AIX Version 4 Desktop Handbook

Document Number GG24-4451-00

December 1994

International Technical Support Organization
Austin Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xv.

First Edition (December 1994)

This edition applies to the AIXwindows Desktop which is part of AIX Version 4.1.1.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 632B Building 821 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document provides an overview for using and customizing the AIXwindows Desktop delivered with AIX Version 4.1.1. The information provided is intended to help desktop users and system administrators better understand how-to-use and customize the AIXwindows Desktop and its facilities. Each of the major components of the desktop are covered with an overview and detailed how to use information. Additionally, most components include customization information and examples that guide the reader through the necessary steps to tailor the desktop to their individual needs.

A unique feature of this document is the question cross reference appendix that lists the commonly asked questions on how to use and customize the desktop. For each question there is an associated reference that points the reader to the appropriate section in this document where the answer to the question can be found.

This document assumes that the reader has at least user level and basic customization level knowledge of AIX and AIXwindows.

(217 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help users and system administrators use and customize the AIXwindows Desktop that is delivered with AIX Version 4.1.1. The information in this publication is not intended as the specification of any programming interfaces that are provided by AIXwindows Desktop. See the PUBLICATIONS section of the IBM Programming Announcement for AIX Version 4.1.1 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
AIXwindows	CUA
IBM	InfoExplorer
Workplace Shell	

The following terms are trademarks of other companies:

Postscript	Adobe Systems Incorporated

FrameMaker	Frame Technology Corporation
Hewlett-Packard, HP and VUE	Hewlett-Packard Company
XDM, XDMCP and X-Windows	Massachusetts Institute of Technology
Motif	Open Software Foundation
SCO	Santa Cruz Operation
DeskSet, OPEN LOOK, SunSoft and ToolTalk	SunSoft Incorporated
Sun	Sun Microsystems Incorporated
USL	Unix System Laboratory Incorporated
UNIX	X/Open Company Ltd.

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to supplement the AIXWindows Desktop documentation by providing an overview and how to information on the more commonly used features of the AIXWindows Desktop in AIX Version 4.1.1.

This document is intended for AIX Version 4.1.1 users and system administrators who would like to better understand how to use and customize the AIXWindows Desktop.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction”

  This chapter describes the scope of this document as well as information on the COSE Consortium that developed the desktop specifications. Additionally, this chapter includes and overview of the desktop and its main components.

- Chapter 2, “Login Manager”

  This chapter provides an overview and customization information on the Login Manager component of the desktop. The customization information includes reference information and examples for changing the appearance of the login interface and the message of the day screen.

- Chapter 3, “Session Manager”

  This chapter provides an overview and customization information on the Session Manager component of the desktop. Included in this chapter is a description of the various sessions used by the Session Manager and how applications can automatically be restarted by the Session Manager. The customization information includes reference information and examples for changing the session and window manager started by the Session Manager.

- Chapter 4, “Front Panel”

  This chapter provides an overview and customization information on the Front Panel component of the desktop. A detailed description of the inter-workings of the front panel is provided to supply the necessary background information needed to understand the procedures and examples for altering and creating front panels included in this chapter.

- Chapter 5, “Style Manager”

  This chapter provides an overview and operation information for the Style Manager component of the desktop. Included are descriptions of the various Style Manager tools that allow the user to interactively configure much of the appearance and functionality of the desktop and its components.

- Chapter 6, “Workspace Manager”

  This chapter provides an overview and customization information for the Workspace Manager component of the desktop. This component provides the window management functions of the desktop. The customization information includes reference information and examples for creating and altering the various window manager menus and keyboard and mouse button bindings. Also included is information on migrating existing Motif window manager definitions for use by the desktop.

- Chapter 7, “File Manager”

  This chapter provides an overview and operation information for the File Manager component of the desktop. Also included is a description of the object concept used by the desktop to visually represent and manipulate the various files and directories in the filesystem.

- Chapter 8, “Launching Applications From the Desktop”

  This chapter describes how applications are setup to be started from the desktop. Additionally this chapter describes how data files can be classified into groups that associate a common appearance and set of operations to be performed on the data files.

- Chapter 9, “Application Manager”

  This chapter provides an overview and integration information for the Application Manager component of the desktop. The integrating information describes what integrating an application means and how to perform the integration into the Application Manager.

- Chapter 10, “Help Manager”

  This chapter provides an overview and operation information for the Help Manager component of the desktop.

- Chapter 11, “Icons”

  This chapter provides information about icons. Included is an example which illustrates how to use the desktop provided tool to create an icon. Then how to appropriately scale and name that icon so that it can be used by the desktop to represent actions or data files. Also included is a procedure that can be used to convert icons from other products for use by the desktop.

- Chapter 12, “Administering the Desktop”

  This chapter provides information on the directory structure used by the desktop, how to configure Xstations to use the desktop, how to print from the desktop, how the desktop locates configuration and customization files and how to setup and use the desktop in a networked environment.

- Chapter 13, “Localization”

  This chapter provides information on how to use and configure the desktop to support a native language environment.

- Appendix A, “The Desktop Directories and Files”

  This appendix provides a description for some of the various files and directories used by the desktop.

- Appendix B, “Question Cross Reference”

  This appendix provides a cross-reference to the information provided in this document in the form of questions. For each question listed, there is an associated reference to the appropriate section in this document where the question’s answer can be found.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *AIXwindows Desktop Advanced User's and System Administrator's Guide*, SC23-2671.
- *AIXwindows to AIXwindows Desktop Migration Guide*, SC23-2531.

International Technical Support Organization Publications

- *IBM Xstation Handbook*, GG24-3695.

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

International Technical Support Organization Bibliography of Redbooks, GG24-3070.

To get listings of ITSO technical publications (known as "redbooks") online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Publications

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain books on a variety of products.

Acknowledgments

The advisor for this project was:

Mark Kressin
International Technical Support Organization, Austin Center

The authors of this document are:

Angelo Aloia
IBM Italy

Roger Bedford
IBM United Kingdom

Maria-Katharina Kinsele
IBM Germany

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

David Ballard, Troy Cline, Sylvia Moore and Jon Werner
Desktop Development Team - IBM Austin

Casey Cannon
IBM Austin

Chapter 1. Introduction

The AIXwindows desktop in AIX Version 4.1 and AIX Version 4.1.1 is a partial implementation of the Common Desktop Environment (CDE) from the COSE consortium. This document covers various aspects of using, customizing and administering this partial implementation. The terms desktop and CDE are used interchangeably through this document. Both terms refer to the partial implementation of the COSE CDE specification available with AIX Version 4.1 and 4.1.1.

This section will introduce the COSE consortium and present an overview of the Common Desktop Environment.

1.1 The COSE Consortium

COSE is an acronym that stands for the Common Open Software Environment and represents a consortium of six leading UNIX vendors working to accelerate the Open System process. The participating vendors are:

- Hewlett-Packard Company (HP)
- International Business Machines Corporation (IBM)
- The Santa Cruz Operation (SCO)
- SunSoft Incorporated
- Univel
- Unix System Laboratory Incorporated (USL)

The COSE consortium was formed to accelerate the Open System Process by sharing technology between the participating vendors in an effort to develop common specifications and standards for products developed in two key areas:

- Established Technology
  - Common Desktop Environment
  - Networking
  - Graphics
- Emerging Technology
  - Multimedia
  - Object Technology
  - System Administration

These common specifications and standards are intended to help enable cost effective technology evolution. While at the same time imposing standardization on developed products in order to facilitate product interchangeability, data exchange and increased user productivity through common user interfaces.

1.2 The Common Desktop Environment

The Common Desktop Environment is actually a set of specifications jointly developed by members of the COSE consortium. The purpose of the specifications is to define a graphical desktop environment that provides a consistent look and feel across multiple UNIX platforms. Additionally, the specifications define standard Application Programming Interfaces (APIs) that application developers can write to in order to utilize the facilities of the desktop in their programs. These desktop APIs are the same across all versions of UNIX that support the CDE specifications. This is a significant advantage for application developers because it allows them write to one set of platform independent APIs and not multiple platform dependent APIs.

The technology foundation on which the CDE specification was developed is made up of the best features from the following products:

- The VUE desktop interface from HP
- Common User Interface Architecture (CUA) and Workplace Shell from IBM
- Motif from from the Open Software Foundation (OSF)
- OPEN LOOK, DeskSet productivity tools, and Tooltalk messaging system from SunSoft
- UNIX SVR4.2 Desktop Manager components and scalable systems technology from USL

This merging of these established technologies allows CDE to be a best of breed world class design developed in a short time frame with a minimal budget.

Since CDE was developed by the COSE consortium whose goal is to advance and support Open Systems, the CDE specification and associated technologies are open licensable to the software industry, thus making CDE truly a Common Desktop Environment.

The CDE specification is being implemented and is available, or soon to be available, on the following UNIX platforms:

- HP Workstations
- IBM Workstations (AIXindows Desktop in AIX Version 4 is a partial implementation)
- Sun Workstations
- Personal Computers running SCO UNIX or USL's UNIX.

1.2.1 CDE Main Components

CDE is made up of various components that interact with each other to form the desktop environment. These components are:

Login Manager

This component of CDE is responsible for providing a user login facility that authenticates and then initiates a desktop session for a given user.

For more information see Chapter 2, "Login Manager" on page 5.

Session Manager

This component of CDE is responsible for maintaining the desktop operating environment between user sessions. When a particular user logs out of the desktop the Session Manager will record which supported applications the user had running at log out time.

Note: An application must conform to the ICCM conventions for session management in order to be restarted automatically by the Session Manager. When the user logs back in, the Session Manager will automatically restart those applications for the user.

For more information see Chapter 3, “Session Manager” on page 21.

Front Panel

The front panel component of CDE is the main user interface into the desktop. Most of the facilities and features of the desktop are accessed via selections that originate from the front panel.

For more information see Chapter 4, “Front Panel” on page 31.

Style Manager

The Style Manager component of CDE allows the user to customize their individual look and feel of CDE. Through the graphical user interface of the Style Manager the user can customize the way CDE displays and uses:

- Colors
- Fonts
- Backdrops (wallpaper)
- The keyboard
- The mouse
- The internal speaker
- The screen
- Various Window Manager functions
- Various start up options

For more information see Chapter 5, “Style Manager” on page 73.

Workspace Manager

The Workspace Manager component of CDE is really the Window Manager that controls the window environment. The user routinely interacts with the window manager every time they open, close, move and re-size windows, as well as other less obvious day-to-day operations. Although it is functionally similar to the mwm (Motif window manager) that was previously available with the older versions of the AIX operating system, a few new functions have been added/changed to handle some of the new features of CDE. The most notable of these features is a new concept called workspaces. Workspaces can be thought of as virtual root windows. In the Motif window manager environment, all of the windows from your applications are opened on the same root window. The CDE desktop adds another level of virtual root windows above the real root window. Instead of having just one root window to open application windows on, you can now have as may as you would like. Each of these virtual root windows is called a workspace. The desktop allows you to easily switch between these workspaces by a simple press of

a button in the front panel. Since your workspaces are separate and distinct, you can only have one workspace displayed at any given time.

For more information see Chapter 6, “Workspace Manager” on page 87.

File Manager

The File Manager component of CDE is a tool that contains a graphical user interface which allows a user to graphically view and manipulate objects. These objects are really iconic representations of files in the AIX file system with associated actions (operations that can be performed on the files). For example, a document that was created with a word processor and stored in the AIX file system, appears in the File Manager with an appropriate icon and could be selected to edit it with the word processor or be dragged from the File Manager on to the printer control in the front panel to be printed. This combination of an icon representing a file in the file system with its associated actions make up an object which can be viewed and manipulated by the File Manager. Another way of looking at the File Manager is that it will allow you to browse the contents of the AIX file system graphically. You can then manipulate the contents of the file system through mouse and/or keyboard interaction.

For more information on the File Manager see Chapter 7, “File Manager” on page 109.

Application Manager

The Application Manager component of CDE is the application repository under CDE. Applications available to a user are graphically represented in a format similar to the File Manger.

For more information see Chapter 7, “File Manager” on page 109.

Help Manager

The Help Manager component of CDE is the help system through which the user can access assistance on using the desktop or one of its components.

For more information see Chapter 10, “Help Manager” on page 151.

Chapter 2. Login Manager

The Login Manager manages a collection of X displays, both local and 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). The Login Manager provides services similar to those provided by init, getty and login on character terminals: prompting for loginID and password, authenticating the user and starting a session.

The Login Manager is responsible for the following tasks:

- Reading initial configuration files
- Starting the X server for local displays
- Displaying the login screen and validating user login IDs and passwords
- Invoking the Common Desktop Environment Session Manager

The Login Manager is based on MIT's X Display Manager (XDM). Specifically the version of XDM that was available in X11R4.

The Login Manager has one main executable, two supporting clients and a shell script:

/usr/dt/bin/dtlogin
: This is the main Login Manager executable and is the controlling daemon. It explicitly manages any displays specified and listens on the XDMCP port for remote service requests. The dtlogin process spawns a child dtlogin process to manage each display that the Login Manager is supporting. Thus there will always be one more dtlogin processes running than displays supported by the Login Manager.

/usr/dt/bin/dtgreet
: The dtgreet executable is started by a child dtlogin to present the graphical login screen shown in Figure 1 on page 7. Here the user is able to enter a userID and password as well as select some session start-up options. Once the user has been authenticated, dtgreet terminates.

/usr/dt/bin/Xsession or /etc/dt/Xsession
: The dtlogin process then executes either the customized version in /etc/dt or the system default version in /usr/dt/bin of the Xsession configuration shell script. This script is responsible for:

  1. Loading some environment variables
  2. Running the user's personal environment variable configuration file ($HOME/.dtprofile)
  3. Running any global configuration files located in the /etc/dt/config/Xsession.d and /usr/dt/config/Xsession.d directories
  4. Running the dthello transition program
  5. Starting a messaging daemon
  6. Building the Application Manager folders
  7. Starting the user's session. This is normally the desktop Session Manager, but other types of sessions may be started if specifically specified or if the Session Manager is unavailable. The Xsession

script will first try and start the desktop Session Manager, but if that is unavailable, it will in turn try and start XDM, X11 or a default window manager with a terminal window as the user's session

/usr/dt/bin/dthello

The dthello executable is called by Xsession and provides a visual transition from the dismissal of the login screen to the start of the user session. It is normally used to display a welcome message to the user.

2.1 Customizing the Appearance of the Login Window

The graphical user interface presented by the Login Manager can be customized by the system administrator to change the appearance presented to the user. Changes to the login interface can be made by adding or modifying the values assigned to X resources that are specific to the Login Manager. These resources are stored in Login Manager configuration file (Xresources). After changing these resources, a resource in another configuration file (Xconfig) needs to be changed to point to the modified resource file. The Login Manager resources which can be specified are listed in Table 1 on page 8.

The following procedure is recommended for changing the login interface.

1. Log in as the root user.
2. If the /etc/dt/config/Xresources file does not exist, copy the /usr/dt/config/C/Xresources file to the /etc/dt/config directory. This creates a system wide customizable X resource file that contains resources that are specific to the Login Manager.

   Note: The original file in the /usr/dt/config directory is the supplied default file. By copying this configuration file into the /etc/dt directory (you may need to create the dt subdirectory of /etc) you ensure that any changes that you make to this file will be retained between desktop version upgrades.
3. Modify as desired the Login Manager X resources listed in Table 1 on page 8.
4. If the /etc/dt/config/Xconfig file does not exist, copy the /usr/dt/config/Xconfig file to the /etc/dt/config directory. This file contains an X resource (`DtLogin*resources`) which tells the desktop which X resource file is to be used to configure the Login Manager. By default, the resource points to the supplied /usr/dt/config/$LANG/Xresources default file. Since you have created a system wide customized version of this file in steps two and three, the resource needs to be changed to point to your new file, /etc/dt/config/Xresources. Just as in step one, copying the default version of the /usr/dt/config/Xconfig file into the /etc/dt directory and then modifying it, ensures that any changes that you make to this file will be retained between desktop version upgrades.
5. Force the Login Manager to re-read the Xconfig file by issuing the following command:

   ```
   kill -HUP pid
   ```

   Where pid is the process ID of the Login Manager. To obtain this ID issue the following command:

   ```
   cat /var/dt/Xpid
   ```

The returned number is the pid that should be used in the `kill` command.

