can customize the interface for each market.

- The UIL compiler detects a variety of errors that high-level and low-level XUI Toolkit routines do not detect. For example, the UIL compiler issues diagnostic messages if you specify the wrong type of value for an argument or if you specify an unsupported argument.
- The UIL compiler aids in the prototyping of user interfaces for applications. You can develop and test a variety of interfaces, even if the application's functional routines are unavailable.

XUI Resource Manager

For more information about DRM, see the Guide to the XUI User Interface Language Compiler.

The XUI Resource Manager (DRM) is the facility that takes data from the compiled UIL specification file (called the UID database) and creates the widgets. DRM translates the information in the UID database into low-level widget creation calls. If you use UIL to create a user interface, your application must include a special set of routines that invoke DRM. Using DRM routines reduces the time required for applications to create widgets.

4

Customizing Your DECwindows Environment

You can personalize your DECwindows applications by modifying entries in the X Resource Manager database file called .Xdefaults.

This chapter introduces the basics you need to customize your DECwindows environment. For details on individual applications, see the *ULTRIX Worksystem Software Reference Pages*.

Why Customize with .Xdefaults?

You can use DECwindows as it comes packaged for you or you can create a personalized environment. The Session Manager Customize menu lets you change several options, including keyboard, window, pointer, and security settings. Using the Session Manager Customize menu, you can examine the preset settings, then choose and save new settings.

However, some preset conditions cannot be changed with menu items. Such conditions include whether or not the application starts as a window or icon, and the size and location of an application's startup window.

Instead, you can modify these preset conditions by registering your user preferences with the X Resource Database Manager (xrdb). You can specify these preferences either at the command line or by way of a database file called .Xdefaults.

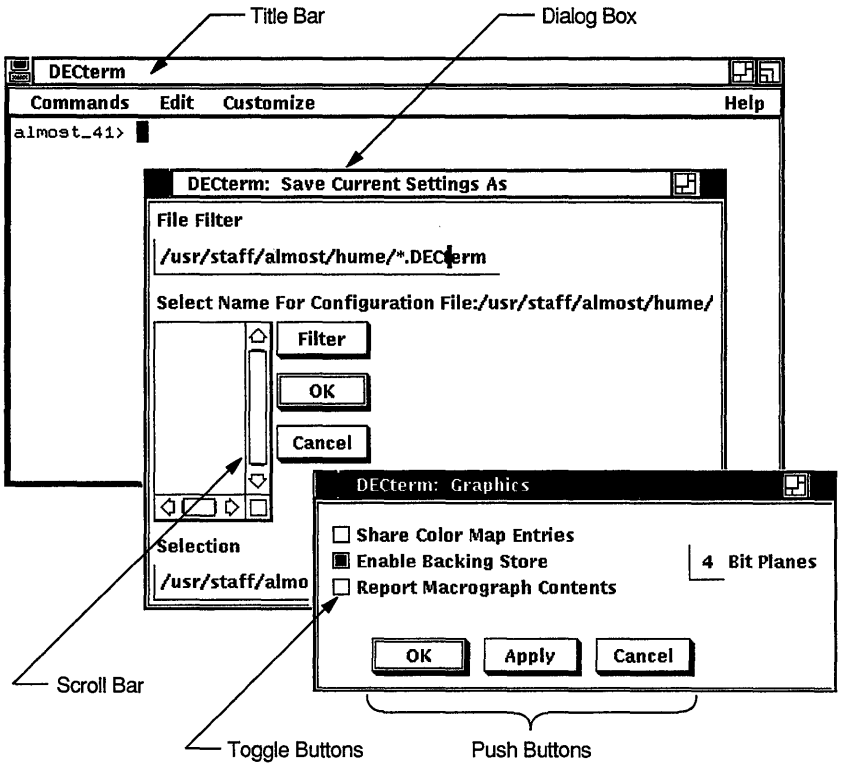
For example, by adding the appropriate entries, you can change:

- An application's startup state or screen position
- The height and width, in pixels, of a window
- The background and foreground color of window components such as push buttons, scroll bars, or title bars
- The fonts that appear in labels or titles
- The way keyboard keys and mouse buttons map to text editing functions

Getting Started

The XUI Toolkit provides a standard set of window objects (widgets) that DECwindows applications can use. Some of these widgets include menus, push buttons, toggle buttons, dialog boxes, and scroll bars. Figure 4–1 illustrates some of these window components.