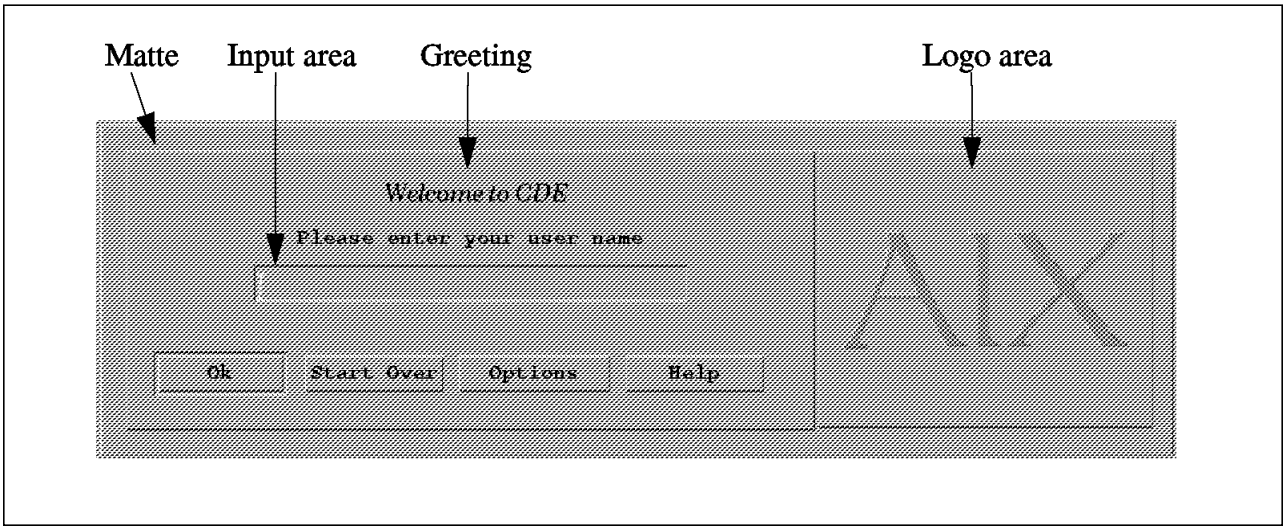


Figure 1. Default Login Manager interface

The following X resources can be added or changed in the /etc/dt/config/Xresources file to alter the visual appearance of the login screen:

Table 1 (Page 1 of 2). Login Manager Visual Appearance Resources		
Dialog window color resources - Resources that describe the colors used in the login dialog window		
Dtlogin*foreground	**Description:**	Foreground color of the login dialog window
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	black
Dtlogin*background	**Description:**	Background color of the login dialog window
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	light-gray
Dtlogin*highlightColor	**Description:**	Color used for highlighting(active input area) on the login dialog window
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	#ef506f
Font resources - Resources that describe the fonts used on the login dialog window		
Dtlogin*labelFont	**Description:**	Font used for buttons and labels
	Value Type:	Resource font string
	Default:	Displays less than 1024 pixels wide: -dt-interface system-medium-r-normal-s*-*-*-*-*-*-*-*-* Displays greater than 1024 pixels wide: -dt-interface system-medium-r-normal-l*-*-*-*-*-*-*-*-*
Dtlogin*textFont	**Description:**	Help and error dialog font
	Value Type:	Resource font string
	Default:	Displays less than 1024 pixels wide: -dt-interface user-medium-r-normal-s*-*-*-*-*-*-*-*-* Displays greater than 1024 pixels wide: -dt-interface user-medium-r-normal-l*-*-*-*-*-*-*-*-*
Cursor resource - Resource that describes the mouse pointer shape used on the login dialog window.		
Dtlogin*workspaceCursor	**Description:**	Specifies either the MIT ″X″ or the left pointer cursor
	Value Type:	True for left pointer or False for the MIT ″X″ cursor
	Default:	True
Name and password resources Resources that describe the size of the name and password input areas		
Dtlogin*login_text.columns	**Description:**	Login name input area size
	Value Type:	Integer number of columns to allow for input
	Default:	20
Dtlogin*passwd_text.columns	**Description:**	Password input area size
	Value Type:	Integer number of columns to allow for input
	Default:	20
Language menu name mappings		
Dtlogin*xx_XX.languageName	**Description:**	Maps language name xx_XX to the specified character string
	Value Type:	Text string of your choice
	Default:	Usually Language name - Code set (eg. English (United States) - IBM-850)
Greeting resources - Resources describe the greeting used on the login and password screens		
Dtlogin*greeting.foreground	**Description:**	Foreground Color of the greeting text
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	black
Dtlogin*greeting.background	**Description:**	Background Color of the greeting area
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	light-gray

Table 1 (Page 2 of 2). Login Manager Visual Appearance Resources		
Dtlogin*greeting.labelString	**Description:**	Specifies the string to use for the welcome message. Multiple lines can be specified by including newline characters (\n) in the text. If the token %LocalHost% is included in the text, it will be replaced with the name of the host providing login service. If the token %DisplayName% is included in the text, it will be replaced with the display name
	Value Type:	Text string of your choice
	Default:	Welcome to %LocalHost%
Dtlogin*greeting.persLabelString	**Description:**	Specifies the string to use for the welcome message on the password screen. Multiple lines can be specified by including newline characters (\n) in the text. If the token %s is included in the text, it will be replaced with the user name entered on the login screen
	Value Type:	Text string of your choice
	Default:	Welcome %s
Dtlogin*greeting.alignment	**Description:**	Alignment of the welcome message
	Value Type:	AlIGNMENT_LEFT, ALIGNMENT_Center or ALIGNMENT_RIGHT
	Default:	ALIGNMENT_CENTER
Dtlogin*greeting.fontlist	**Description:**	Font to be used for the welcome message
	Value Type:	Resource font string
	Default:	Displays less than 1024 pixels wide: -dt-interface system-medium-r-normal-xl*-*-*-*-*-*-*-*-* Displays greater than 1024 pixels wide: -dt-interface system-medium-r-normal-xxl*-*-*-*-*-*-*-*-*
Logo resources - Resources that define the appearance of the logo and surrounding area		
Dtlogin*logo*bitmapFile	**Description:**	Specifies the absolute path name to the bitmap or pixmap file to be used for the logo
	Value Type:	Bitmap or pixmap file name
	Default:	/usr/dt/appconfig/icons/C/Dtlogo.pm
Dtlogin*logo*Background	**Description:**	Background color for the logo
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	White
Dtlogin*logo*topShadowPixmap	**Description:**	Specifies the pixmap to use for the logo border shadow
	Value Type:	Bitmap file name
	Default:	25_foreground
Dtlogin*logo*x	**Description:**	X offset for the logo from the left edge of the logo area
	Value Type:	Number of pixels
	Default:	0
Dtlogin*logo*y	**Description:**	Y offset for the logo from the top edge of the logo area
	Value Type:	Number of pixels
	Default:	0

2.1.1 Customizing the Login Manager Example

This example will modify the default Login Manager interface from the one pictured in Figure 1 on page 7 to the version shown in Figure 2 on page 10.

The specific changes are:

- Change the logo to a new logo
- Change the welcome greeting to read ″Howdy! Welcome to Texas″
- The name of the user′s display will be used in the above greeting instead of the default local host name. This will allow each user to have a greeting that

identifies their own display. In this example we are on a display named Texas

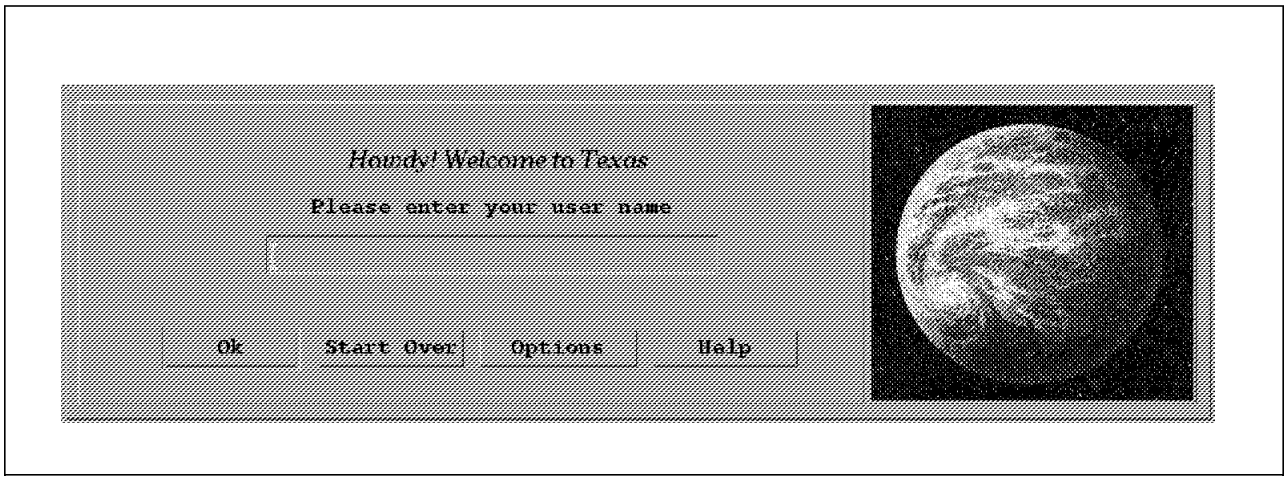


Figure 2. New Login Manager Interface

The procedure shown in 2.1, “Customizing the Appearance of the Login Window” on page 6 was used to change the following Login Manager resources in the file /etc/dt/config/Xresources:

1. The line containing `Dtlogin*logo*bitmapFile` was was modified by removing the two ! characters at the beginning of the line. The name of our desired pixmap file was subsituted for the the text on the same line that reads:

   `< bitmap or pixmap file >`

   Note: The logo file must be in a bitmap or pixmap format. In our case, our original image was in a gif format. The gif format was converted to pixmap using an appropriate tool. See 2.2, “Changing the Logo on the Login Screen” on page 11 for one possible method of converting an image file, such as a gif file, to a pixmap file.

   The new line looks like:

   `Dtlogin*logo*bitmapFile: /usr/local/bitmaps/Texas.pm`

2. The line containing `Dtlogin*greeting*labelString` was modified by removing the two ! characters at the beginning of the line. Our new string `"Howdy, Welcome to %DisplayName%"` was entered to replace the old greeting.

   The new line looks like:

   `Dtlogin*greeting*LabelString: Howdy, Welcome to %DisplayName%`

Note:

Because the display name for the console is set by default to :0, it is necessary to make a separate Xresources file for the console than for the Xstations in order to avoid having the console greeting read "Howdy! Welcome to :0". This is done by:

1. Copying the /etc/dt/config/Xresources file to /etc/dt/config/Xresources.console
2. Renaming the /etc/dt/config/Xresources file to /etc/dt/config/Xresources.xstations
3. Editing the /etc/dt/config/Xconfig file and changing the resources line to point the Xresources file for Xstations. The line should read:

   ```
   Dtlogin*resources: /etc/dt/config/Xresources.xstations
   ```

   And then duplicate the resources line and modify the second one to point to the version of the Xresources file for the console. The line should read:

   ```
   Dtlogin*_0*resources: /etc/dt/config/Xresources.console
   ```

4. Edit the console Xresources file: /etc/dt/config/Xresources.console to further modify the greeting line changed above to read:

   ```
   Dtlogin*greeting*LabelString: Howdy, Welcome to %LocalHost%
   ```

   This will cause the greeting on the console to display correctly.

2.2 Changing the Logo on the Login Screen

The Login Manager is capable of displaying X11 pixmaps or bitmaps in the logo area of the login user interface. Images that are not in a pixmap or bitmap format must be converted into a pixmap or bitmap through several available tools. If the image is less than 256x256 pixels in size then the the desktop icon editor (dticon) along with the following procedure can be used to convert the image:

1. Find an appropriate tool to display the image to be converted in an X window
2. Bring up the desktop icon editor by selecting its icon. You can usually find the icon on either the **Personal Applications** subpanel or in the **DesktopTools** application group found under the **Application Manager**.
3. With both the icon editor and the image to be converted displayed on the same screen, select the **Grab Screen Image** option from dticon's **Edit** pull down menu. This action will allow you to use the left mouse button to outline the image your are trying to convert for use as the new login logo. When you release the mouse button the area you outlined will be copied into the icon editor.
4. You can then modify the image using the icon editor as desired. For more information see Chapter 11, "Icons" on page 161.
5. Select the desired output format (bitmap or pixmap) by selecting your desired choice on the **Output Format** option of dticon's **Options** menu.
6. Select **Save As ...** from dticon's **File** pulldown menu
7. Enter a name and location to store the image file
8. Click **OK** to save the file.

Note: You may experience an error about exceeding the maximum number of colors when doing the screen grab in step 3. If this happens, use an image editor to dither the image to use fewer colors and then retry the grab again.

Now that you have your desired image in a bitmap or pixmap format, see 2.1, “Customizing the Appearance of the Login Window” on page 6 for information on how to modify the Login Manager’s resource (`Dtlogin*logo*bitmapFile`) to include your new logo. You can also refer to the example in 2.1.1, “Customizing the Login Manager Example” on page 9.

2.3 Customizing the Message of the Day

The transition screen displayed between user login and session start up is called the message of the day (motd) screen and contains both a transition message and an optional message of the day. The transition message is usually a short message that welcomes the user to the desktop environment. The message of the day can be a much longer message that notifies the user of some pertinent information. These messages, with associated display colors and font, can be modified by using either of the following procedures:

The program that displays the motd screen is called dthello. It can be customized in either one or both of these places:

- /etc/dt/app-defaults/$LANG/Dthello
- /etc/dt/Xsession

2.3.1.1 Changing /etc/dt/app-defaults/$LANG/Dthello

The /etc/dt/app-defaults/$LANG/Dthello file is a local customization copy of the system default /usr/dt/app-defaults/C/Dthello X resource file. The X resources in the /etc/dt/app-defaults/$LANG/Dthello file can be modified to change the behavior of dthello. This method of motd customization is recommended unless you need to concatenate multiple motd text files to form a single motd message. See 2.3.1.2, “Changing /etc/dt/Xsession” on page 13 for information on specifying multiple motd files. The following table shows the available dthello resources that can be modified to change the appearance of the motd screen.

Table 2 (Page 1 of 2). Message of the Day Resources

Color resources - Resources that define the colors used on the message of the day screen		
Dthello*vforeground	**Description:**	Foreground color of the text on the motd window
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	dynamic (chosen based on visual type of the screen)
Dthello*vbackground	**Description:**	Background color of the motd screen
	Value Type:	Predefined color name or RGB hex value(ie.#RRGGBB)
	Default:	dynamic (chosen based on visual type of the screen)
Font resources - Resources that describe the font used to display the motd text		
Dthello*vfont	**Description:**	Font used for transition message and motd text
	Value Type:	Resource font string
	Default:	dynamic

Table 2 (Page 2 of 2). Message of the Day Resources		
Message resources - Resources that define the text to be displayed on the motd screen		
Dthello*string	**Description:**	Specifies the transition message text string to be displayed first on the motd screen
	Value Type:	Character string. May include (\n) character if a multi-line message is desired
	Default:	Opening the AIXwindows Desktop
Dthello*file	**Description:**	Specifies an ASCII text file whose contents are displayed concatenated to the bottom of the transition message **Note:** Only one file may be specified using this resource. If more than 1 file is desired, use the Xsession method described in 2.3.1.2, "Changing /etc/dt/Xsession" on page 13. Also if your file contains a lot of text for the user to read, see 3 on page 15 for a few things that you need to consider.
	Value Type:	Full path name of the motd ascii text file
	Default:	none

The following is the recommended procedure for modifying the /etc/dt/app-defaults/$LANG/Dthello application defaults file.

1. If the /etc/dt/app-defaults/$LANG/Dthello file does not exist, copy the system default version of the /usr/dt/app-defaults/$LANG/Dthello file to the /etc/dt/app-defaults/$LANG directory using the following commands:

   ```
   mkdir /etc/dt
   mkdir /etc/dt/app-defaults
   mkdir /etc/dt/app-defaults/$LANG
   cp /usr/dt/app-defaults/C/Dthello /etc/dt/app-defaults/$LANG
   ```

   The first three commands make the /etc/dt/app-defaults/$LANG directory and are only needed if this directory has not been previously created.

   Note: Making a copy of the configuration files in the /etc/dt directory ensures that any changes that you make to these files will be retained between desktop version upgrades.

2. Edit the /etc/dt/app-defaults/$LANG/Dthello file and add or change any of the desired resources.

3. Save the changes to the file.

The next time the motd screen is displayed, the changes will be visible.

2.3.1.2 Changing /etc/dt/Xsession

The /etc/dt/Xsession file is a local customization copy of the system default /usr/dt/bin/Xsession file. You may recall from the beginning of this chapter that the Xsession file is the shell script that calls dthello and thus can be modified to change the way dthello is called.

Caution

The Xsession file is a very complicated shell script that is crucial to the operation of the desktop. Be very careful when modifying this file to only make the described changes. Other alterations can cause undesirable results. Further more, the changes that you make to this file will have to be re-done when you upgrade versions of the desktop in order to pickup any changes that may have occurred in the system default version of this file.

The dthello program accepts the following arguments:

-display <display>
Specifies the X display to display the motd screen on. Since it is recommended that dthello is only called by CDE and not by a user, this argument should not be specified by the user.

-fground <color>
Specifies a color to use for the foreground on the motd screen. For example: -fground red.

-bground <color>
Specifies a color to use for the background on the motd screen. For example: -bground blue.

-fnt <fontname>
Specifies the font to used for the the text on the motd screen. For example: -fnt rom10.

-string <message>
Specifies the transition message to be displayed.

-file <fullpath filename>
Specifies a ASCII text file that contains the message to be displayed. The contents of the file is appended to the default welcome string or the specified value of the -string argument.

Note: If you desire to have only the contents of the file displayed, you need to specify -string "". Also note that leading blanks on text lines in the file are ignored.

You can specify up to 5 files to be concatenated together to form the motd. Each file name must be preceded by the -file argument. See the example in 2.3.2, "Example of Changing the Message of the Day Screen" on page 16.

-timeout <seconds>
Specifies the number of seconds dthello will wait before giving up on the window manager to process the window. Since it is recommended that dthello only be called by CDE and not by a user, this should not normally be specified by the user.

The following is the recommended procedure for modifying the Xsession file.

1. If the /etc/dt/Xsession file does not exist, copy the /usr/dt/bin/Xsession file to the /etc/dt using the following commands:

   ```
   mkdir /etc/dt
   cp /usr/dt/bin/Xsession /etc/dt/Xsession
   ```

   The first command makes the /etc/dt directory and is only needed if this directory has not been previously created.

   Note: Making a copy of the configuration files in the /etc/dt directory ensures that any changes that you make to these files will be retained between desktop version upgrades.

2. Edit the file /etc/dt/Xsession and locate the line that reads:

   ```
   DTHELLO_ARGS=""
   ```

   Insert the desired arguments from above (except `-string`) between the set of double quotes. If you plan to specify the `-string` argument, you need to put it directly on the line that calls dthello. This is necessary to avoid problems

with quoted (″ ″) strings within a quoted string under the shell. To include the -string argument locate the following line in the file and add your -string argument before the & at the end of the line. The second line is a completed example.

```
$DTHELLO $DTHELLO_ARGS &
$DTHELLO $DTHELLO_ARGS -string ″New message here″ &
```

Caution

If the arguments that you specify as part of the DTHELLO_ARGS string do not fit on one line, you may use the \ character at the end of a line as a continuation. But be careful that the \ is the last character on the line. If an inadvertent space is left after the \, dthello will not be called.

3. If you have a message of the day that contains a lot of text to be read by the user there are two things that you may need to consider:
   - You can insert a sleep command to cause the system to sleep, giving the user a chance to read the motd. To do this, locate the line that calls dthello and reads:

     ```
     $DTHELLO $DTHELLO_ARGS &
     ```

     Add the following line right below it:

     ```
     sleep xx
     ```

     Where xx is the number of additional seconds you want to give the user to read the motd.
   - You may also need to change the font that is used to display the text so that all of your message will fit on the screen.
4. Save the changes to the file.
5. In order for the desktop to find your changes, you also need to modify the Xconfig file to point to your modified version of the Xsession file. To do this:
   a. If the /etc/dt/config/Xconfig file does not exist, copy the /usr/dt/config/Xconfig file into the /etc/dt/config directory using the following commands.

      ```
      mkdir /etc/config
      cp /usr/dt/config/Xconfig /etc/dt/config/Xconfig
      ```

      The first command makes the /etc/dt/config directory and is only needed if this directory has not been previously created.

   b. Add or change the DtLogin*session line in the /etc/dt/config/Xconfig file to point to /etc/dt/Xsession. For example:

      ```
      DtLogin*session:  /etc/dt/Xsession
      ```

   c. Save the file.

   d. Force the Login Manager to re-read the Xconfig file by issuing the following command:

      ```
      kill -HUP pid
      ```

      Where pid is the process ID of the Login Manager. To obtain this ID issue the following command:

      ```
      cat /var/dt/Xpid
      ```

      The number returned from this command is the pid that should be specified on the kill command.

The next time the motd screen is displayed, the changes will be visible.

2.3.2 Example of Changing the Message of the Day Screen

The following example illustrates a multi-file message of the day. Because we are using more than one file to specify the message of the day, the Xsession method was used. For more information on the Xsession method see 2.3.1.2, “Changing /etc/dt/Xsession” on page 13.

This example sets up the message of the day so that 3 different departments are able to contribute to the displayed message.

1. The first department is the system administration department and their message file is called /u/sysadmin/motd and contains the following text:

```
Welcome CDE Users
------------------------------------------------------------------------
System Notes                                            Date: 1/15/1995

System Status:
--------------
System A - UP
System B - UP
System C - DOWN for Maintenance

Printer Status:
---------------
PSA     - UP
PSB     - UP
PSOLOR  - UP Note: RESERVED for Marketing from 13:00
                   to 15:00 in order to print the new brochure.
```

2. The second department is site security. Their file is named /u/security/motd and contains:

```
------------------------------------------------------------------------
Notes from Security

**** There is a Green Honda Accord Licence: XXX-123 ****
**** in the parking lot with its lights on.         ****

Please try and remember that the site will be closed this
weekend for heating system maintenance.
```

3. The last department is the cafeteria. Their file is called /u/cafe/motd and contains:

```
------------------------------------------------------------------------
Today's Menu

Entree:               Vegetable:
------------          ------------
Roast Chicken         Fresh Peas
Spaghetti             Squash
Pot Roast             Corn on the Cob

Desert:               Hours: 11:00am to 1:30pm
--------------
Chocolate Cake
```

```
    Peach Yogurt
    Cherry Cobbler
```

The entry in /etc/dt/Xsession looks like this:

```
DTHELLO_ARGS=" -file /u/sysadmin/motd \
               -file /u/security/motd \
               -file /u/cafe/motd -fnt rom10"
```

Because we only want the motd message to be displayed and not the transition message, we changed the line that calls dthello to include the -string argument with a null parameter. See 2 on page 14 more information.

```
$DTHELLO $DTHELLO_ARGS -string "" &
```

Because of the length the resultant motd message, the following sleep command was also inserted into the Xsession file to hold the transition screen long enough to give the user a chance to read the messages.

```
sleep 15
```

The message of the day will now look like this:

```
Welcome CDE Users
-------------------------------------------------------------------------
System Notes                                              Date: 1/15/1995

System Status:
--------------
System A - UP
System B - UP
System C - DOWN for Maintenance

Printer Status:
---------------
PSA     - UP
PSB     - UP
PSOLOR  - UP Note: RESERVED for Marketing from 13:00
                   to 15:00 in order to print the new brochure.

-------------------------------------------------------------------------
Notes from Security

**** There is a Green Honda Accord Licence: XXX-123 ****
**** in the parking lot with its lights on.         ****

Please try and remember that the site will be closed this
weekend for heating system maintenance.

-------------------------------------------------------------------------
Today's Menu

Entree:               Vegetable:
------------          ------------
Roast Chicken         Fresh Peas
Spaghetti             Squash
Pot Roast             Corn on the Cob

Desert:               Hours: 11:00am to 1:30pm
--------------
Chocolate Cake
Peach Yogurt
Cherry Cobbler
```

2.4 Accessing the Desktop Without Using the Login Manager

The desktop can be started without using the the Login Manager. AIX Version 4 can be configured to automatically start the Login Manager or present the user with a command line login prompt. If the command line prompt option is selected, the Login Manager is not started by the system. The desktop can then be started by initializing the X Windows server from the command line and specifying the desktop as the client to start. After X initializes, the desktop will be started and available for use. When you exit the desktop, the X server will be killed and you will be returned to the command line prompt.

For example:

The following procedure will switch AIX from automatically starting the Login Manager to offering the command line login. Then from the command line, the desktop will be started:

1. Change the login interface from the desktop to command line by using the following procedure:
   a. Log in or su to become the root user.
   b. Start up SMIT.
   c. Select the **System Environments** option from the SMIT menu.
   d. Select the **Change System User Interface** option from the System Environments menu.
   e. Change the Select System Login User Interface parameter from `CDE environment` to `Command line`
   f. Press Enter
   g. Exit SMIT and reboot the system.
2. Log in through the command line interface as the desired user.
3. At the system prompt, enter the first command if the /etc/dt/Xsession file exists or enter the second command if /etc/dt/Xsession does not exist. You may remember that /etc/dt/Xsession will only exist if the default version of the file has been customized by your system administrator.

   ```
   xinit /etc/dt/Xsession
   xinit /usr/dt/bin/Xsession
   ```

The desktop will now start and be available for use. When you log out of the desktop, the X server will terminate and you will be returned to the command line prompt.

2.5 Using the Desktop Installed on Another System

Another system on the network that has the desktop installed can be used as your desktop server. If the remote desktop system has the Login Manager running it will accept a login request from your local system.

The following procedure will illustrate how to make this connection:

1. Verify that the desktop and the Login Manager are running on the remote system.
2. Log in to the local system through the command line interface.

If you are using AIX Version 4 on your local system, select the Command Line Login option from the menu under the Options button on the desktop login interface. Then press the Enter key to start the AIX command line login facility.

3. At the system prompt enter the following command:

   ```
   /usr/bin/X11/X -query <remote.host>
   ```

   Where remote.host is the host name of the remote desktop server. For example:

   ```
   /usr/bin/X11/X -query cde.server
   ```

   This will cause the system with the host name cde.server to display the desktop Login Manager interface on your local system.

Chapter 3. Session Manager

The Session Manager is the component of the desktop software that is responsible for initializing each user's desktop session after login. It will automatically configure the desktop to either reflect the way the desktop was the last time the user logged out or to a configuration that was previously saved by the user. These two automatic configuration possibilities are called the current session and the home session respectively. These sessions are in essence snapshots of the way the desktop was when the session was saved and contain information on:

- Applications running at the time the session was saved
- Any desktop customization changes that were made by the user
- The values of all of the X resources known to the X resource manager

Each time a user logs out of the desktop, the Session Manager takes a snapshot of the desktop just before logging off the user. This snapshot is saved as the current session and can be used the next time the user logs in to recreate the desktop to look the way that it did when the user logged out.

Similarly, the user can request the Session Manager at anytime to take a snapshot of the desktop and store it as the home session. For more information on saving the home session see 3.1, "Customizing the Session Manager" on page 22.

The current and home sessions are illustrated in the following example:

> The desktop can be setup to prompt the user at logout for the desired session (home or current) to be loaded the next time that they log in. With this option enabled, the home session could be selected at the end of the day to configure the desktop into a state that the user could use when they login in the morning. This session could include the applications that the user needs to use every morning. Then later, when the user leaves for lunch they could select that current session be loaded when they return from lunch. This would put the desktop back the way that it was before leaving for lunch. This enables them to pick up where they left off before going to lunch. At the end of the day the user could once again, select the home session option to return the desktop to the configuration needed when they log in the following morning. For more information on selecting the desired start up session see 3.1, "Customizing the Session Manager" on page 22.

The Session Manager saves the session snapshot information in session files located in each user's home directory. The files are called:
For the current session:

```
$HOME/.dt/sessions/current/dt.resources - X resources
$HOME/.dt/sessions/current/dt.session   - Running applications
$HOME/.dt/sessions/current/dt.settings  - Desktop customizations
$HOME/.dt/sessions/current/dtxxxxx      - Application state information
```

For the home session:

```
$HOME/.dt/sessions/home/dt.resources  - X resources
$HOME/.dt/sessions/home/dt.session    - Running applications
$HOME/.dt/sessions/home/dt.settings   - Desktop customizations
$HOME/.dt/sessions/home/dtxxxxx       - Application state information
```

Where dtxxxxx is a randomly generated name which is given to each application state information file that is created by the Session Manager on behalf of each application that is working with the Session Manager to store its state information. You may find several of these files in either or both of the session directories.

Note: The names and functions of these files are provided for information purposes only. All of these files are maintained by the Session Manager and must not be modified by the user.

In summary, the Session Manager saves and restores the user's desktop automatically for each user. Because of this, the changes a user makes to their desktop environment, to either customize it to their preferences or by starting up their commonly used applications, are not lost each time the user logs out. This eliminates the need to start over each time the user logs back in.

3.1 Customizing the Session Manager

The following customizations can be made by each user for their own desktop:

- Select which session to use at login

  The session (current or home) used by the Session Manager at login is set through the Startup option of the Style Manager (pictured in Figure 3 on page 23). There are three choices:

  – Resume current session - This option will restore the desktop environment to the way that it was when the user last logged out.

  – Return to Home session - This option will restore the desktop environment to the way that it was when the home session was saved. See Save the home session below for information on how to save the home session.

  – Ask me at Logout - Selecting this option will cause the desktop to prompt the user during log out to select which session (current or home) the Session Manager should use the next time the user logs back in.

- Save the home session

  The home session is saved on the Startup option of the Style Manager. Whenever the **Set Home Session ...** option is selected, the current desktop environment (running applications, X resources and desktop configuration) is saved as the home session. If the home session is selected to be started by the session manager at login (see above), the desktop environment will be restored to this state regardless of what further changes the user makes before logging out.

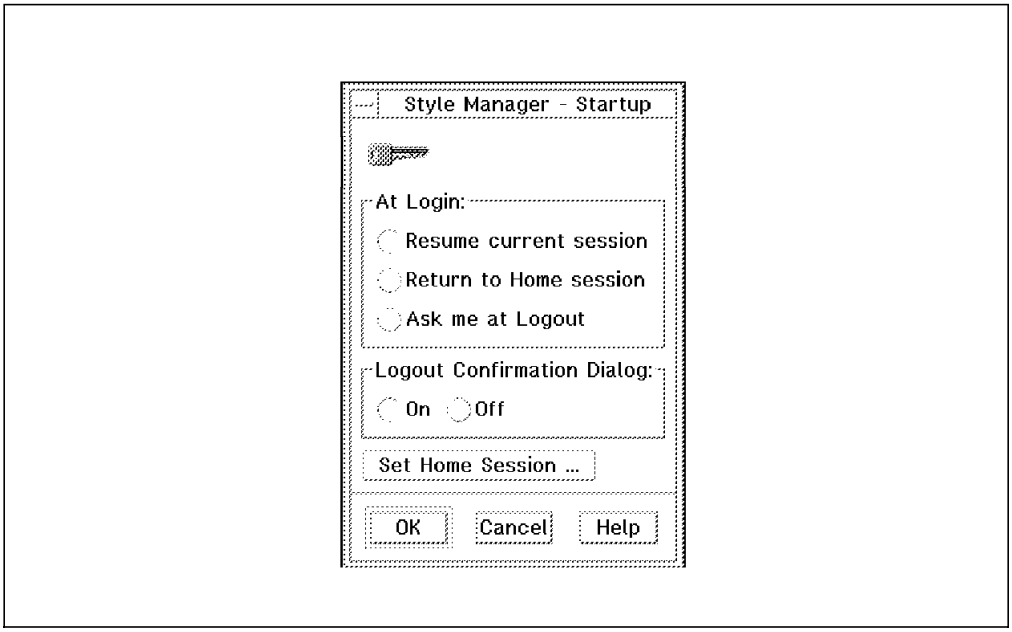


Figure 3. Start up Option of the Style Manager

- Additional commands can be run automatically after login

  If you have additional commands or applications that you want the Session Manager to execute after start up, you can place them in the file:

  `$HOME/.dt/sessions/sessionetc`

  This file is not supplied as part of the desktop. If you wish to use this facility, you will need to create this file. Make sure that you create the file with execute permission. This file is a shell script that will be executed by the Session Manager during the login process. For example, if you want to use the xsetroot command to set your root window cursor you could put the following lines in the sessionetc file:

  ```
  #!/bin/ksh
  xsetroot -cursor_name gumby -fg blue -bg red &
  ```

  Note: Commands must be run in the background (& parameter). Also, do not put commands in this file to start applications that are going to be started by the Session Manager automatically as a result of being in the selected (home or current) session file.

- Commands can be run automatically at logout.

  If you have commands that you want the Session Manager to execute after the user has initiated the logout process, you can place them in the file:

  `$HOME/.dt/sessions/sessionexit`

  This file is not supplied as part of the desktop. If you wish to use this facility, you will need to create this file. Make sure that you create the file with execute permission. This file is a shell script that will be executed by the Session Manager during the logout process. For example, you may want to use this facility to clean up any temporary files that you may have created with your applications during your desktop session.

  Note: The commands in the sessionexit file can not be X Window applications or require any user interaction. This is because the desktop is in the process of logging out the user and has terminated their session.

- Using an alternate window manager

  The choice of which window manager the desktop uses can also be customized by setting the: dtsesson.wmStartupCommand resource. The default window manager is the dtwm. For more information on dtwm see Chapter 6, “Workspace Manager” on page 87. However if you want to use a different window manager such as the Motif window manager (mwm), you can change this resource to point to the executable of the desired window manager. For example to change the desktop to use the Motif window manager:

  1. Add or modify the the wmStartupCommand resource in your .Xdefaults file look like the following:

     ```
     dtsesson.wmStartupCommand: /usr/bin/X11/mwm
     ```

     See 3.5, “The Session Manager’s Role With Respect to X Resources” on page 29 for important information about using the .Xdefaults file.

  2. The next time you log out and then back in, the desktop will be using the Motif window manager instead of dtwm.

- Controlling what is stored in the session files

  You should not modify the session files created by the Session Manager by editing them. You can indirectly customize the information that is stored in the session files by setting up the desktop environment as desired before the Session Manager takes the session snapshot.

  For example, if you want certain X resources to be set each time you log in. You can set them before you save your home session or log out. This will ensure that the Session Manager will restore them the next time you log in. Similarly, if you want to have a certain application started each time you log in, you can have it up on the desktop when you set your home session or log out. If the application supports session management the application will automatically be restarted by the session manager. For more information on applications that support session management see 3.2, “Applications Which Can Be Restarted By the Session Manager.”

3.2 Applications Which Can Be Restarted By the Session Manager

The Session Manager will automatically start after a user login any applications that support the ICCCM Session Management protocol and were running when the user’s selected session file (current or home) was created.

When a user logs off of the desktop or saves the home session, the Session Manager locates all the X-Windows applications that the user had running when the log out process was initiated. For each of these applications the Session Manager checks to see if the application has stored its start up command string in the XA_WM_COMMAND property of the application’s toplevel window. If this has been done, the Session Manager will save this start up string in the dt.session file of either the current or home session depending on whether the user is logging out or saving their home session. Later when the user logs back in, using the session file (current or home) that contains the application start up command string, the Session Manager will issue the start up command string on behalf of the user to re-launch the application.

There are two ways that an application can set its start up command string in the XA_WM_COMMAND property of its toplevel window:

1. It can be set automatically by the X-Windows toolkit when the application is initialized. This is generally done when the application developer uses the appropriate toolkit call to initialize the toolkit and passes the applications command line parameters to this subroutine call. For example the following call will initialize the toolkit and create an application toplevel window with the XA_WM_COMMAND property set to the command string used to start up the application (argv[0]).

   ```
   topLevel = XtAppInitialize(&appContext, "MyApplication",
                              (XrmOptionDescList)NULL, 0, &argc, argv,
                              fallbackResources, (ArgList)NULL, 0);
   ```

   A lot of applications will use this or a similar technique to create their toplevel window. If they do, the Session Manager will be able to pick up the application's start up command string, store it in a session file and later re-launch the application for the user at login.

2. An application developer can register to receive the WM_SAVE_YOURSELF message and explicitly set the startup command string using the XSetCommand X-Windows subroutine call. The registration process (shown in the following example) is usually done as part of the application initialization process. The setting of the startup command string is generally done in a callback routine that is invoked when the application receives a WM_SAVE_YOURSELF message from the Session Manager. See the following example for an example of such a callback routine. The WM_SAVE_YOURSELF message indicates that the user has requested that X-Windows shutdown the application. This will happen when a user logged out while an application is running. As a result of this, The Session Manager will send the WM_SAVE_YOURSELF message to all applications that have registered to receive this message. When each registered application receives the WM_SAVE_YOURSELF message it should then store it's start up command string in the XA_WM_COMMAND property of it's top level window. The Session Manager is then able to pick up this command string and store it in the session file.

   The reason that some application developers will choose to use this second method for setting the starting command string is that this method allows them to dynamically create the startup string to pass some information back to themselves when the application is restarted. This information can include a clue on the current state of the application before it was shut down so that it can be returned to that state when it is re-launched. For example, the desktop File Manager does this so that the directory being displayed in it's window at logout is the same directory that is re-displayed when you log back in. The following code example uses this method:

   ```
   #include <Xm/Protocols.h>

   Atom XaWmSaveYourself;
   Display *dpy;
   void saveYourselfCB();
   char *current_directory;

   dpy = XtDisplay(toplevel);
   /* Setup and register to receive the WM_SAVE_YOURSELF message */
   XaWmSaveYourself = XmInternAtom(dpy,"WM_SAVE_YOURSELF", False);
   XmAddWMProtocols(toplevel, &XaWmSaveYourself, 1);
   ```

```
/* Register the callback routine to handle */
/* the WM_SAVE_YOURSELF message */
XmAddWMProtocolCallback(toplevel,XaWmSaveYourself,
                        saveYourselfCB,toplevel);
.
.
.


void saveYourselfCB(widget, client_data, cbs)
Widget widget;
XtPointer client_data;
XmAnyCallbackStruct *cbs;
{
 /* Setup the command string array to contain the re-launch command */
 char *command[3]={"/usr/local/bin/MyDirViewer","-directory",""};

 /* Store the pointer to the currently being displayed */
 /* directory as the third element in the commands array */
 command[2] = current_directory;

 /* Set the WM_COMMAND property on the top level */
 /* window for command re-start to the current state.*/
 XSetCommand(dpy, XtWindow(toplevel), command, 3);
}
```

Assuming that the MyDirViewer application was displaying the contents of the directory /usr/lpp/X11 at logout. The above code segment will cause the following start up command string to be saved in the current session file by the Session Manager:

```
/usr/local/bin/MyDirViewer -directory /usr/lpp/X11
```

You can use the xlsclients X-Windows sample program to see if an application will support being re-started by the Session Manager. The xlsclients command will display the startup command string for each application which is running when the command was issued. This string should have been stored by each application in the XA_WM_COMMAND property of its toplevel window. You may find that some applications do not do this. The xlsclients command will show which running applications have done this. If an application has an entry in the output of the xlsclients command like the following, it has stored its startup string as necessary to be restarted:

```
starlight.austin.ibm.com /usr/dt/bin/aixterm
```

This example shows that an aixterm was running and set the above startup command string in the XA_WM_COMMAND property of its toplevel window.