Figure 4–1 Window Components



The X Resource Database Manager (xrdb) is a database manager that assigns attributes to widgets and groups of widget components. You manipulate this database by editing the .Xdefaults file.

The .Xdefaults file contains entries that provide information about the appearance or characteristics of a widget. These widget characteristics are referred to as **resources**. You provide entries in the file .Xdefaults in the form of resource specification strings.

Most DECwindows applications read the .Xdefaults file during startup and use the defined entries to customize the appearance and characteristics of their windows.

Creating the .Xdefaults File

The Resource Manager and DECwindows applications read the .Xdefaults from your home directory. You can create an .Xdefaults file in your home directory or copy a sample .Xdefaults file from the /usr/skel directory. You can modify, remove, or add to the entries in the sample file to customize your windows.

Example 4–1 provides a sample .Xdefaults file. Note that comment lines begin with an exclamation point (!).

Example 4–1 Sample .Xdefaults File

```
#ifdef COLOR
*highlight:      #ffff00000000
*background:     #bfffffffc00
#endif
!
!  *** Session Manager ***
!
❶ sm.create_terminal:              1
#ifdef COLOR
sm.pointer_foreground:  #000
sm.pointer_background:  #fff
#endif
!
!  *** Customize Icon Box ***
!
❷ wm.default.sticky:            true
!
❸ wm*iconStyle:                    1
!
❹ wm*Geometry:          =150x200-0-0
!
❺ wm*forceBars:                    false
!
!  *** Customize DECterm ***
!
❻ wm*DXterm*iconStyle:             0
!
❼ wm*DXterm*iconName:              My_Node
!
DXterm*saveLinesOffTop:                    on
DXterm*DECterm.scrollVertical:             on
DXterm.main.terminal.autoWrapEnable:       on
!
#ifdef COLOR
DXterm*background:       wheat
DXterm*foreground:       black
#endif
!
!  *** Customize Mail ***
!
❽ dxmail*initialState:             3
!
```

The following notes describe the effect that some of the resource specifications in Example 4–1 have on an application or window.

❶ Tells the Session Manager to create one terminal emulator window at startup

❷ Locks your icon box window as the bottom window on the screen

❸ Sets the icon size to small for all applications

❹ Sets the size and location of the icon box

❺ Specifies that scroll bars will not appear in the icon box

❻ Sets the DECterm icon size to large

❼ Redefines the DECterm icon name to My_Node.

❽ Causes Mail to start as an icon.

Specifying Resources

The .Xdefaults file is a list of resource specification strings you enter or modify to customize your DECwindows application. Each resource specification in an .Xdefaults file must have the following format:

[*name**] *resource*: *value*

name — Specifies the name or name string representing the application or widget to which the resource assignment should apply. If this argument is not specified, the resource assignment is globally available to all applications.

You must separate names in this string with either an asterisk (*) or a period (.), as follows:

* Specifies that the resource definition applies to all widgets even if classes or names are omitted in the specification string.

. Indicates that the resource definition applies only to the widget named.

resource — Specifies the resource (characteristic) you want to modify.

value — Specifies the value that is to be assigned to the resource.

Each component of a resource specification has an associated widget class. Several widgets or widget attributes have the same class. For example, the foreground color used to draw the pointer and the text cursor are all delivered as instances of the Foreground.

You can use the name string to restrict the resource specification to a type of widget, for example, a scroll bar or text window. Each widget is associated with a widget class. You can specify a type of widget in one of two ways:

- **Class identifier**—Widgets classified according to common properties and functions (for example, main window, scroll bar, menu bar, or push button)
- **Name identifier**—Name of a particular item in a widget class (for example, the Dismiss push button)

Widget classes and names are case sensitive. Most names begin with initial lowercase letters; compound names use initial capital letters to separate names, for example:

```
backgroundPixel
```

In the resource specification, be sure to specify names in the following order:

1. Application name
2. Widget class
3. Additional widget names
4. Resource name

For more information about widgets and the widget hierarchy, see the section Identifying Widgets.

When specifying widget names, it is important to understand that widgets are arranged in a hierarchical fashion. The syntax you use when customizing must match the class and object hierarchy built into the toolkit widgets.