If a running application does show up in the output of the xlsclients command, you can use the xprop X-Windows sample program to check if the application has registered to receive the WM_SAVE_YOURSELF message. Simply issue the xprop command and select the toplevel window of the application in question. If you see an entry for WM_SAVE_YOURSELF on the line that looks like the following:

```
WM_PROTOCOLS(ATOM): protocols
```

The application has registered to receive the WM_SAVE_YOURSELF message. Although this does not guarantee that the application will store its startup command string in the XA_WM_COMMAND property at log out, it does give a pretty good indication that the application might support it.

Of course, the simplest way to find out if an application supports being restarted by the Session Manager is to simply have the application up when you log out. Then see if the Session Manager can restart it when you log back in.

3.3 Display Dependant Sessions

The Session Manager supports display dependant sessions. This means that a particular user can configure the Session Manager to save and restore different current and home sessions depending on which display they are currently using. For example, you may be using a system console with an 19 inch color display at one work location and an Xstation with a 14 inch monochrome display at another. When you log in with your user ID at either work location, you would like the Session Manager to configure your desktop appropriately for the workstation (console vs Xstation) that you are logging in on. You can do this by using the following procedure:

1. Log in with your userID on one of your workstations to be setup as a display dependent session. In the above example, you might log in on the console.
2. Configure the desktop the way you would like to use it from this workstation. For example use the Style Manager to set your desired colors and fonts. You may also want to start up applications that you would like the Session Manager to restart for you at login. Remember the application must support being restarted by the Session Manager. See 3.2, “Applications Which Can Be Restarted By the Session Manager” on page 24 for more information.
3. Determine your hostname and X Window display number by issuing the following command from a terminal window:

   ```
   echo $DISPLAY
   ```

   The output might look like this if you are on display 0 of the host named starlight:

   ```
   starlight.austin.ibm.com:0.0
   ```

   The name before the either the first period or the colon, which ever comes first in the output, is your unqualified hostname. The first number after the colon is your X Window display number. If you are on a system console your output might look like this:

   ```
   unix:0.0 or :0.0
   ```

   In this case, you can still get the X Window display number from the output, but the hostname will have to be determined using the output of the `hostname` command. Just as in the `echo` command above, the unqualified hostname is the first name before the first period.
4. Log out of the desktop.
5. Log into the desktop using the single window session. For information on single window sessions, see 3.4, “Single Window Sessions” on page 28.
6. Copy the $HOME/.dt/sessions directory to a directory that contains your unqualified hostname and your X Window display number. For example, for the host name starlight, and X Window display number zero you would copy $HOME/.dt/sessions to $HOME/.dt/starlight:0 using the following command:

   ```
   cp -r $HOME/.dt/sessions $HOME/.dt/starlight:0
   ```
7. Log out of the single window session.

From now on when you log in and out on this display you will be retrieving and storing your session information in the files that you set up for this display.

You can repeat the above procedure for any other workstations that you want to be display dependant.

Please remember the following when you use display dependent sessions:

When you log out on one display and back in on another display, the Current Session, which will be restored by the Session Manager, will be the one that was saved the last time you logged on at this display. Not the one that was saved the last time you logged out. Remember you last logged on at another display - not this one.

3.4 Single Window Sessions

The single window session is a special session that the Login Manager provides. It allows you to log in to the system and correct configuration errors you may have made in the Session Manager configuration files. It is possible to introduce errors into these configuration files which will prevent the Session Manager from starting up your desktop session when you log in. If this happens you can:

1. Log in by selecting the **Single Window Session** option on the **Sessions** pull down menu that is accessed from the **Options** button on the Login Manager's login interface. This will start up the Motif window manager and bring up a terminal window.
2. Browse the file:

   `$HOME/.dt/errorlog`

   Locate the configuration errors that caused the Session Manager to fail to start your session.
3. Correct the errors in your configuration file(s).
4. Exit the terminal window and the Login Manager's login interface will re-appear.
5. Try to log in again. If unsuccessful, repeat the above procedure until you are able to successfully log in.

The single window session can be customized by the system administrator to bring up more than one terminal window or other applications if desired. To customize the single window session:

1. Log in as root.
2. If the /etc/dt/config/Xfailsafe file does not exist, copy the /usr/dt/config/Xfailsafe file to /etc/dt/config/Xfailsafe.
3. Edit the /etc/dt/config/Xfailsafe file and insert any desired commands just before the last line in the file that brings up the terminal emulator. All commands that you insert must be run in the background (& parameter). You may notice that the last line in the file that brings up the terminal emulator is not run in the background. This is because when you exit the terminal emulator, the single window session is shut down.
4. Save the file.

Be sure and test the single window session option after you make any changes to it. If you do not, and inadvertently introduce an error in this configuration file,

you may not be able to use the single window option when need it. If this happens you can use the **Command Line Login** option on the Login Manager′s **Options** menu to still get into the system to fix your errors.

3.5 The Session Manager′s Role With Respect to X Resources

As mentioned in the beginning of this chapter, one of the functions of the Session Manager is to save and restore X resource settings for the user. The Session Manager maintains, in the respective session file, a complete dump of the X resources that were in effect when the user′s home or current sessions are saved. Later, when the user logs back in, the Session Manager reloads the X resource database from the X resource dump that was saved in the session being loaded. Then, the Session Manager merges the resources from your .Xdefaults file into the X resource database. Because your .Xdefaults file is merged last, the resources that you specify in your .Xdefaults file will override duplicate resources already in the database. This strategy works fine for adding and changing resources in the database. However, this strategy can cause a problem if you are trying to remove a resource from the resource database. For example, suppose you put the following resource in your .Xdefaults file to turn on the scroll bar facility of the aixterm application:

```
aixterm.scrollBar: true
```

The next time you log in, the Session Manager will read your .Xdefaults file and insert the scroll bar resource into the X resource database. As a result any aixterm that you now open will now have the scroll bar facility available. When you log out or save your home session, the Session Manager will save this resource in either the current or home session file. Later if you decide that you no longer want the scroll bar facility, you can not simply remove the resource from your .Xdefaults file, as you could in the past. This is because the Session Manager has saved this resource in its session files and will automatically reload it for you when it starts up. There are two options available to remove this unwanted resource:

- Use the **EditResources** action in the DesktopTools folder of the Application Manager to remove the resource from the X resource database. Then log out (if using the current session) or save your home session (if you are using the home session) to update the Session Manager′s version of the resource database in the respective session file.

  Note

  We have found that the **EditResources** action does not always work. You can get around this problem by creating the following shell script to perform the same function.

  ```
  #!/bin/ksh
  /usr/lpp/X11/bin/xrdb -q > /tmp/$$.xrdb
  ${EDITOR:-dtpad} /tmp/$$.xrdb
  /usr/lpp/X11/bin/xrdb -nocpp /tmp/$$.xrdb
  rm -rf /tmp/$$.xrdb
  ```

- You can in most cases alter the resource in your .Xdefaults file to specify a different value for the resource. In this example, you could specify a value of false to turn off the scroll bar facility. For example,

  ```
  aixterm.scrollBar: false
  ```

Now log out (if using the current session) or save your home session (if you are using the home session) to update the Session Manager's version of the resource database in the respective session file.

Even though there exists this minor problem with removing resources, it is still recommended to maintain your X resources in the .Xdefaults. This is largely due to the fact that the next version of the desktop will overcome this resource removal limitation and fully support the .Xdefaults file.

Chapter 4. Front Panel

The Front Panel is the main desktop user interface. It is typically located at the bottom of the screen on all workspaces and provides a central location for accessing the desktop facilities, frequently used applications, devices and information. The default Front Panel pictured in Figure 4 includes from left to right:

- A clock
- Access to the desktop calendar program
- Access to the File Manager
- Access to Personal applications
- Access to the desktop mail program
- A display lock
- Access to the various workspaces
- A busy indicator
- An exit button
- Access to the printer queues and a printer information application
- Access to the Style Manager
- Access to the Application Manager
- Access to the Help Manager
- A trash can

The Front Panel can be customized by the user to alter the available facilities and/or the physical arrangement of the panel. The following sections will provide you with some background on the inter-workings of the front panel as well as information on how to customize an existing front panel and information on how to create a new front panel.

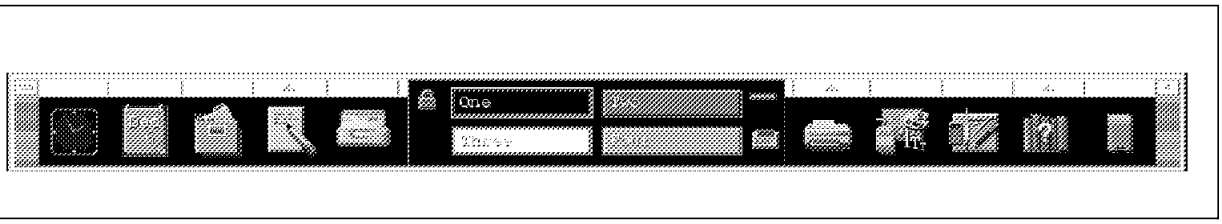


Figure 4. Default Front Panel

4.1 Inter-Workings of the Front Panel

The front panel is made up of control and container components that define the visual layout and functionality of the front panel. This section will describe:

- The components that make up the front panel
- How the front panel is constructed
- The component definition syntax that is used to define the front panel

4.1.1 Front Panel Components

The Front Panel interface is made up of seven different types of control components and four types of container components that group the controls. The available control components are:

- Icon - The icon control is the most common type of control used in the front panel. This type of control is actually a desktop object imbedded into the front panel. See 7.1, “Object” on page 110 for an explanation of desktop objects. Each icon control is represented by the object′s icon in the front panel. When an icon control is activated, through either a single mouse click on the icon or another object is dropped on to it, the underlying object′s action that corresponds with the activation method used (mouse click or dropped object) will be invoked. For example, the printer icon in the front panel is part of the printer object and has both the single mouse click and dropped object behaviors defined in the object′s definition. If you activate the printer object with a single mouse click on the icon in the front panel, you will bring up the desktop printers application. This object also allows you to print files by dropping a file′s object directly on to the printer icon in the front panel. The icon of an object used in the front panel can either be static or animated, which changes and animates when you select it. The icon control also has facilities that extend the functionality of an object. These include the capability to have the object′s icon change when an identified file is created or when mail arrives. Additionally, the icon control can be set up to allow only one instance of the object′s action to be active at any given time.
- Blank - This type of control is used exclusively for spacing of the various controls in the front panel layout.
- Busy - This control blinks to indicate that the front panel is actively processing your selection or drop request. This control only blinks when the front panel is processing a request. After the front panel finishes the request the indicator will stop blinking. This can be confusing, because this indicator is not a system busy indicator or a disk access light. If the control that you select launches an application, the indicator will probably stop blinking long before the application has finished initializing. Again, this is because the blinking stops when the desktop finishes launching the application not when the application finishes initializing or is running.
- Client - This type of control allows you to display the graphics of an X-Windows program directly in the front panel. Obviously the X-Windows client should produce graphic output that is appropriately sized to be displayed in the front panel.
- Clock - This control is the clock in the default front panel.
- Date - This control is the calendar in the default front panel. When selected, this control will bring up the desktop calendar program.
- File - This type of control is used to represent a data file in the front panel. When a file control is selected, the action associated with the control is executed and is passed the file name of the file that is represented by the control. For example, you can configure the front panel to have a file control that represents an inventory database that will automatically be loaded into your database package when you select the control.

There are four types of container components that are used to hold these controls in the front panel:

- The Main Panel - The main panel, shown in Figure 5 on page 33, is the outermost container of the front panel. It holds the other containers and has the following special controls of its own:
  - Menu Button - The menu button displays the front panel system menu. This menu contains buttons to restore, move, minimize, lower and refresh the front panel. Additionally there is a button that will log you off of the desktop.
  - Iconify Button - The iconify button, when pressed, iconifies the front panel. When the front panel is iconified and the icon box is not being used, the front panel icon will be located in the lower left corner of the display.
  - Position Handle - The position handle is used to reposition the front panel. To use the handle, select the handle area by pressing the left mouse button and hold it down. While holding the button down drag the front panel to the desired location. The handle can also be used to access the front panel's system menu by pressing the right mouse button while in the handle area.

  There can only be one main panel container in the front panel.

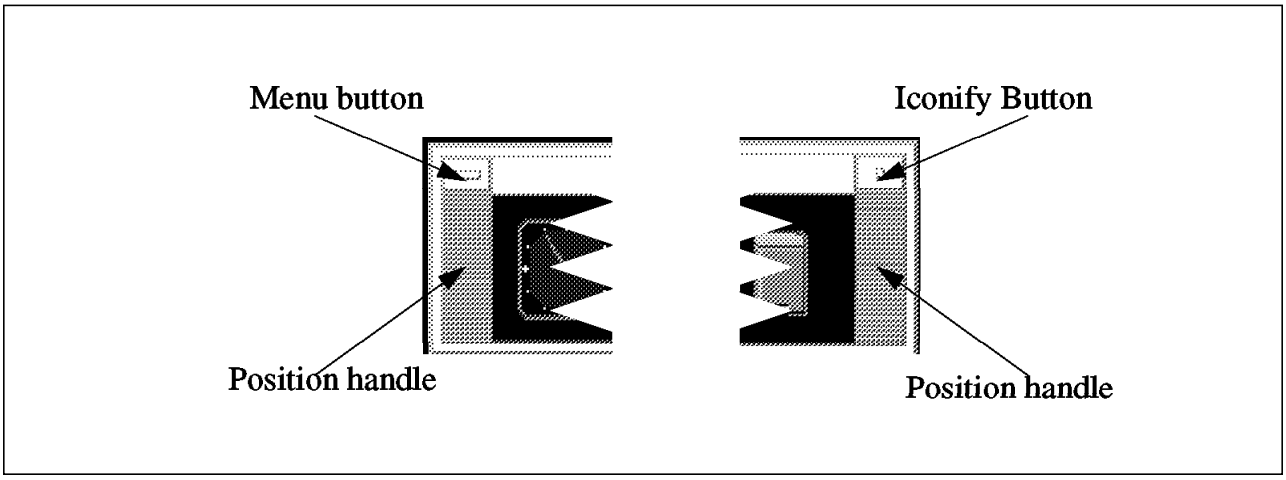


Figure 5. Main Panel Container

- Boxes - Boxes are containers that reside in the main panel container and actually hold the controls that make up the front panel. The default front panel contains one box that holds all (except for the above mentioned controls in the main panel container) of the controls in the front panel. Boxes lay out their contents in a horizontal fashion. If more than one box is used in the front panel, the boxes are stacked on top of each other. Figure 6 shows the default front panel's only box and its contents.

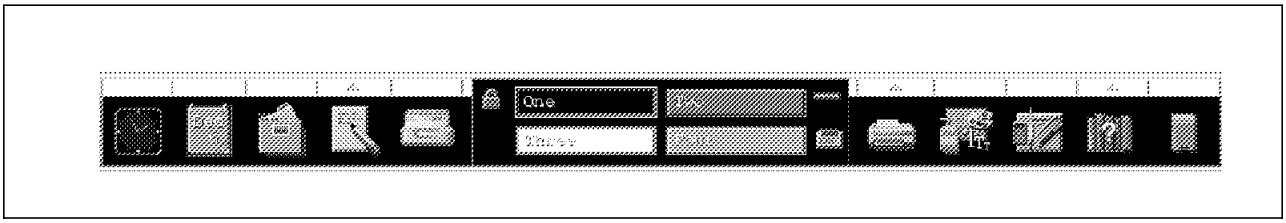


Figure 6. Default Front Panel Box Container and Contents

- The Switch - The switch, like the one pictured in Figure 7 on page 34, is a specialized container that can contain controls that are positioned around the workspace buttons in the center. The workspace buttons are special controls that, when selected, switch the user between their various workspaces. There can only be one switch container in the front panel.

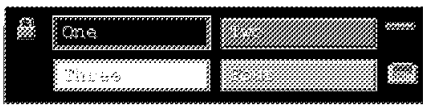


Figure 7. Default Front Panel Switch Container

- Subpanels - A subpanel, like the one shown in Figure 8 on page 35, is a slide up container which can be attached to any of the controls in the front panel. If a front panel control has a subpanel attached to it, an upwards pointing triangle will appear in the rectangle just above the control. When this rectangle is selected with a mouse click, the associated subpanel will slide up revealing its contents. The subpanel container is generally used to hold controls that you want to have easily available but are not needed to be displayed all the time. In the default front panel the help control is an example of a control that has a subpanel attached to it. Subpanels can be torn off of the front panel by clicking and holding the left mouse button down while the cursor is in the open subpanel's title area. The subpanel can then be dragged to the desired location and dropped by releasing the mouse button. When a subpanel is closed it is automatically reattached to the front panel. Subpanels can be closed by:
  - Selecting the **Close** option on the subpanel window menu.
  - Selecting one of the controls on the subpanel. When one of the controls on the subpanel is selected, the action associated to the control is invoked, and the subpanel is automatically closed by the desktop. This is a default behavior of subpanels and can be changed in a front panel definition. For more information see the SUBPANEL_UNPOST keyword of the PANEL component described in 4.1.3, "Front Panel Component Definition Syntax" on page 35.
  - Selecting the triangle you used to slide up the subpanel. You may notice that the triangle is pointing downwards when the subpanel is displayed.

Figure 8. Default Front Panel Help Subpanel Container

4.1.2 How the Front Panel is Constructed

During user log in and Workspace Manager restart, the desktop constructs the front panel from various front panel component definition statements (see 4.1.3, “Front Panel Component Definition Syntax”) that may exist an many different files located in several different directories. The DTDATABASESEARCHPATH variable, described in 8.3, “Data Types” on page 124, is used to locate the files that contain the component definition statements. The contents of any file that has a .fp suffix in the DTDATABASESEATCHPATH is used to construct the front panel definition. Essentially the search path gathers all the .fp files from the users: $HOME/.dt/types directory, the system administrator′s customization directory: /etc/dt/appconfig/types/$LANG and the system defaults directory: /usr/dt/appconfig/$LANG. The contents from all of these .fp files is merged into the front panel definition that define the front panel. The first component of each unique name that is found in the search path is used to define that given component. If other components with the same name and parent (CONTAINER_NAME and CONTAINER_TYPE) are found later in the search path they will be ignored. For example, suppose you define your own control component named: Mail with the parent container being Top. You save your component definition in a file called myfp.fp in your $HOME/.dt/types directory. Since your file will be found first in the search path, the front panel mail control will be built using your Mail component definition. The system default definition for the Mail component will be ignored.

4.1.3 Front Panel Component Definition Syntax

The desktop front panel is constructed from various front panel component definitions. These definitions define the way the front panel will look and operate. There are six different types of front panel component definitions. They correspond (with the exception of the ANIMATION component) to the containers and controls that make up the front panel components as described in 4.1.1, “Front Panel Components” on page 32. Figure 9 on page 36 shows the hierarchical relationship between the components.

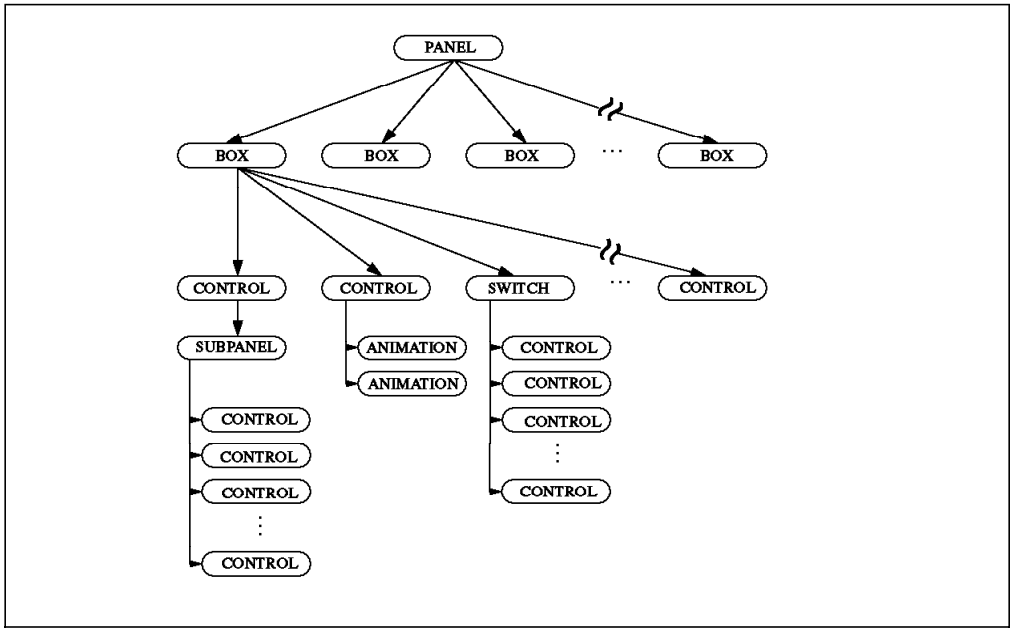


Figure 9. Front Panel Component Hierarchy

The front panel component definitions are:

1. Panel - This is the top level component that names and describes the appearance and behavior of the front panel itself. This component is the main panel container described in 4.1.1, “Front Panel Components” on page 32. There can only be one panel component in any given front panel. The panel component is defined using the following syntax:

```
PANEL Panel name
{
 KEYWORD value
  .
  .
  .
}
```

Where:

Panel name

The name of the panel being defined or modified. If you are creating a new front panel, this should be a unique name. If you are modifying (by overriding) an existing panel or the default front panel, this name should be the same as the panel being modified. The default front panel is called FrontPanel.

KEYWORD value

Identifies a component parameter and its value.

Table 3 (Page 1 of 2). Panel Component Keywords

KEYWORD	Value	Default	Description
DISPLAY_HANDLES	True or False	True	Use front panel positioning handles.
DISPLAY_MENU	True or False	True	Use front panel system menu.
DISPLAY_MINIMIZE	True or False	True	Use front panel minimize button.
RESOLUTION	high, medium, low or match_display	match_display	Which set of icons to use in the front panel.
CONTROL_BEHAVIOR	single_click or double_click	double_click	Number of mouse clicks needed to select a control.

Table 3 (Page 2 of 2). Panel Component Keywords

KEYWORD	Value	Default	Description
SUBPANEL_UNPOST	True or False	True	Automatically close subpanels when a subpanel control is selected.
DISPLAY_CONTROL_LABELS	True or False	False	Display control labels in the front panel.
LOCKED	True or False	False	Prevent a component definition of identical type, name and parent from overriding (replacing) this definition.
HELP_STRING	string		Help string to display when help is requested on a front panel component. Since each component type can have a help string associated with it, there is a precedence used in deciding which help string to display. Control help takes precedence over box help, switch help and subpanel help. Box help takes precedence over panel help. This keyword is only used if the HELP_TOPIC keyword has not been specified.
HELP_TOPIC	topic name	FPOnItemFrontPanel	Help topic to display from either the default help volume or the help volume specified by the HELP_VOLUME keyword. Like the HELP_STRING keyword, each component type can have a help topic associated with it and the same precedence rules apply.
HELP_VOLUME	topic name	FPanel	Help volume to use in conjunction with the HELP_TOPIC.

2. Box - This component defines a container for all of the remaining component types. In essence, it is a grouping mechanism that is used to establish the layout of the front panel. Each box is a single row container that is populated from left to right. Since a box is only one row deep, multiple row front panels are possible by stacking these boxes one on top of the other. The box component is defined using the following syntax:

```
BOX Box name
{
 KEYWORD   value
  .
  .
  .
}
```

Where:

Box name

The name of the box being defined or modified. If you are creating a new box, this should be a unique name. If you are modifying (by overriding) an existing box or the box in default front panel, this name should be the same as the box being modified. The only box in the default front panel is called Top.

KEYWORD value

Identifies a component parameter and its value.

Table 4. Box Component Keywords

KEYWORD	Value	Default	Description
CONTAINER_NAME	PANEL component name		Name specified on the PANEL component definition of the front panel that is to be the parent of this box.
POSITION_HINTS	First/Last/1-99	1	Specifies the vertical top to bottom ordering of the boxes in the front panel. When two boxes have the same value for POSITION_HINTS, the first one read from the configurations is placed first.
LOCKED	True or False	False	Prevents a component definition of identical type, name and parent from overriding (replacing) this definition.
DELETE	True or False	False	Used to override and remove a non-locked component from the front panel. This is necessary to eliminate system default front panel components without replacing the default files. To use DELETE, just a copy of the component definition to be deleted with the additional DELETE keyword is stored in a front panel definition file (file with a .fp extension) located in the user's $HOME/.dt/types directory.
HELP_STRING	string		Help string to display when help is requested on a front panel component. Since each component type can have a help string associated with it, there is a precedence used in deciding which help string to display. Control help takes precedence over box help, switch help and subpanel help. Box help takes precedence over panel help. This keyword is only used if the HELP_TOPIC keyword has not been specified.
HELP_TOPIC	topic name	FPOnItemFrontPanel	Help topic to display from either the default help volume or the help volume specified by the HELP_VOLUME keyword. Like the HELP_STRING keyword, each component type can have a help topic associated with it and the same precedence rules apply.
HELP_VOLUME	topic name	FPanel	Help volume to use in conjunction with the HELP_TOPIC.

3. Control - This component defines the functions and facilities that are available from the front panel. You might think of them as access points which the user can select to access the various desktop features, as well as their applications. For example, the default front panel contains an icon which when selected with a single mouse click, will bring up the mailer program. Similarly, there is another icon that brings up the File Manager facility. Each of the functions and facilities that can be accessed from the front panel are defined through one of these control definitions. There are actually seven different types of controls that can be defined. See 4.1.1, "Front Panel Components" on page 32 for a description of the seven different control types. The control component is defined using the following syntax:

```
CONTROL Control name
{
 KEYWORD   value
 .
 .
 .
}
```

Where:

Control name

The name of the control being defined or modified. If you are creating a new control, this should be a unique name. If you are modifying (by overriding) an existing control this name should be the same as the control being modified. The controls in the default front panel are named:

Clock
Clock in the front panel

Date
Calendar access point

Home
File Manager access point

TextEditor
Text editor access point

Mail
Mailer access point

Blank1
Blank space between the mail control and the workspace switch

Blank2
Blank space between the workspace switch and the printer control

Printer
Printer queues and application access point

Style
Style Manager access point

Applications
Application Manager access point

Help
Help Manager access point

Trash
Trash can in the front panel

Lock
Display lock access point in the workspace switch

Busy
Busy indicator in the workspace switch

Blank
Blank area in the lower left corner of the workspace switch

Exit

Exit button in the workspace switch

Term

Terminal window access point in the Personal Applications subpanel located above the TextEditor control

IconEditor

Icon editor access point in the Personal Applications subpanel located above the TextEditor control

HelpOverview

Help Overview access point in the Help subpanel located above the Help control

FPHelp

Front Panel help access point in the Help subpanel located above the Help control

OnItem

On item help access point in the Help subpanel located above the Help control

KEYWORD value

Identifies a component parameter and its value.

Table 5 (Page 1 of 3). Control Component Keywords

KEYWORD	Value	Default	Description
CONTAINER_NAME	BOX, SUBPANEL or SWITCH component name		Name specified on the BOX, SUBPANEL or SWITCH component definition that is to be the parent of this control.
CONTAINER_TYPE	(BOX, SUBPANEL or SWITCH)		Type of container that is to be the parent of this control.
POSITION_HINTS	First/Last/1-99	1	Specifies the ordering of controls in the parent container. A box is populated from left to right. The first control in the box is POSITION_HINT 1. A subpanel is populated from top to bottom. The top control (not including the install drop zone) in the subpanel is POSITION_HINT 1. The switch is populated from left to right and top to bottom. The control in the upper left hand corner of the switch is in POSITION_HINT 1. When two controls and/or the switch have the same value for POSITION_HINTS, the first one read from the configurations is placed first.
TYPE	blank, busy, client, clock, date, file or icon		Type of control be defined.
ICON	Icon file name		File name for the icon that is to be used to represent the control in the front panel. The icon must exist in the DTICONSEARCHPATH and must be specified as the base name only. The desktop will append the appropriate suffixes (.m, .l, .s, or .t & .pm or .bm) as needed.

Table 5 (Page 2 of 3). Control Component Keywords

KEYWORD	Value	Default	Description
ALTERNATE_ICON	Icon file name		File name for the icon that is to be used to replace the normal icon in a control that has specified a MONITOR_TYPE or in a control that is of type BUSY. In a control that has specified a MONITOR_TYPE, the alternate icon is displayed when the file being monitored changes or mail arrives. In the control that is of type busy, the alternate icon is used to create the blinking effect by toggling between the icon and the alternate icon. The icon must exist in the DTICONSEARCHPATH and must be specified as the base name only. The desktop will append the appropriate suffixes (.m, .l, .s, or .t & .pm or .bm) as needed.
PUSH_ACTION	Name of an Action		Name of the Action that is to be invoked when the control is selected though a mouse click.
PUSH_ANIMATION	Name of an ANIMATION component		Name of the ANIMATION component that is to be invoked when the control is selected through a mouse click.
DROP_ACTION	Name of an Action		Name of the Action that is to be invoked when the control is selected though an object drop.
DROP_ANIMATION	Name of an ANIMATION component		Name of the ANIMATION component that is to be invoked when the control is selected through an object drop.
PUSH_RECALL	True or False	True	Specifies that only one process can be started by the control. If the process is already running, it is displayed within the current workspace and shuffled to the top of the window stack. The value for the CLIENT_NAME keyword is used to identify the process associated with the PUSH_RECALL behavior.
CLIENT_NAME	X-Windows client name		Specifies a name used to associate a control with an executable. It is used when the control type is CLIENT (an X client running within the front panel) or ICON when the PUSH_RECALL keyword is True. The value for CLIENT_NAME is the first string (res_name) in the WM_CLASS property on the application's top level window.
CLIENT_GEOMETRY	widthxheight (in pixels)		Specifies the space (in pixels) to reserve in the front panel for controls of type client.
MONITOR_TYPE	mail/file		Specifies what is to be monitored. File is specified to monitor a file for changes and mail is specified to monitor for incoming mail.
FILE_NAME	Full path file name		Specifies the name of the file to monitor for changes when the MONITOR type is set to file.
LOCKED	True or False	False	Prevents a component definition of identical type, name and parent from overriding (replacing) this definition.
DELETE	True or False	False	Used to override and remove a non-locked component from the front panel. This is necessary to eliminate system default front panel components without replacing the default files.

Table 5 (Page 3 of 3). Control Component Keywords

KEYWORD	Value	Default	Description
HELP_STRING	string		Help string to display when help is requested on a front panel component. Since each component type can have a help string associated with it, there is a precedence used in deciding which help string to display. Control help takes precedence over box help, switch help and subpanel help. Box help takes precedence over panel help. This keyword is only used if the HELP_TOPIC keyword has not been specified.
HELP_TOPIC	topic name	FPOnItemFrontPanel	Help topic to display from either the default help volume or the help volume specified by the HELP_VOLUME keyword. Like the HELP_STRING keyword, each component type can have a help topic associated with it and the same precedence rules apply.
HELP_VOLUME	topic name	FPanel	Help volume to use in conjunction with the HELP_TOPIC.

4. Subpanel - This component defines the subpanel container. Like the box container, it is used to group its contents. However, unlike the box, a subpanel can only contain controls. Additionally, only one of these controls can be displayed in the front panel at any given time. The others in the subpanel are hidden on a slide up menu and only displayed when the user requests the menu. Also, unlike the box, the subpanel lays out its contents in a single vertical column format. Subpanels are generally used to group similar controls that need to be readily available, but are not used as often as the ones which are displayed in the front panel itself. The subpanel component is defined using the following syntax:

```
SUBPANEL Subpanel name
{
 KEYWORD   value
  .
  .
  .
}
```

Where:

Subpanel name

The name of the subpanel being defined or modified. If you are creating a new subpanel, this should be a unique name. If you are modifying (by overriding) an existing subpanel, this name should be the same as the subpanel being modified. The subpanels in the default front panel are named:

PersAppsSubpanel

Subpanel above the TextEditor control

PersPrintersSubpanel

Subpanel above the printer control

HelpSubpanel

Subpanel above the help control

KEYWORD value

Identifies a component parameter and its value.

Table 6. Subpanel Component Keywords

KEYWORD	Value	Default	Description
CONTAINER_NAME	Control component name		Name specified on the CONTROL component definition that is to be the parent of this subpanel.
TITLE	String		Title to be displayed on top of the subpanel.
CONTROL_INSTALL	True or False	True	Include the install drop zone at the top of the subpanel.
LOCKED	True or False	False	Prevents a component definition of identical type, name and parent from overriding (replacing) this definition.
DELETE	True or False	False	Used to override and remove a non-locked component from the front panel. This is necessary to eliminate system default front panel components without replacing the default files.
HELP_STRING	string		Help string to display when help is requested on a front panel component. Since each component type can have a help string associated with it, there is a precedence used in deciding which help string to display. Control help takes precedence over box help, switch help and subpanel help. Box help takes precedence over panel help. This keyword is only used if the HELP_TOPIC keyword has not been specified.
HELP_TOPIC	topic name	FPOnItemFrontPanel	Help topic to display from either the default help volume or the help volume specified by the HELP_VOLUME keyword. Like the HELP_STRING keyword, each component type can have a help topic associated with it and the same precedence rules apply.
HELP_VOLUME	topic name	FPanel	Help volume to use in conjunction with the HELP_TOPIC.

5. Switch - This component defines the switch container. It is used to hold a few controls and the workspace switch buttons, which allow the user to toggle between their workspaces. There can only be one switch defined in any given front panel. The switch component is defined using the following syntax:

```
Where:
SWITCH Switch name
{
 KEYWORD value
 .
 .
 .
}
```

Switch name

The name of the switch being defined or modified. If you are creating a new switch, this should be a unique name. If you are modifying (by overriding) an existing switch this name should be the same as the switch being modified. The only switch in the default front panel is named: Switch.

KEYWORD value

Identifies a component parameter and its value.

Table 7. Switch Component Keywords

KEYWORD	Value	Default	Description
CONTAINER_NAME	Box component name		Name specified on the box component definition that is to be the parent of this switch.
POSITION_HINTS	First/Last/1-99	1	Specifies the ordering of the controls and the switch in the box container. The switch is populated from left to right and top to bottom. The control in the upper left hand corner of the switch is in POSITION_HINT 1. When two controls and/or the switch have the same value for POSITION_HINTS, the first one read from the configurations is placed first.
NUMBER_OF_ROWS	1-N		The number of rows to be displayed in the switch. For each row there is an available control position at the beginning and at the end of the row. Therefore the number of available controls in the switch is 2 times the value of the NUMBER_OF_ROWS keyword.
LOCKED	True or False	False	Prevents a component definition of identical type, name and parent from overriding (replacing) this definition.
DELETE	True or False	False	Used to override and remove a non-locked component from the front panel. This is necessary to eliminate system default front panel components without replacing the default files.
HELP_STRING	string		Help string to display when help is requested on a front panel component. Since each component type can have a help string associated with it, there is a precedence used in deciding which help string to display. Control help takes precedence over box help, switch help and subpanel help. Box help takes precedence over panel help. This keyword is only used if the HELP_TOPIC keyword has not been specified.
HELP_TOPIC	topic name	FPOnItemFrontPanel	Help topic to display from either the default help volume or the help volume specified by the HELP_VOLUME keyword. Like the HELP_STRING keyword, each component type can have a help topic associated with it and the same precedence rules apply.
HELP_VOLUME	topic name	FPanel	Help volume to use in conjunction with the HELP_TOPIC.

6. Animation - This is a special component which is used to define the sequence of icons that are used to simulate animation when a control which has been defined with either the PUSH_ANIMATION or DROP_ANIMATION keywords. The animation component is defined using the following syntax:

```
ANIMATION Animation name
{
 ANIMATION icon name, [millisecond delay]
 .
 .
 .
}
```

Where:

Animation name
The name of the animation sequence being defined. This should be a unique name.

icon name
The name of an icon in the animation sequence. The icon must exist in the DTICONSEARCHPATH and must be specified as the base name only. The desktop will append the appropriate suffixes (.m, .l, .s, or .t & .pm or .bm) as needed.

millisecond delay
Optional delay (in milliseconds) to wait before displaying the icon specified on the next ANIMATION statement in the definition. The default time is 200 milliseconds.

4.2 Customizing an Existing Front Panel

There are two available methods for customizing an existing front panel:

- Interactive - This method uses the built in customization facilities of the desktop to interactively make changes to the front panel.
- Manual - This method customizes the front panel by either:
  - Altering or overriding the front panel component definitions that are used to build the front panel
  - Changing X Window resources that specify front panel parameters

This section will show you how to use these customization methods by explaining how to:

- Add and remove controls in a front panel
- Add and remove subpanels in a front panel
- Add and remove controls in a subpanel
- Add, remove and rename workspaces in a front panel
- Preventing users from changing controls and subpanels

4.2.1 Adding and Removing Controls in a Front Panel

Controls can be added and removed from a front panel by using the interactive customization method or the manual customization method. Although easy to use, the interactive customization method has the following limitations that do not exist when using the manual customization method:

- The physical size of the front panel cannot altered using this method. This means that you can add controls to the front panel as long as they replace existing controls. Similarly, you can delete controls from the front panel as long as they are replaced by another control.