Depending on how you specify the widget name in your resource assignment string, the change you assign to one widget can affect its related widgets. Similarly, if you do not name all of the related widget components, the change may not take effect.

'or detailed ıformation bout naming ıidgets and esources, see he section More ıbout Widgets nd Resources.

You can find a complete list of resource names, classes, and types in the file /usr/include/X11/StringDefs.h. You can find a complete list of color names in the file /usr/lib/X11/rgb.txt.

When you finish adding resource specifications to your .Xdefaults file, you must load changes into the server to have them take effect. To load changes, take one of the following actions:

- Quit the Session Manager then start a new session

  or

- Run the Resource Manager to reload the .Xdefaults file into the server. To run the Resource Manager, type the following command:

  ```
  % /usr/bin/X11/xrdb .Xdefaults
  ```

Note: Some changes will not take effect unless you quit your current session and then start new one.

Resource Specification Examples The following examples show specifications you can add to the file .Xdefaults to customize your DECwindows applications. These examples illustrate the use of application names, widget names, and delimiters to control the effect of the resource specification.

Example 1—Specification string that affects all applications:

```
*background:  green
```

This specification changes the background color of all DECwindows applications. Note that without the asterisk, the resource specification would be ignored.

Example 2—Specification string that affects one application:

```
dxmail*background: green
```

The addition of the application name dxmail causes this specification to change the background color of each of the Mail application's windows. Other applications are not changed.

Example 3—Specification string that affects one component in one application:

```
dxmail.background: green
```

The use of the period (.) delimiter instead of the asterisk (*) causes this specification to change the background color of Mail's main window only. Mail's other windows are not changed.

Example 4—Specification string that affects one occurrence of a component in an application.

```
dxmail*folderArea*background: green
```

The addition of the widget name identifier folderArea causes this specification to change the background color of a particular area only—all of the folder buttons in Mail's main window.

More About Widgets and Resources

The sample .Xdefaults file in Example 4–1 provides enough information to get you started in personalizing your worksystem applications. This section provides detailed information about widgets and resources. You should review this section before trying more sophisticated customizing.

Locating Resources

Resources are assigned when the application program creates a widget or by the application's defaults file. You can assign personal preferences at the command line, through the .Xdefaults file in your home directory, or any other file. Use the Resource Manager to load the values into the server. You can assign resources in any of the following categories:

- General
- Widget-specific
- Application-specific

Precedence for Setting Resources

Resources are set for widgets in the following precedence:

- Instances take precedence over specific classes. For example,

  ```
  *scrollbar.background
  ```

  has precedence over

  ```
  *scrollbar.Background
  ```

- Any instance or class explicitly stated takes precedence over one that is omitted. For example,

  ```
  dxmail*Background
  ```

  takes precedence over

  ```
  Background
  ```

- Name strings bound by a period (.) take precedence over those bound by an asterisk (*).

For example,

```
dxmail.geometry
```

has precedence over

```
dxmail*geometry
```

- An application can have its own set of database files that provide defaults. These defaults override resource values assigned when the widget is created.
- User-supplied resource values supplied on the application command line or in the personal resource database file .Xdefaults override any widget or application defaults.

You can find a complete list of resource names, classes, and types in the file /usr/include/X11/StringDefs.h.

General Resources Table 4–1 describes the general widget resources.

Table 4–1 General Resources

Resource	Description
x	Specifies the placement (in pixels) of the left side of the widget window relative to the inner upper corner of the present window. The default is centered in the parent window.
y	Specifies the placement (in pixels) of the upper left corner of the widget window relative to the inner upper left corner of the parent window. The default is centered in the parent window.
width	Specifies the width (in pixels) of the widget window.
height	Specifies the height (in pixels) of the widget window.
border	Specifies the border pattern.
borderWidth	Specifies the border width of the widget window.
borderPixMap	Specifies the color of the widget window's border.
foreground	Specifies the color of foreground objects in the widget window.
background	Specifies the color of background objects in the widget window.
font	Specifies the text font used in the widget.

Widget-Specific Resources Widgets also have resources not specified by general resources for all classes of widgets. Table 4–2 describes the widget-specific resources.