- Only controls of type icon and file can be added to the front panel. Furthermore, only those icon and file controls created from objects existing in the File or Application Manager can be added to the front panel with interactive customization. For example, you can create an object, using the CreateAction facility, that calls an action to bring up your favorite application. Because this object can reside in the File Manager, you can interactively add it to the front panel. However, you cannot, for example, add client controls or icon controls that monitor a file to the front panel interactively. This is because these types of controls require facilities that are not part of an object.
- Controls in the workspace switch cannot be added or removed. interactively.
- Existing controls cannot be changed, they can only be replaced by different controls.

If the control that you wish to add falls under one of these limitations, you will need to use the manual customization method.

4.2.1.1 Adding Controls Using Interactive Customization

In order for a control to be interactively added to the front panel, the control must first exist as an object in either the File or Application Manager. Use the procedure described in 8.4, “Action and Data Type Database” on page 126 or 8.6, “Passing Arguments to Actions” on page 130 to create an object for the control that you want to add. When you are creating the icons for the object, create both the 48x48 and 32x32 pixel icons and place them in the desktop icon path. For information on object icons sizes and locations see Chapter 11, “Icons” on page 161. Once the object exists, the following procedure will add it to the front panel:

1. Use the procedure given in 4.2.3.1, “Adding Controls to a Subpanel Using Interactive Customization” on page 52 to copy the desired object from the File or the Application manager on to a subpanel that is above the control in the desired front panel position.
2. Move the mouse cursor over the icon in the subpanel that is to be copied into the front panel.
3. Click the right mouse button to reveal the control menu.
4. Select the **Copy to Main Panel** option from this menu. The result of this action is that the large version (48x48 pixels) of the control′s icon will replace the icon that was in the front panel. If the control does not have a 48x48 pixel icon, undesirable results may occur.

The control in the front panel is now ready for use.

For example, suppose that you want to add a control in the 4th position of the default front panel which brings up your host terminal emulator. First create an object that calls as its action your host terminal emulator program. Don′t forget to create both the 48x48 and the 32x32 pixel icons. Use the above procedure to add the object to the Personal Applications subpanel and then copy the host terminal emulator control down from the subpanel into the front panel.

4.2.1.2 Removing Controls Using Interactive Customization

A control that is in a front panel can only be removed interactively by copying another control from the subpanel above the control in place of the control to be removed. Use the procedure in 4.2.1.1, “Adding Controls Using Interactive Customization” on page 46 steps two, three and four 4 to replace the unwanted control with a different control. Then if desired, you can use the procedure in 4.2.3.2, “Removing Controls From Subpanels Using Interactive Customization” on page 52 to remove the unwanted control from the subpanel.

For example, suppose that you want replace the control that brings up the text editor with one that brings up an aixterm and vi. This can be done by simply adding a new control to the front panel using the procedure in 4.2.1.1, “Adding Controls Using Interactive Customization” on page 46. Then using the above procedure, remove the unwanted control that brings up the desktop text editor.

4.2.1.3 Manual Customization of Controls

Without creating a new front panel definition, you can override your existing front panel definition to add and/or remove controls from the front panel. Since this customization method alters the definition from which the front panel is constructed, you are not bound by the limitations of interactive customization.

The following example will illustrate how to customize the default front panel by overriding the system default definition to:

1. Replace the clock control with a control that brings up a host terminal emulator
2. Add a control to the end of the front panel which brings up FrameMaker
3. Delete the date control
4. Interchange the positions of the mail and trash controls

When these changes are complete the new front panel will look like the one shown in Figure 10

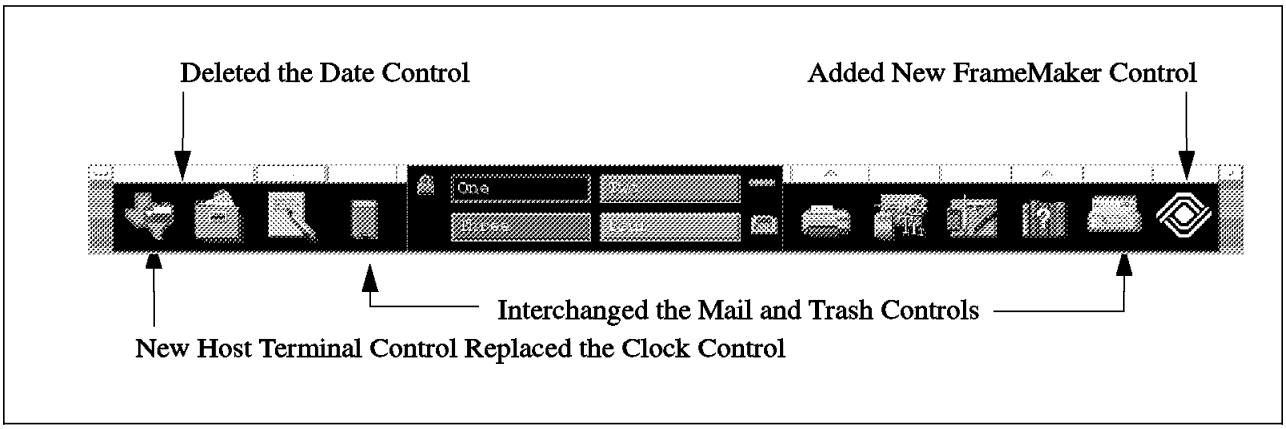


Figure 10. Front Panel Created by Overriding the Default Front Panel

Replacing a Control in the Front Panel: To replace a control in the front panel simply create a new control component definition for the new control. The component name and the following keywords should be the same as the control being replaced:

```
POSITION_HINTS
CONTAINER_NAME
```

```
CONTAINER_TYPE
```

We used the following procedure for our example:

1. Create an icon for the new control.

   We went ahead and created both the 32x32 and 48x48 pixel icons. Even though we are initially placing the control in the front panel, it later might be moved up into a subpanel, so we also created the 32x32 size. For more information on icons see 11.1, “Creating Your Own Icons” on page 161. Both of our icons were placed in the directory: $HOME/.dt/icons.

2. Create a new action using the CreateAction tool described in 8.4, “Action and Data Type Database” on page 126.

   We called it AUSVMR. It brings up our host terminal emulator program when it is selected.

3. Create the new control component definition.

   We placed it in the file $HOME/.dt/types/OverRide.fp and it contained the following:

```
CONTROL Clock          <=== The same name as the Clock control
{
 TYPE           icon   <=== Type of our new control
 CONTAINER_NAME Top    <=== Name of the container holding the Clock
 CONTAINER_TYPE BOX    <=== Type of the container holding the Clock
 POSITION_HINTS 1      <=== The position of the Clock control
 ICON           Texas  <=== Our new icon from $HOME/.dt/icons
 PUSH_ACTION    AUSVMR <=== The action we created with CreateAction
}
```

4. Restart the Workspace Manager.

Adding a Control in the Front Panel: To add a control in the front panel simply create a new control component definition for the new control and parent it to the existing front panel. We used the following procedure for our example:

1. Create or locate an icon that will be used for the new control. Make sure that you have both the 32x32 and 48x48 pixel icons. Even though we are initially placing the control in the front panel, it later might be moved up into a subpanel. That it why we also need the 32x32 size. For more information on icons see 11.1, “Creating Your Own Icons” on page 161.

   Both of our frame icons were placed in the directory: $HOME/.dt/icons.

2. Create a new action using the CreateAction tool as described in 8.4, “Action and Data Type Database” on page 126 or select an existing action.

   For this example we used the Maker action that was created in 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132.

3. Create the new control component definition.

   We placed it in the file $HOME/.dt/types/OverRide.fp and it contained the following:

```
CONTROL Maker          <=== The name for our new control
{
 TYPE           icon   <=== Type of our new control
 CONTAINER_NAME Top    <=== Name of the default front panel container
 CONTAINER_TYPE BOX    <=== Type of the default front panel container
 POSITION_HINTS LAST   <=== The position of the Maker control
 ICON           frame  <=== Our new icon from $HOME/.dt/icons
```

```
 PUSH_ACTION    Maker  <=== The action we created or selected to use
}
```

4. Restart the Workspace Manager.

Deleting a Control in the Front Panel: To delete a control in the Front Panel simply create a new control component definition. The component name and the following keywords should be the same as the control being deleted:

```
CONTAINER_NAME

CONTAINER_TYPE
```

Use the DELETE keyword in the new definition to remove the unwanted control. We used the following procedure for our example:

1. Create the new control component definition.

   We placed it in the file: $HOME/.dt/types/OverRide.fp and it contained:

```
CONTROL Date           <=== The same name as the Date control
{
 CONTAINER_NAME Top    <=== Name of the container holding the Date
 CONTAINER_TYPE BOX    <=== Type of the container holding the Date
 DELETE         True   <=== Delete this control
}
```

2. Restart the Workspace Manager.

Interchanging Control Positions: To interchange the positions of two or more controls in a front panel, copy their definitions and interchange the POSITION_HINTS keywords. We used the following procedure for our example:

1. Copy the component definitions for the controls to be interchanged into your $HOME/.dt/types directory.

   We copied the control component definitions for the Mail and Trash controls and placed them in the file: $HOME/.dt/types/OverRide.fp.

2. Swap the POSITION_HINTS lines among the controls to reflect the desired new positions.

   We specified the POSITION_HINTS lines to reflect the new desired positions as shown below:

```
CONTROL Mail
{
 TYPE           icon
 CONTAINER_NAME Top
 CONTAINER_TYPE BOX
 POSITION_HINTS 13    <=== New position
 ICON           DtMail
 LABEL          Mail
 ALTERNATE_ICON DtMnew
 MONITOR_TYPE   mail
 PUSH_ACTION    Mailer
 PUSH_RECALL    True
 CLIENT_NAME    Mail
 HELP_TOPIC     FPOmItemMail
 HELP_VOLUME    FPanel
}
CONTROL Trash
{
 TYPE           icon
```

```
 CONTAINER_NAME Top
 CONTAINER_TYPE BOX
 POSITION_HINTS 5   <=== New position
 ICON           Fptrsh
 LABEL          Trash
 ALTERNATE_ICON Fptrsh7
 FILE_NAME      $HOME/.dt/Trash/.trashinfo
 MONITOR_TYPE   file
 PUSH_ACTION    OpenTrash
 DROP_ACTION    TrashFile
 DROP_ANIMATION TrashDrop
 HELP_TOPIC     FPOmItemTrash
 HELP_VOLUME    FPanel
}
```

3. Save the file.
4. Restart the Workspace Manager.

4.2.2 Adding and Removing Subpanels from a Front Panel Control

Each control in the front panel, with the exception of the controls on the main panel (menu button, iconify button and positioning handles), the controls in the workspace switch (display lock, exit button, busy indicator and workspace buttons), and blank controls can have a subpanel attached to it. An easy way to determine if a control is capable of having a subpanel is to see if there is a subpanel access area just above the front panel control. Figure 11 shows a few controls from the default front panel with their subpanel access areas.

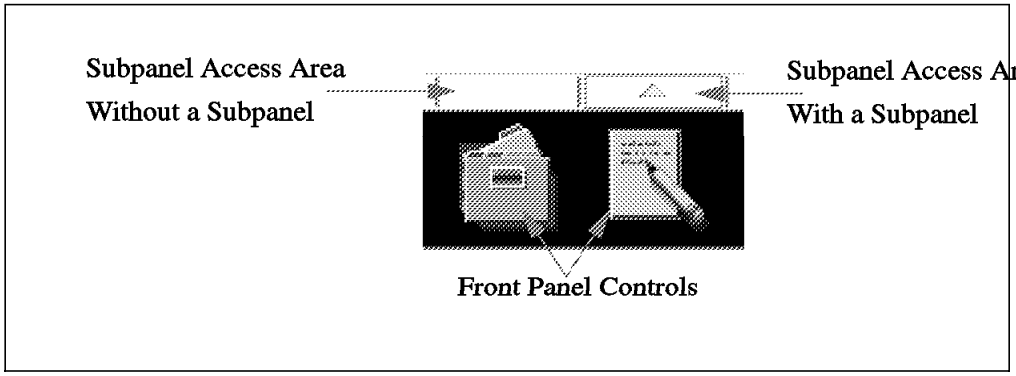


Figure 11. Front Panel Controls and Subpanel Access Areas

Subpanels are easily added and removed using interactive customization. The only limitation with interactively adding subpanels is that you are unable to turn off (remove from the subpanel) the Install Icon area in the subpanel. If this is area is not needed or wanted, you will have to add the subpanel using manual customization.

4.2.2.1 Adding Subpanels Using Interactive Customization

To interactively add a subpanel to a front panel control:

1. Move the mouse cursor over the front panel icon that is to have the new subpanel.
2. Click the right mouse button to reveal the control menu.
3. Select the **Add Subpanel** option from this menu. This option will not be available if the control already has a subpanel.

4. The subpanel access area above the control will change. You will now find the upwards pointing triangle in the area which indicates that a subpanel is now available for the control.

4.2.2.2 Removing Subpanels Using Interactive Customization

Subpanels can easily be removed with the following procedure:

1. If the subpanel to be removed still contains controls, it is recommended that they be removed from the subpanel before the subpanel itself is removed. This can be done by repeating the procedure given in 4.2.1.2, “Removing Controls Using Interactive Customization” on page 47 until the subpanel is empty. When the subpanel is empty (except for the Install Icon control), proceed to the next step to delete the subpanel.
2. Move the mouse cursor over the front panel icon that is to have its subpanel deleted.
3. Click the right mouse button to reveal the control menu.
4. Select the **Delete Subpanel** option from this menu.
5. A deletion confirmation dialog box will pop up. If the subpanel referred to in the dialog box is the intended subpanel to be removed, press the **OK** button. Otherwise, press the **Cancel** button to cancel removing the subpanel.
6. The subpanel access area above the control will change. The upwards pointing triangle in the area will have been removed indicating that this control no longer has a subpanel.

4.2.2.3 Manual Customization of Subpanels

As mentioned earlier, if you do not want the Install Icon area to be present on a subpanel, you need to use the manual customization method. This example will illustrate how to override the default front panel definition to remove the Install Icon area from the subpanel above the help control.

1. Copy the component definition for the help subpanel from the default front panel and place it in the file: $HOME/.dt/types/OverRide.fp
2. Add the following line to the definition:

   ```
   CONTROL_INSTALL False
   ```

   The new definition looks like this:

   ```
   SUBPANEL HelpSubpanel
   {
     CONTAINER_NAME        Help
     TITLE                 Help
     CONTROL_INSTALL       False
   }
   ```

3. Save the file.
4. Restart the Workspace Manager.

4.2.3 Adding and Removing Controls From a Subpanel

Like controls in the front panel, controls can be added and removed using either the interactive or manual customization methods. There is a limitation that goes along with interactively adding controls to a subpanel that does not exist if you use the manual customization method:

- Only objects that exist in either the File or Application Managers can be copied onto a subpanel to create a new control. For example, you can create an object, using the CreateAction facility, that calls an action to bring up your favorite application. Because this object can reside in the File Manager, you can interactively add it to the front panel.

4.2.3.1 Adding Controls to a Subpanel Using Interactive Customization

Any object that is displayable in either the File or Application Manager can be copied into one of the subpanels attached to one of the front panel controls. For example, suppose that you used the procedure given in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128 to create an object that brings up a host terminal emulator. You would now like this new object to be available on one of the subpanels above a control in the front panel. You can use the following procedure to accomplish this:

1. In order for an object to be displayed in a subpanel, you need to be sure that you have created and appropriately stored at least the 32x32 size icon for your new object. For information on object icons sizes and locations see Chapter 11, “Icons” on page 161.
2. Open the File Manager or the Application Manager to the location where your new object exists. For this example you will probably find the host terminal emulator in your home directory.
3. Open the subpanel which is to hold the new object. If the desired subpanel does not yet exist, use the procedure in 4.2.2.1, “Adding Subpanels Using Interactive Customization” on page 50 to add the new subpanel. For this example we want to have our new host terminal emulator object be on the existing Personal Applications subpanel which is located above the 4th control from the left on the default front panel.
4. Move the mouse cursor over the object to be copied to the subpanel in the File Manager or Application Manager.
5. Click and hold down the left mouse button.
6. Drag the object’s icon from the File or Application Manager over to the **Install Icon** entry in the open subpanel.
7. Release the left mouse button. The result of this action will add the icon and title for the object to the subpanel.

The new host terminal emulator object (which is now called a control because it is part of the front panel) is now ready for use on the subpanel.

4.2.3.2 Removing Controls From Subpanels Using Interactive Customization

All controls except the Install Icon area on a subpanel can easily be removed from the subpanel by using the following procedure:

1. Locate the control in the subpanel that is to be removed.
2. Click the right mouse button to reveal the control menu.
3. Select the **Delete** option from this menu. The result of this action is that a deletion conformation dialog box will pop up.
4. If the control referred to in the dialog box is the control to be removed, press the **OK** button. Otherwise press the **Cancel** button. If **OK** was selected, the

control will be removed from the subpanel. Otherwise the remove request will be ignored.

4.2.3.3 Manual Customization of Subpanel Controls

Without creating a new front panel definition, you can override the existing front panel definition that creates your front panel to add and/or remove controls from subpanels. Since this customization method alters the definition from which the front panel is constructed, you are not bound by the limitations of interactive customization.

The following example will illustrate how to customize the default front panel by overriding the system default definition to add a control to the Personal Applications subpanel that will monitor for the creation of a particular file to indicate when it is created. For our example we used the following procedure:

1. Create both an icon and an alternate icon for the new control.

   We went ahead and created both the 32x32 and 48x48 pixel icons. Even though we are initially placing the control in the subpanel, it later might be moved down into the front panel, so we also created the 48x48 size. For more information on icons see 11.1, “Creating Your Own Icons” on page 161. Both of our icons were placed in the directory: $HOME/.dt/icons.

2. Create the new control component definition.

   We placed it in the file $HOME/.dt/types/OverRide.fp and it contained:

```
CONTROL MyFileMonitor
{
 TYPE           icon             <=== Type of our new control
 CONTAINER_NAME PersAppsSubpanel <=== Name of the subpanel
 CONTAINER_TYPE SUBPANEL         <=== Type of the container holding the control
 POSITION_HINTS Last             <=== The position of the new control
 ICON           NoChange         <=== Our normal icon in $HOME/.dt/icons
 ALTERNATE_ICON Changed          <=== Our alternate icon in $HOME/.dt/icons
 MONITOR_TYPE   file             <=== We want to monitor a file for creation
 FILE_NAME      /tmp/watchfile   <=== File that we want know when it is created
}
```

3. Restart the Workspace Manager.

4.2.4 Adding, Removing and Renaming Workspaces

Adding, removing and renaming a workspace can easily be done interactively by a user for their own front panel. On a system wide basis, the system administrator can only change the initial number and names of workspaces that are given in the default front panel for new users only. This is because the first time a user logs out, the Session Manager saves information on the number and the names of workspaces a user had at logout. This information will override any system administrator provided settings when the user logs back in. This is necessary in order to restore the desktop back the way it was when the user logged out.

4.2.4.1 Interactively Adding Workspaces for Existing Users

Workspaces can easily be added by existing users with the following procedure:

1. Log in to the desktop as the user who needs the new workspace.
2. Move the mouse cursor over any one of the workspace buttons in the front panel.

3. Click the right mouse button to reveal the workspace button menu.
4. Select the **Add Workspace** option from this menu.
5. A new workspace button will be added into the switch area of the front panel. This new workspace will initially be named New.
6. Use the procedure in 4.2.4.3, “Interactively Renaming a Workspace” to rename the workspace as desired.

4.2.4.2 Interactively Removing Workspaces for Existing Users

Workspaces can easily be removed by existing users with the following procedure:

1. Log in to the desktop as the user who needs a workspace removed.
2. Move the mouse cursor over the workspace buttons in the front panel to be removed.
3. Click the right mouse button to reveal the workspace button menu.
4. Select the **Delete** option from this menu. The workspace will be removed and the front panel will be automatically reconfigured to remove the blank space (if any) left by the removed workspace.

Note: If the removed workspace contained windows for active client applications, the application′s windows will automatically moved to another workspace. If the removed workspace was the active workspace, the application′s windows will be moved to the first workspace. If the removed workspace was an inactive workspace, the application′s windows will be moved to the active workspace.

4.2.4.3 Interactively Renaming a Workspace

Each workspace has a unique name associated with it and is used to help identify your various workspaces. The names of your workspaces are found on the workspace buttons located in the switch container of the front panel. Figure 12 shows the workspace switch and the workspace buttons from the default front panel.

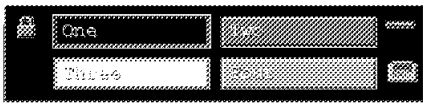


Figure 12. Default Front Panel Workspace Switch and Workspace Buttons

The name of any workspace can be easily changed. Each workspace name must conform to the following two rules.

1. The name can contain any printable character.
2. The name must be unique. No two workspaces can have the same name. By default, the workspace buttons in the front panel are limited to displaying 12 characters in the default font. Although the name of a workspace can be longer than 12 characters, only the first 12 characters of the name will be visible on the button. Unique names that are longer than 12 characters but have the first 12 characters in common, may appear to be the same on the workspace buttons.

There are two different methods to interactively rename workspaces:

1. Changing the name of a user's active workspace

   The following procedure allows a user to change the name of their workspace that is currently being displayed by the desktop:

   a. Move the mouse cursor over the workspace button in the front panel that is to be renamed.

   b. Single click the left mouse button. This action will give you a blinking cursor in the workspace button. You can use this cursor to edit the name of the workspace.

   c. When you have finished editing the name, press the keyboard Enter key to activate the change.

2. Changing the name of a user's inactive workspace

   The following procedure allows a user to change the name of a workspace that not currently being displayed by the desktop:

   a. Move the mouse cursor over the workspace button to be changed.

   b. Click the right mouse button to reveal the workspace button menu.

   c. Select the **Rename** option from this menu.

   d. You will be given a blinking cursor in the workspace button. You can use this cursor to edit the workspace name.

   e. When you have finished editing the name, press the keyboard Enter key to activate the new name.

4.2.4.4 Manually Changing the Default Number of Workspaces Given to New Users

The number of workspaces that are initially given to a new desktop user can be changed using the following procedure, which modifies some of the workspace X Window resources:

1. Log in as the system administrator.
2. Edit the /etc/dt/app-defaults/$LANG/Dtwm file. If this file does not exist copy the system default /usr/dt/app-defaults/$LANG/Dtwm file to /etc/dt/app-defaults/$LANG/Dtwm.
3. Locate and change the workspace count line. The default line will look like:

   ```
   Dtwm*workspaceCount:    4
   ```

   The value of this resource line (4) is the number of workspaces that are given to a new desktop user. Change this value to the desired number of workspaces. For example, to have each new user initially be given six workspaces when they first log in, change the workspace count line to look like the following:

   ```
   Dtwm*workspaceCount:    6
   ```

4. Set up the workspace title resource entries in the same file to match the new number of workspaces. See 4.2.4.5, "Manually Changing the Default Workspace Names" on page 56 for information on changing the workspace names for new users. Please make sure that the number of workspace title resource entries equals the new number of workspaces.
5. Save the changes you make to this file.

The new number of workspaces and their names will be used the next time a new user logs in to the desktop.

Note: This is for new users only. Once a user has logged into the desktop the default number and names of the workspaces that were active during the initial login will be saved in the users session files. That is unless the user added/removed workspaces and/or renamed their workspaces after login. In this case, the new number of workspaces and whatever names the user gave the workspaces will be saved in the user′s session files.

4.2.4.5 Manually Changing the Default Workspace Names

This procedure will edit the desktop resources that define the names given to the default workspaces for a new user′s desktop.

1. Log in as the system administrator.
2. Edit the /etc/dt/app-defaults/$LANG/Dtwm file. If this file does not exist copy the system default /usr/dt/app-defaults/$LANG/Dtwm file to /etc/dt/app-defaults/$LANG/Dtwm.
3. Locate the workspace title lines. The default lines will look like:

   ```
   Dtwm*ws0*title: One
   Dtwm*ws1*title: Two
   Dtwm*ws2*title: Three
   Dtwm*ws3*title: Four
   ```

   The value (One, Two, Three or Four) for each of the title resource lines is the title for the corresponding workspace. You can change these resource values to be the names of the new default workspaces. For example, changing the title lines to look like the following will produce workspaces named: Office, AIX, Inventory and News.

   ```
   Dtwm*ws0*title: Office
   Dtwm*ws1*title: AIX
   Dtwm*ws2*title: Inventory
   Dtwm*ws3*title: News
   ```

4. Save the changes you make to this file.

The new workspace names will be used the next time a new user logs in to the desktop.

Note: This is for new users only. Once a user has logged into the desktop the default workspace names that were active during their initial login will be saved in their session files. That is unless the user renamed their workspaces after login. In this case whatever name the user gave the workspaces will be saved in the user′s session files.

4.2.5 Preventing Changes to a Control and or a Subpanel

There may be circumstances under which you may want to prevent a desktop user from changing controls and/or subpanels in their front panel. To facilitate this need, the desktop provides a customization keyword that can be included in the front panel component definition. This keyword prevents a user from changing (by overriding it with a new definition) a specified control or subpanel. This parameter can be specified on a system wide basis for the default front panel, system administrator provided front panels, or on a per user basis for user created front panels.

Note: This feature of the desktop is not a completely secure mechanism. An experienced user can easily bypass this facility to re-enable the capability to

change protected controls and subpanels. However, they can only do this on a user by user basis. They cannot globally affect all users.

The following procedure for implementing this locking mechanism is the same regardless of whether you are trying to protect the default front panel or one of your customized front panels which you have previously created using the procedure given in 4.3, “Creating a New Front Panel” on page 58

1. Log in or su with a userID that has permission to modify the front panel definition files used to construct the front panel that you are trying to protect. If you are trying to protect the default front panel, log in or su with the root or system administrator userID.
2. Locate and edit the front panel definition file that contains the component(s) that you are trying to protect. If you are trying to protect the component(s) in the default front panel, edit the /etc/dt/appconfig/types/$LANG/dtwm.fp file. If this file does not exist copy the system default version from /usr/dt/appconfig/types/$LANG/dtwm.fp to /etc/dt/appconfig/types/$LANG/dtwm.fp.
3. In the definition file you will find component definitions for controls and subpanels that look like the following for the clock control and the Personal Applications subpanel from the default front panel:

```
CONTROL Clock
{
  TYPE                  clock
  CONTAINER_NAME        Top
  CONTAINER_TYPE        BOX
  POSITION_HINTS        1
  ICON                  Fpclock
  LABEL                 Clock
  HELP_TOPIC            FPOnItemClock
  HELP_VOLUME           FPanel
}

SUBPANEL PersAppsSubpanel
{
  CONTAINER_NAME        TextEditor
  TITLE                 Personal Applications
}
```

4. The locking mechanism is enabled by inserting the following line into each control and/or subpanel component definition that you would like to protect:

```
  LOCKED                True
```

   For example to protect the user from deleting the clock, the modified control component definition would look like the following:

```
CONTROL Clock
{
  TYPE                  clock
  CONTAINER_NAME        Top
  CONTAINER_TYPE        BOX
  POSITION_HINTS        1
  ICON                  Fpclock
  LABEL                 Clock
  HELP_TOPIC            FPOnItemClock
  HELP_VOLUME           FPanel
  LOCKED                True
}
```

Remember that you need to insert the LOCKED: keyword into each control and or subpanel component definition that you want to protect. It is not sufficient to insert the LOCKED keyword in a subpanel in order to protect all the controls in the subpanel. Each control in the subpanel must be individually protected.

5. Save the file.

The above changes will be in effect the next time the modified front panel is loaded at login or after restarting the Workspace Manager.

4.3 Creating a New Front Panel

The simplest way to create a new front panel is to copy an existing front panel definition and modify it as desired for your new panel. The following steps will show you how to create a new front panel from an existing front panel.

1. Locate a front panel definition to use as a model. Try to select a front panel definition for your model that is close to the ideas that you have for your new front panel. You can follow the path defined by the DTDATABASESEARCHPATH environment variable to search through the available panel definitions. You might also consider using the system default front panel in /usr/dt/appconfig/types/C/dtwm.fp for your model.
2. Copy your selected model definition into your $HOME/.dt/types directory giving it any file name you desire, as long as it has the .fp suffix. For example, MyPanel.fp or TheNewPanel.fp are both valid file names for front panel definitions.
3. Edit your model definition. Starting with the PANEL component. Change the name on the PANEL statement to a unique name for your new panel. Also, add, delete or change any of the PANEL keyword statements as desired for your new panel. For example, the following PANEL component definition was created from the system default front panel by renaming it from FrontPanel to MyOwnFrontPanel. We also changed the CONTROL_BEHAVIOR to double-click.

```
PANEL MyOwnFrontPanel
{
  DISPLAY_HANDLES       True
  DISPLAY_MENU          True
  DISPLAY_MINIMIZE      True
  CONTROL_BEHAVIOR      double_click
  HELP_TOPIC            FPOnItemFrontPanel
  HELP_VOLUME           FPanel
}
```

Now work your way down the front panel component hierarchy, shown in Figure 9 on page 36, adding, deleting or changing the component definitions in your model as desired. Refer to the following sections for information on working with the various component types:

4.3.1, “Creating Boxes” on page 60

4.3.2, “Creating Controls” on page 60

4.3.3, “Creating Subpanels” on page 64

4.3.4, “Creating the Switch” on page 65

4.3.5, “Creating an Animated Control” on page 66

When doing these changes remember the following:

- Each container has a name. All components from all definition files that identify a particular container name as its parent will be loaded into that container. This can sometimes cause problems if you do not rename your containers when modifying another front panel definition to make your own front panel. For example, suppose you use the system default front panel as your model and modify it to create a two row front panel by adding another box called Bottom. Now suppose that you move the clock component definition from the Top box to the Bottom box. If you do not change the name of the original box from Top to a unique name, when you restart the Workspace Manager to build your new panel, you will find the clock in both your Top and Bottom boxes. This is because the system default front panel is still in the search path. Therefore, the clock definition (whose parent is Top) is added into your container called Top. Also, since your clock definition has a different parent (Bottom) than the definition found in the default front panel (Top), the desktop considers them to be two different controls and puts the clock in your Bottom box as you requested.
- Make sure that the CONTAINER_NAME and CONTAINER_TYPE keyword statements in all your components correctly identify each component's parent. For any given component (except the ANIMATION components) you should be able to trace the CONTAINER_NAME and CONTAINER_TYPE statements all the way up to the PANEL definition. See Figure 13.

4. Save the new front panel file.
5. Restart the Workspace Manager.

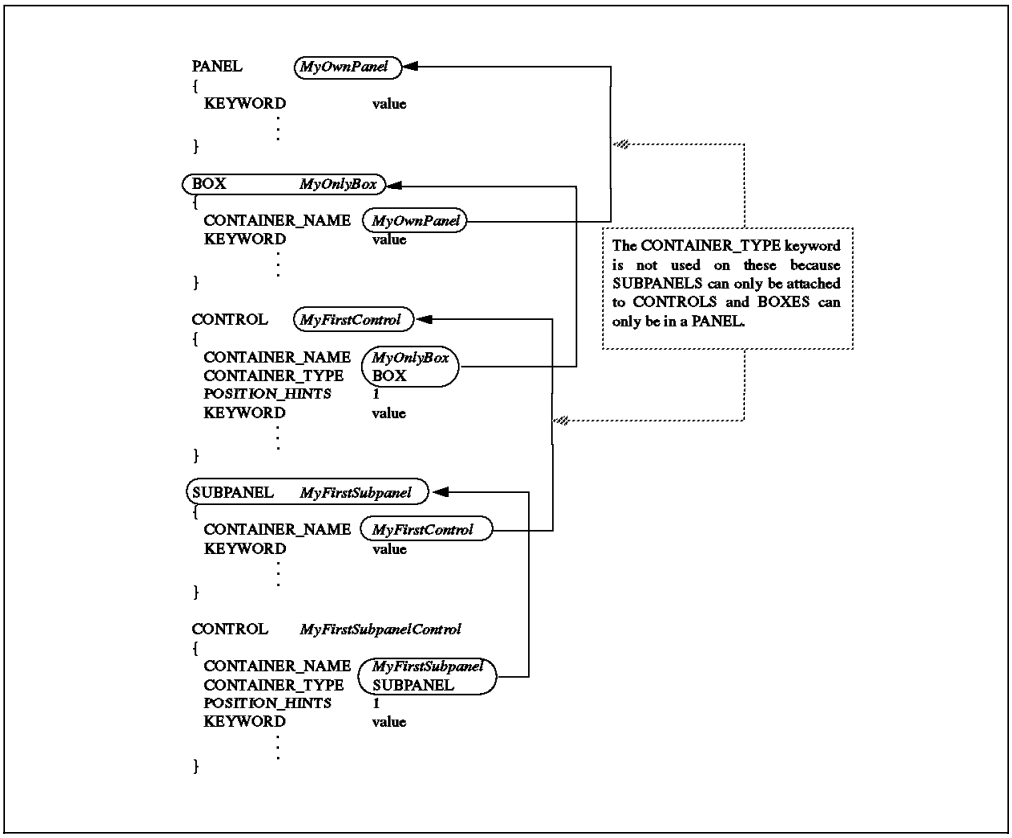


Figure 13. Front Panel Component Parent Trace

4.3.1 Creating Boxes

A front panel definition must contain at least one box component. Each box creates one row in the front panel. When using multiple boxes to create a multi-row front panel, be sure and give each box a unique name. To define a box component:

1. Use the box component syntax described in step 2 on page 37 to define the box and place it in a file, with a .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the box a unique name
   - Use the CONTAINER_NAME keyword to define which front panel definition holds the control
   - Use the POSITION_HINTS keyword to position the box within the front panel container
   - If you do not want this control to be overridden by another definition use the LOCKED keyword and specify a value of True
   - If this control is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information
2. Save the front panel file
3. Restart the Workspace Manager to rebuild the desktop

An example of a box can be found in the example front panel found in 4.4, “Example Front Panel” on page 66.

4.3.2 Creating Controls

There are seven different types of controls that can be defined in a front panel:

1. Icon
2. Blank
3. Busy
4. Client
5. Clock
6. Date
7. File

See 4.1.1, “Front Panel Components” on page 32 for more information on the seven different control types.

The blank, busy, clock, and date control types are self-explanatory and can be copied and used directly from the default front panel definition by only modifying the control′s parent, position and other keywords as needed. The icon, client and file control types are more complex and will be covered in the following sections.

4.3.2.1 Icon Controls

This is the most common type of control in a front panel definition. This control represents a desktop object that is imbedded into the front panel or one of its subpanels. Before you can add an icon control you must first:

1. Create and place in the DTICONSEARCHPATH both the 48x48 and 32x32 pixel icons which will be used to represent the control. See 11.1, “Creating Your Own Icons” on page 161 for more information on icons.
2. Create an action that will be associated with the icon control. The easiest way to do this is to create a desktop object using the CreateAction facility described in 8.4, “Action and Data Type Database” on page 126.

Once the action and icons have been created you can create the icon control by either:

- Dragging the object that contains the desired action from the File Manager and dropping it onto an install icon target area in a subpanel. For more information see 4.2.3.1, “Adding Controls to a Subpanel Using Interactive Customization” on page 52.
- Defining a control component in a front panel definition file which calls the desired action. This is done by:

  1. Using the control component syntax described in step 3 on page 38, to define the control and place it in a file, with a .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
     - Give the component a unique name.
     - Use the CONTAINER_NAME and CONTAINER_TYPE keywords to define which parent container will hold the control.
     - Use the POSITION_HINTS keyword to position the control within the parent container.
     - Use the PUSH_ACTION and or DROP_ACTION keywords to specify the desired action to be performed when the control is selected or an object is dropped onto the control.
     - Specify the icon to be used for the control with the ICON keyword. If the icon is to animate when selected. Use the PUSH_ANIMATION keyword to specify the icon animation sequence to be displayed if the control is selected with a mouse click. Use the DROP_ANIMATION keyword to specify the icon animation sequence to be displayed if the control is selected with an object being dropped onto it. If you are defining a control that monitors a file or for incoming mail, you will also need to include the ALTERNATE_ICON keyword to specify the icon that is to be displayed when the monitored file is created or mail arrives.
     - Include the LABEL keyword if you want a label to be displayed along with the control.
     - Use the PUSH_RECALL keyword if you only want the user to bring up one instance of the action. For example, the control that brings up the Style Manager uses this keyword to ensure that the user only brings up one copy of the Style Manager.
     - If this control is to be setup to monitor a file, use the MONITOR_TYPE keyword to specify the value: file. Then use the FILE_NAME keyword to specify the file to monitor. If this control is to be set up to monitor

for incoming mail use the MONITOR_TYPE keyword to specify the value: mail.

- If you do not want this control to be overridden by another definition, use the LOCKED keyword and specify a value of True.
- If this control is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information.

2. Saving the front panel file.
3. Restarting the Workspace Manager to rebuild the desktop.

Several examples of icon controls can be found in the example front panel found in 4.4, “Example Front Panel” on page 66.