Table 4–2 Widget-Specific Resources

Resource	Description
label	Specifies the string that defines the text of the label.
wordWrap	Specifies a Boolean value of either true (1) or false (0). If true, lines are broken at word breaks (that is, text does not go off the right edge of the widget window).
scrollVertical	Specifies a Boolean value of either true (1) or false (0). If true, a scroll bar is added to allow the user to scroll vertically through the text.
translations	Specifies the translation table used by the input module to determine the mapping of keystrokes and mouse actions.
shape	Specifies a Boolean value of either true (1) or false (0) that indicates the toggle button shape (either rectangular or oval). If true, the toggle button shape is rectangular.

Application-Specific Resources Some applications define unique resources for widgets. For example:

- The PrintCommand resource in dxmail (Mail).
- The windowManagerName, terminalEmulatorName, createTerminal, and startupState resources in dxsession (Session Manager).
- The default.sticky resource in dxterm (DECterm).

You find application-specific resources in the application's individual default file. The default files for individual applications are located in /usr/lib/X11/app-defaults/*application-name*. Customize these widgets by referencing them by name in your .Xdefaults file.

Example 4–2 and Example 4–3 show sample application default files similar to those you can find in /usr/lib/X11/app-defaults.

Example 4-2 Default File for the DECwindows Clock Application

```
! The basics
!
Clock.x:                          0
Clock.y:                          0
Clock.width:                    150
Clock.height:                   105
!
! Make it pretty
!
Clock*fontFamily:     *-*-Times-Bold-R-Normal--*-*-*-*-P-*-*-*
!
! Set the components
!
Clock.Clock.AnalogOn:             1
Clock.Clock.DigitalOn:            1
Clock.Clock.DateOn:               1
!
! Set the alarm
!
Clock.Clock.AlarmOn:              0
Clock.Clock.AlarmPM:              0
Clock.Clock.AlarmHour:           12
Clock.Clock.AlarmMinute:         00
!Clock.Clock.AlarmMessage:       Alarm Message
!
! You can change colors by either affecting the whole clock
!
Clock*foreground:               DarkSlateGrey
Clock*background:               Turquoise
!
! or individual parts
!
!Clock.Clock.Analog.background:          DarkSlateGrey
!Clock.Clock.Analog.foreground:          Turquoise
!Clock.Clock.Date*background:            DarkSlateGrey
!Clock.Clock.Date*foreground:            Turquoise
!Clock.Clock.Digital.background:         DarkSlateGrey
!Clock.Clock.Digital.foreground:         Turquoise
```

Example 4-3 Default File for the DECwindows Notepad Application

```
*copyrightMessage:        Notepad - DECwindows Text Editor
notepad.geometry:                 500x650+4+120
*enableBackups:                   on
!
*Text.font:     -DEC-Terminal-Medium-R-Narrow--14-140-75-75-C-6-ISO8859-1
*DwtText.font: -DEC-Terminal-Medium-R-Narrow--14-140-75-75-C-6-ISO8859-1
!
*workArea*scrollVertical:         TRUE
*workArea*wordWrap:               TRUE
*copyright:                       on
*changesUntilCompress:            1
!
*DwtMenu*New.label:               New
*DwtMenu*OpenDialog.label:        Open...
*DwtMenu*OpenSelected.label:      Open Selected
*DwtMenu*IncludeDialog.label:     Include...
*DwtMenu*Save.label:              Save
*DwtMenu*SaveAs.label:            Save As...
*DwtMenu*Revert.label:            Revert
*DwtMenu*Quit.label:              Quit
*DwtMenu*Exit.label:              Exit
*DwtMenu*Undo.label:              Undo
*DwtMenu*Redo.label:              Redo
*DwtMenu*UndoDialog.label:        Undo...
!
*DwtMenu*Cut.label:               Cut
*DwtMenu*Copy.label:              Copy
*DwtMenu*Paste.label:             Paste
!
*DwtMenu*NextSelected.label:                Next Selected
*DwtMenu*PreviousSelected.label:            Previous Selected
*DwtMenu*NextDialog.label:                  Next...
*DwtMenu*PreviousDialog.label:              Previous...
*DwtMenu*NextIncrementalDialog.label:       Next Incremental...
*DwtMenu*PreviousIncrementalDialog.label:   Previous Incremental...
*DwtMenu*ReplaceOnce.label:                 Replace Once...
*DwtMenu*ReplaceSelected.label:             Replace Within Selected Area...
*DwtMenu*ReplaceAll.label:                  Replace All...
*DwtMenu*GoToTop.label:                     Go to Top
*DwtMenu*GoToBottom.label:                  Go to Bottom
*DwtMenu*GoToSelectedLine.label:            Go to Selected Line Number
*DwtMenu*GoToLine.label:                    Go to Line...
*DwtMenu*positionMenu.label:                Navigate
*DwtMenu*fileMenu.label:                    File
*DwtMenu*editMenu.label:                    Edit
*DwtMenu*findMenu.label:                    Search
*DwtMenu*replaceMenu.label:                 Replace
*DwtMenu*optionsMenu.label:                 Customize
*DwtMenu*searchOptionsDialog.label:         Set Search Options...
!
*DwtMenu*helpMenu.label:                    Help
!
*workArea.borderWidth: 1
```

Identifying Widgets

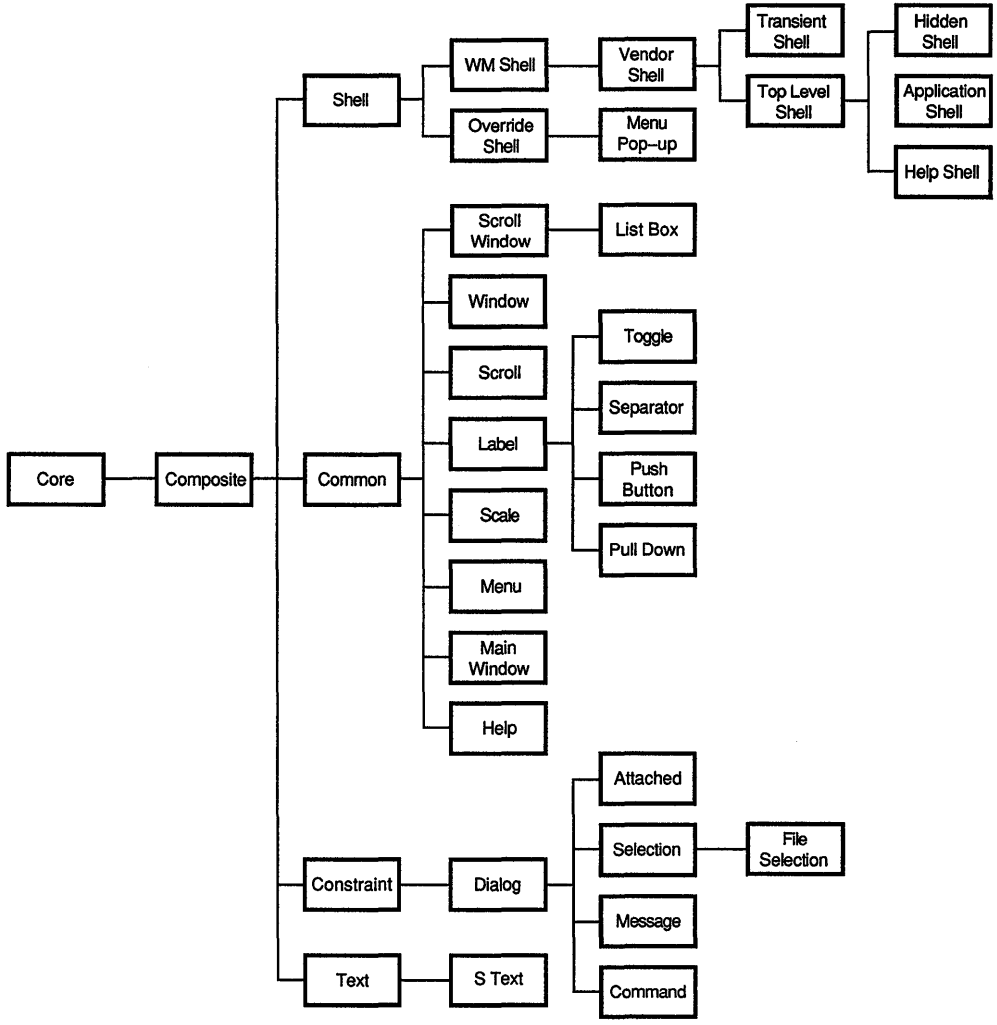
A widget is a window component that displays output or accepts input. Widgets can be simple interface objects like labels and buttons, or complex objects like menus and scroll bars. Widgets are logically connected to application functions. When you interact with a widget (for example, by making a menu selection), information in the widget makes the application respond appropriately.

Widget Hierarchy Widgets are arranged in a hierarchical fashion. This hierarchy establishes a relationship between widget components. For example, the top-level widget (parent) of an application may have several related subwidgets (children).

Depending on how you specify your resource assignment string, the change you assign to one widget can affect its related subwidgets, or it may not take effect if you do not name all of the related widget components. For example, if an asterisk (*) is used in the specification string, the resource is applied to the entire subwidget tree.

Figure 4–2 illustrates the widget hierarchy.

Figure 4–2 Widget Hierarchy



ZS–0303–R

Naming Widgets in Resource Specification Strings Table 4–3 describes some of the widgets you can name in .Xdefaults resource specification strings. Note that when using a name from Table 4–3 in a resource specification string, you must prefix the name with the string "Dwt". For example, DwtAttachedDialogBox.

Table 4–3 XUI Toolkit Widgets

Widget	Description
Boxes	
AttachedDialogBox	A type of dialog box that when resized, changes the position of the subwidgets it contains. Widgets can be attached to any edge of the attached dialog box. This allows the subwidgets to remain visible when the attached dialog box is resized.
CautionBox	A type of dialog box that displays a warning message before an irreversible action is executed.
DialogBox	A composite widget that contains subwidgets. Each subwidget displays information or requests or handles input from the user.
FileSelection	A type of dialog box that queries the user for a file specification.
Help	A widget that presents helpful information on the application you are using.
ListBox	A box that presents the user with a list of choices. A list box includes a scroll bar, by default, to aid in viewing list items.
MessageBox	A type of dialog box that displays a message.
PopupAttachedDialogBox	A type of attached dialog box that does not get clipped by its parent.
PopupDialogBox	A type of the dialog box that does not get clipped by its parent.
RadioBox	A box that presents the user with a list of choices, only one of which can be selected at any one time.
SelectionBox	A box that presents the user with a list of choices, only one of which can be selected at a time.

(continued on next page)

Table 4–3 (Cont.) XUI Toolkit Widgets

Widget	Description
Boxes	
WorkInProgressBox	A type of dialog box that displays a "Work in Progress" message.
Buttons	
PulldownMenu	Presents a list of choices that are not visible until selected from a menu.
PushButton	Displays a rectangular border around a label. The label defines the immediate action of the button, for example, OK or Dismiss in a dialog box.
ToggleButton	Provides control of an "on" or "off" state.
Menus	
Menu	Presents a list of choices to the user.
MenuBar	Presents a list of choices arranged horizontally, each of which contains a set of pull-down menus.
OptionMenu	Presents a "one of many" choice while displaying the current status of choices.
Windows	
CommandWindow	Creates a command entry area.
MainWindow	Top-level window that contains subwidgets, for example, a title bar. All other application windows are subordinate to the main window.
Window	A simple window for an application.

(continued on next page)

Table 4–3 (Cont.) XUI Toolkit Widgets

Widget	Description
Miscellaneous	
Label	Displays read-only, constant text in a widget.
Scale	Allows the choice of a value from within a range of values.
ScrollBar	A screen object that the application or user uses to scroll through display data too large for the display area.
Separator	Draws a bar or line between list or menu entries.
TextEdit	Allows lines of text to be edited.

Changing Key Definitions

You can use the .Xdefaults file to customize text widgets by adding resource specifications that change the way keys are mapped to text editing commands. Table 4–4 lists the text editing keys you can redefine and describes the preset text editing function for each key.

Table 4–4 Text Editing Keys and Preset Functions

Key	Editing Function
⇒ (Right arrow)	Moves the cursor one character to the right. In a single-line field, the cursor does not move when positioned at the end of the line. In a multi-line field, the cursor moves into the first position of the next line.
⇐ (Left arrow)	Moves the cursor one character to the left. In a single-line field, the cursor does not move when positioned at the beginning of the line. In a multi-line field, the cursor moves into the last position of the previous line.
⇑ (Up arrow)	In a multi-line field, moves the cursor up one line.
⇓ (Down arrow)	In a multi-line field, moves the cursor down one line.
F12	Positions the cursor at the beginning of the line.
SHIFT/F12	Positions the cursor at the end of the line.

(continued on next page)

Table 4–4 (Cont.) Text Editing Keys and Preset Functions

Key	Editing Function
F13	Deletes characters to the left of the cursor through the beginning of the word, and shifts to the left all text to the right of the deleted characters.
SHIFT/F13	Deletes the characters to the right of the cursor through the beginning of the word, and shifts to the left all text to the right of the deleted character.
Delete	Deletes the character before the cursor and shifts to the left all text to the right of the deleted character.
SHIFT/Delete	In overstrike mode, deletes the character under the cursor and shifts to the left all text to the right of the deleted character.

Assigning New Text Editing Commands A translation specification string in your .Xdefaults file assigns a text editing command to a named key or mouse button combination. To assign a new editing command to a named key, use the following syntax:

*Text.translations: *key: function*

key	Specifies the key or mouse button that is used to invoke the named function.
function	Specifies the function to be invoked when the named key is pressed.

You can also specify the following mouse buttons in a translation specification string:

Btn1	Left button or Right button
Btn2	Middle button
Btn3	Right button or Left button

You can assign the following actions to each mouse button:

Down	Action is invoked when the button is pressed
Up	Action is invoked when the button is released
PtrMoved	Action is invoked when the mouse button is down and the pointer is moved

Sample Translations Resource Specification Example 4–4 provides text translation entries you can add to your .Xdefaults file. These resource specifications assign EMACS-like key bindings to your keyboard keys.

Example 4–4 Sample Text Specifications

```
!
! DECtoolkit text bindings
!
*DwtText.translations:
!
Shift<Key>0xff53:    forward-word()\n\
<Key>0xff53:         forward-character()\n\
Shift<Key>0xff51:    backward-word()\n\
<Key>0xff51:         backward-character()\n\
<Key>0xff54:         next-line()\n\
<Key>0xff52:         previous-line()\n\
Shift<Key>0xffff:    delete-next-character()\n\
<Key>0xffff:         delete-previous-character()\n\
Shift<Key>0xffc9:    end-of-line()\n\
<Key>0xffc9:         beginning-of-line()\n\
Shift<Key>0xffca:    delete-next-word()\n\
<Key>0xffca:         delete-previous-word()\n\
<Key>0xff0d:         newline()\n\
Shift<Key>0xff0a:    delete-next-word()\n\
<Key>0xff0a:         delete-previous-word()\n\
!
!
!
Shift ~Ctrl ~Alt<Btn1Down>:                extend-start()\n\
Shift ~Ctrl ~Alt Button1<PtrMoved>:        extend-adjust()\n\
Shift ~Ctrl ~Alt<Btn1Up>:                  extend-end()\n\
Ctrl ~Shift ~Alt<Btn1Down>: grab-focus()   set-insertion-point()\n\
Ctrl ~Shift ~Alt Button1<PtrMoved>:        \n\
Ctrl ~Shift ~Alt<Btn1Up>:                  \n\
~Shift ~Ctrl ~Alt<Btn1Down>: grab-focus()  set-insertion-point() select-start()\
~Shift ~Ctrl ~Alt Button1<PtrMoved>:       extend-adjust()\n\
~Shift ~Ctrl ~Alt<Btn1Up>:                 extend-end()\n\
~Shift ~Ctrl ~Alt<Btn3Down>:               stuff()\n\
!
!
!
Ctrl<Key>f:          forward-character()\n\
Ctrl<Key>b:          backward-character()\n\
Alt<Key>f:           forward-word()\n\
Alt<Key>b:           backward-word()\n\
Alt<Key>]:           forward-paragraph()\n\
Ctrl<Key>[:          backward-paragraph()\n\
Ctrl<Key>a:          beginning-of-line()\n\
```

(continued on next page)

Example 4–4 (Cont.) Sample Text Specifications

```
Ctrl<Key>e:              end-of-line()\n\
Ctrl<Key>n:              next-line()\n\
Ctrl<Key>p:              previous-line()\n\
Ctrl<Key>v:              next-page()\n\
Alt<Key>v:               previous-page()\n\
Alt<Key>\\<:             beginning-of-file()\n\
!
!
Alt<Key>\\>:             end-of-file()\n\
Ctrl<Key>z:              scroll-one-line-up()\n\
Alt<Key>z:               scroll-one-line-down()\n\
Ctrl<Key>d:              delete-next-character()\n\
Ctrl<Key>h:              delete-previous-character()\n\
~Shift Alt<Key>d:        delete-next-word()\n\
~Shift Alt<Key>h:        delete-previous-word()\n\
Ctrl<Key>w:              kill-selection()\n\
Ctrl<Key>y:              unkill()\n\
Alt<Key>y:               stuff()\n\
Ctrl<Key>j:              newline-and-indent()\n\
Ctrl<Key>o:              newline-and-backup()\n\
Ctrl<Key>m:              newline()\n\
Ctrl<Key>l:              redraw-display()\n\
Ctrl<Key>k:              kill-to-end-of-line()\n\
Any<Key>:                self-insert()
```

Table 4–5 summarizes the functions you can assign to keys and describes the text-editing action for each.

Table 4–5 Text Functions

Function	Description
forward-character	Moves the cursor forward one character.
backward-character	Moves the cursor backward one character.
forward-word	Moves the cursor forward one word.
backward-word	Moves the cursor backward one word.
forward-paragraph	Moves the cursor forward one paragraph.
backward-paragraph	Move backward one paragraph.
delete-next-character	Deletes the character after the insertion point.
delete-previous-character	Deletes the character before the insertion point.

(continued on next page)

Table 4–5 (Cont.) Text Functions

Function	Description
delete-next-word	Deletes the word after the insertion point.
delete-previous-word	Deletes the word before the insertion point.
delete-next-line	Deletes the line after the insertion point.
delete-previous-line	Deletes the line before the insertion point.
delete-selection	Deletes the selection.
insert-file	Inserts a file into the text.
kill-next-word	Kills the word after the insertion point.
kill-to-end-of-line	Kills the characters after the insertion point to the end of the line.
kill-selection	Kills the selection.
newline-and-indent	Moves the cursor to the next new line and indent.
newline-and-backup	Moves the cursor to the beginning of the previous line.
next-page	Moves the cursor to the next screenful of text.
previous-page	Moves the cursor to the previous screenful of text.
select-all	Selects all the text.
select-word	Selects the word the insertion point is in.
stuff	Inserts the text that was last selected in any window.
unkill	Inserts the text that was last deleted.

Index

A

C

S

T

U

V

W

X

How to Order Additional Documentation

Technical Support

If you need help deciding which documentation best meets your needs, call 800-343-4040 before placing your electronic, telephone, or direct mail order.

Electronic Orders

To place an order at the Electronic Store, dial 800-234-1998 using a 1200- or 2400-baud modem from anywhere in the USA, Canada, or Puerto Rico. If you need assistance using the Electronic Store, call 800-DIGITAL (800-344-4825).

Telephone and Direct Mail Orders

Your Location	Call	Contact
Continental USA, Alaska, or Hawaii	800-DIGITAL	Digital Equipment Corporation P.O. Box CS2008 Nashua, New Hampshire 03061
Puerto Rico	809-754-7575	Local Digital Subsidiary
Canada	800-267-6215	Digital Equipment of Canada Attn: DECdirect Operations KAO2/2 P.O. Box 13000 100 Herzberg Road Kanata, Ontario, Canada K2K 2A6
International	————	Local Digital subsidiary or approved distributor
Internal*	————	SSB Order Processing - WMO/E15 *or* Software Supply Business Digital Equipment Corporation Westminster, Massachusetts 01473

* For internal orders, you must submit an Internal Software Order Form (EN-01740-07).

Reader's Comments

ULTRIX WORKSYSTEM SOFTWARE
Introduction to the ULTRIX
Worksystem Software Environment
AA-MA86D-TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

Please rate this manual:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

What would you like to see more/less of? _____

_____

What do you like best about this manual? _____

_____

What do you like least about this manual? _____

_____

Please list errors you have found in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

What version of the software described by this manual are you using? _____

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Email _____ Phone _____

-- Do Not Tear - Fold Here and Tape ----------

digital™

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 33 MAYNARD MA

POSTAGE WILL BE PAID BY ADDRESSEE

DIGITAL EQUIPMENT CORPORATION
OPEN SOFTWARE PUBLICATIONS MANAGER
ZKO3-2/Z04
110 SPIT BROOK ROAD
NASHUA NH 03062-9987

- Do Not Tear - Fold Here ----------

Cut Along Dotted Line