4.3.2.2 Client Controls

Client controls are actually X Window applications that display their output right in the front panel. To be an effective client control, the X Window application has to be capable of displaying its output in a small window. Although the front panel will re-size itself to fit the application′s output, an application that produces a large output window will make the front panel rather unusable. To define a client control:

1. Use the control component syntax described in step 3 on page 38 to define the control and place it in a front panel file, with the .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the component a unique name.
   - Use the TYPE keyword and specify the value: client.
   - Use the CONTAINER_NAME and CONTAINER_TYPE keywords to define which parent container will hold the control.
   - Use the POSITION_HINTS keyword to position the control within the parent container.
   - Use the CLIENT_NAME keyword to specify the window name of the application to be imbedded into the front panel. The name to specify can be obtained by using the xprop X Window sample program to retrieve the window name of the application while it is running. To do this:
     a. Start the application to be imbedded into the front panel.
     b. In a terminal window issue the xprop command. This command is usually located in the /usr/lpp/X11/bin directory.
     c. Move the mouse pointer over the window to be imbedded into the front panel and click the left mouse button.
     d. In the output of the xprop command displayed in the terminal window you should see a line that looks like the following for the xload sample application :

        ```
        WM_CLASS(STRING) = "xload", "XLoad"
        ```

        The first quoted string is the application window name. This is the string that should be included as the value on the CLIENT_NAME keyword.
   - Use the CLIENT_GEOMETRY keyword to specify the amount of space to reserve in the front panel for the application′s window. The size to

specify can be obtained by using the xwininfo X-Windows sample program to retrieve the size of the running application's window. To do this:

a. Start the application to be imbedded into the front panel.

b. Re-size the application window to the desired size.

c. In a terminal window issue the xwininfo command. This command is usually located in the /usr/lpp/X11/bin directory.

d. Move the mouse pointer over the window to be imbedded into the front panel and click the left mouse button.

e. In the output of the xwininfo command displayed in the terminal window you should see a line that looks like the following for the xload sample program:

```
-geometry 120x62+53+30
```

The numbers that are on both sides of the x character are the width and height of the window in pixels. In this example the width is 120 and the height is 62 pixels. On the CLIENT_GEOMETRY keyword specify this width and height as widthxheight.

- Use the PUSH_ACTION and or DROP_ACTION keywords to specify the desired action to be performed when the control is selected or an object is dropped onto the control.
- Include the LABEL keyword if you want a label to be displayed along with the control.
- Use the PUSH_RECALL keyword if you only want the user to bring up one instance of the action. For example, the control that brings up the Style Manager uses this keyword to ensure that the user only brings up one copy of the Style Manager.
- If you do not want this control to be overridden by another definition use the LOCKED keyword and specify a value of True.
- If this control is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information.

2. Save the front panel file.
3. Start the application to be imbedded in the front panel. When you restart the Workspace Manager in the next step, the applications window will be imbedded into the front panel automatically.

   If the application supports session management protocols, the next time you log out the Session Manager will record the fact that you had the application running and handle restarting it for you on subsequent logins. See 3.1, "Customizing the Session Manager" on page 22 for more information on session management.

   If the application does not support session management, you will need to restart the application by putting an entry in your $HOME/.dt/sessions/sessionetc file. See Chapter 3, "Session Manager" on page 21 for more information on the sessionetc file.
4. Restart the Workspace Manager to rebuild the desktop.

An example of a client control can be found in the example front panel found in 4.4, "Example Front Panel" on page 66.

4.3.2.3 File Controls

File controls when selected run their associated action on a specified file. For example, you can put file controls in the front panel or in subpanels that represent an inventory database, a personnel database and a sales database. When you select any of these databases from the front panel or subpanel, the associated database manager software will be started and the selected database will automatically be loaded. To define a file control:

1. Use the control component syntax described in step 3 on page 38 to define the control and place it in a front panel file, with the .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the component a unique name.
   - Use the TYPE keyword and specify the value: file.
   - Use the CONTAINER_NAME and CONTAINER_TYPE keywords to define which parent container will hold the control.
   - Use the POSITION_HINTS keyword to position the control within the parent container.
   - Use the FILE_NAME keyword to specify the full path of the file to pass to the specified action.
   - Use the PUSH_ACTION keyword to specify the desired action to be performed on the specified file when the control is selected.
   - Use the ICON keyword to specify the icon to be used for the control. If you are defining a control that monitors a file, you will also need to include the ALTERNATE_ICON keyword to specify the icon that is to be displayed when the monitored file changes.
   - Include the LABEL keyword if you want a label to be displayed along with the control.
   - If you do not want this control to be overridden by another definition use the LOCKED keyword and specify a value of True.
   - If this control is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information.
2. Save the front panel file.
3. Restart the Workspace Manager to rebuild the desktop.

4.3.3 Creating Subpanels

Each control in the front panel can have at most one subpanel attached to it. Even though the subpanel is, by its definition, attached to the control in the front panel, you should think of the subpanel as being attached to the front panel control position. This is because the subpanel will remain above the control position in the front panel even if the control to which the subpanel is attached is moved up into the subpanel.

Subpanels can only be attached to front panel controls. They cannot be attached to controls in a subpanel or controls in the switch.

To define a subpanel component:

1. Define or select a front panel control to be the parent of this subpanel.

2. Use the subpanel component syntax described in step 4 on page 42 to define the subpanel and place it in a file, with a .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the subpanel a unique name.
   - Use the CONTAINER_NAME keyword to define which of the front panel controls will have this subpanel attached to it.
   - Use the CONTROL_INSTALL keyword to indicate if the subpanel is to include the install icon area. By default this area will be included.
   - If you do not want this subpanel to be overridden by another definition use the LOCKED keyword and specify a value of True.
   - If this subpanel is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information.
3. Save the front panel file.
4. Restart the Workspace Manager to rebuild the desktop.

An example of a subpanel can be found in the example front panel found in 4.4, “Example Front Panel” on page 66.

4.3.4 Creating the Switch

There can be at most one switch defined in a front panel. The switch is placed in one of the front panel boxes. The switch is also a container that can hold controls. The number of controls which can be put in the switch is at most two times the number of rows in the switch. The controls that are put in the switch can be of type: blank, busy, icon or file. And must have an icon that is 16x16 pixels in size. We found that it is possible to to put a client type control in the switch, as long as the control was placed with POSITION_HINTS one, two or four.

To define a switch component:

1. Define or select a front panel box to be the parent of the switch.
2. Use the switch component syntax described in step 5 on page 43 to define the switch and place it in a file, with a .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the switch a unique name.
   - Use the CONTAINER_NAME keyword to define which of the front panel boxes will have the switch in it.
   - Use the POSITION_HINTS keyword to position the switch within the parent box.
   - Specify the number of rows in the switch on the NUMBER_OF_ROWS keyword.
   - If you do not want the switch to be overridden by another definition use the LOCKED keyword and specify a value of True.
   - If the switch is to have help information available use the HELP_STRING, HELP_TOPIC and HELP_VOLUME keywords to specify the help information.
3. Save the front panel file.
4. Restart the Workspace Manager to rebuild the desktop.

An example of a switch can be found in the example front panel found in 4.4, “Example Front Panel” on page 66.

4.3.5 Creating an Animated Control

The icons that represent file and icon controls in the front panel can animate when selected. To create an animated control:

1. Create a sequence of 64x64 pixel icons that make up the animation sequence to be run when the control is selected and place them the DTICONSEARCHPATH. Different animation sequences can be created for both the push selection and the object drop selection. See 11.1, “Creating Your Own Icons” on page 161 for more information on creating icons.
2. Use the animation component syntax described in step 6 on page 44 to define the animation and place it in a file, with a .fp suffix, in a directory in the DTDATABASESEARCHPATH. In the component definition:
   - Give the animation a unique name.
   - Use multiple ANIMATION keywords to define the icon sequence to display. Each ANIMATION keyword defines the icon to be displayed in sequence. For example, if your animation sequence has 15 icons in it, the 13th icon in the sequence is defined by the 13th ANIMATION keyword in the definition. An optional delay parameter can be appended to any or all ANIMATION keywords to specify a delay in milliseconds to hold the particular icon on the screen before displaying the next one in the sequence.
3. Update the PUSH_ANIMATION and/or the DROP ANIMATION keywords for the control to be animated with the name of this animation component.
4. Save the front panel file.
5. Restart the Workspace Manager to rebuild the desktop..

An example of a animated control can be found in the example front panel found in 4.4, “Example Front Panel.”

4.4 Example Front Panel

The front panel pictured in Figure 14 on page 67 was built from the front panel definition listed following the picture. This definition was created by using the default front panel as a model and then modifying it as needed to create the front panel in the picture. The components identified in the picture correspond to the component definitions in the definition listing.

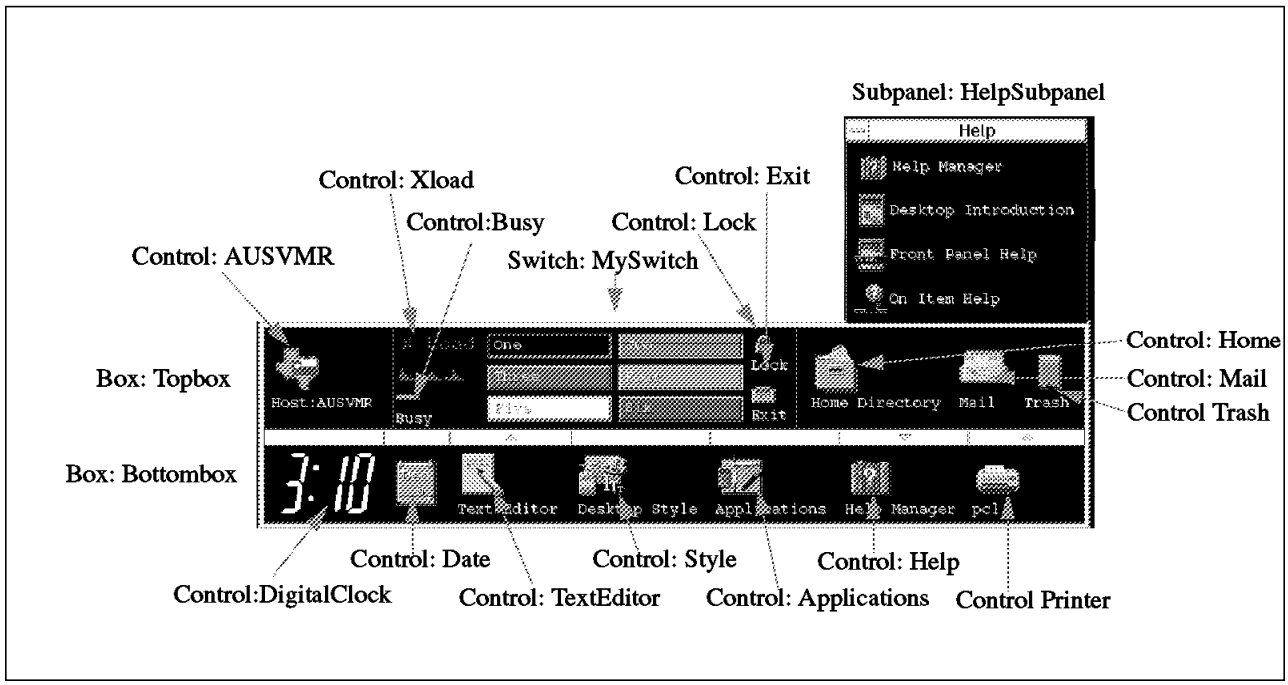


Figure 14. Example Front Panel

```
#
#  This file contains a full definition for a front panel called TwoRowPanel
#  This panel was built using the default front panel as a model.

PANEL TwoRowPanel
{
  DISPLAY_HANDLES       False           <== No display handles
  DISPLAY_MENU          False           <== No window menu
  DISPLAY_MINIMIZE      False           <== No minimize button
  CONTROL_BEHAVIOR      single_click
  HELP_TOPIC            FPOnItemFrontPanel
  HELP_VOLUME           FPanel
  DISPLAY_CONTROL_LABELS True           <== Show labels
}

BOX Topbox                              <== The top box
{
  CONTAINER_NAME        TwoRowPanel
  POSITION_HINTS        first
  HELP_TOPIC            FPOnItemBox
  HELP_VOLUME           FPanel
}


CONTROL AUSVMR                          <== Icon control that only
{                                           has a push action
  TYPE                  icon
  CONTAINER_NAME        Topbox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        1
  ICON                  Texas
  LABEL                 Host:AUSVMR
  PUSH_ACTION           AUSVMR
}



CONTROL Blank1                          <== Blank spacer
{
  TYPE                  blank
  CONTAINER_NAME        Topbox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        2
  ICON                  Fpblank
```

```
  HELP_TOPIC            FPOnItemFrontPanel
  HELP_VOLUME           FPanel
}

SWITCH           MySwitch               <== My new switch
{
  CONTAINER_NAME        Topbox
  POSITION_HINTS        2
  NUMBER_OF_ROWS        3              <== 3 rows deep
  HELP_TOPIC            FPOnItemSwitch
  HELP_VOLUME           FPanel
}

CONTROL Xload                          <== A client in the switch
{
  TYPE                  client
  CONTAINER_NAME        MySwitch
  CONTAINER_TYPE        SWITCH
  POSITION_HINTS        1
  CLIENT_NAME           xload
  CLIENT_GEOMETRY       80x48
}

CONTROL Lock                           <== Icon control with only
{                                          a push action
  TYPE                  icon
  CONTAINER_NAME        MySwitch
  CONTAINER_TYPE        SWITCH
  POSITION_HINTS        2
  ICON                  Fplock
  LABEL                 Lock
  PUSH_ACTION           LockDisplay
  HELP_TOPIC            FPOnItemLock
  HELP_VOLUME           FPanel
}

CONTROL Busy                           <== Busy control
{
  TYPE                  busy
  CONTAINER_NAME        MySwitch
  CONTAINER_TYPE        SWITCH
  POSITION_HINTS        3
  ICON                  Fplite           <== When active it will alternate
  ALTERNATE_ICON        FpliteY              between these two icons
  LABEL                 Busy
  HELP_TOPIC            FPOnItemBusy
  HELP_VOLUME           FPanel
}

CONTROL Exit                            <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        MySwitch
  CONTAINER_TYPE        SWITCH
  POSITION_HINTS        4
  ICON                  Fpexit
  LABEL                 Exit
  PUSH_ACTION           ExitSession
  HELP_TOPIC            FPOnItemExit
  HELP_VOLUME           FPanel
}

CONTROL Blank2                         <== Another blank spacer
{
  TYPE                  blank
  CONTAINER_NAME        Topbox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        3
  ICON                  Fpblank
  HELP_TOPIC            FPOnItemFrontPanel
  HELP_VOLUME           FPanel
}

CONTROL Home                           <== Icon control with only
{                                          a push action
```

```
  TYPE                icon
  CONTAINER_NAME      Topbox
  CONTAINER_TYPE      BOX
  POSITION_HINTS      4
  ICON                Fphome
  LABEL               Home Directory
  PUSH_ACTION         OpenHomeDir
  HELP_TOPIC          FPOnItemHome
  HELP_VOLUME         FPanel
}

CONTROL Mail                          <== Icon control that monitors
{                                         for incoming mail
  TYPE                icon
  CONTAINER_NAME      Topbox
  CONTAINER_TYPE      BOX
  POSITION_HINTS      5
  ICON                DtMail          <== No mail icon
  LABEL               Mail            <== Mail received icon
  ALTERNATE_ICON      DtMnew          <== Monitor for incoming mail
  MONITOR_TYPE        mail
  PUSH_ACTION         Mailer
  PUSH_RECALL         true            <== Only bring up 1 mailer
  CLIENT_NAME         Mail            <== Its X name is Mailer
  HELP_TOPIC          FPOnItemMail
  HELP_VOLUME         FPanel
}

CONTROL Trash                         <== Icon control that monitors
{                                         for the creation of a file
  TYPE                icon
  CONTAINER_NAME      Topbox
  CONTAINER_TYPE      BOX
  POSITION_HINTS      6
  ICON                Fptrsh          <== No trash icon
  ALTERNATE_ICON      Fptrsh7         <== Trash exists icon
  LABEL               Trash
  FILE_NAME           $HOME/.dt/Trash/.trashinfo <== File to watch
  MONITOR_TYPE        file            <== Monitor for file creation
  PUSH_ACTION         OpenTrash       <== Accept both mouse clicks and
  DROP_ACTION         TrashFile           object drops
  DROP_ANIMATION      TrashDrop       <== If an object is dropped animate
  HELP_TOPIC          FPOnItemTrash
  HELP_VOLUME         FPanel
}
BOX Bottombox                         <== The bottom box
{
  CONTAINER_NAME      TwoRowPanel
  POSITION_HINTS      last
  HELP_TOPIC          FPOnItemBox
  HELP_VOLUME         FPanel
}

CONTROL DigitalClock                  <== A client control
{
  TYPE                client
  CONTAINER_NAME      Bottombox
  CONTAINER_TYPE      BOX
  POSITION_HINTS      1
  CLIENT_NAME         dclock
  CLIENT_GEOMETRY     110x48
}

CONTROL Date                          <== Date control
{
  TYPE                date
  CONTAINER_NAME      Bottombox
  CONTAINER_TYPE      BOX
  POSITION_HINTS      2
  ICON                DtCM
  LABEL               Date
  PUSH_RECALL         True            <== Only bring up 1 calendar
  PUSH_ACTION         Calendar        <== Bring up calendar if clicked
  DROP_ACTION         CalenderInsert  <== Add to calendar if dropped
  HELP_TOPIC          FPOnItemDate
```

```
  HELP_VOLUME           FPanel
}

CONTROL TextEditor                      <== Icon control with the same
{                                           push and drop action
  TYPE                  icon
  CONTAINER_NAME        Bottombox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        3
  ICON                  Fppenpd
  LABEL                 Text Editor
  PUSH_ACTION           EditText        <== Empty text editor if clicked
  DROP_ACTION           EditText        <== Load file if dropped
  HELP_TOPIC            FPOnItemTextEditor
  HELP_VOLUME           FPanel
}

SUBPANEL PersAppsSubpanel               <== A subpanel above the
{                                           text editor control
  CONTAINER_NAME        TextEditor
  TITLE                 Personal Applications
}
                                        <== Icon control with just
CONTROL Term                                a push action
{
  TYPE                  icon
  CONTAINER_NAME        PersAppsSubpanel
  CONTAINER_TYPE        SUBPANEL
  POSITION_HINTS        1
  ICON                  Fpterm
  LABEL                 Terminal
  PUSH_ACTION           Terminal
  HELP_TOPIC            FPOnItemTerm
  HELP_VOLUME           FPanel
}

CONTROL IconEditor                      <== Icon control with the same
{                                           push and drop action
  TYPE                  icon
  CONTAINER_NAME        PersAppsSubpanel
  CONTAINER_TYPE        SUBPANEL
  POSITION_HINTS        2
  ICON                  Dtpaint
  LABEL                 Icon Editor
  PUSH_ACTION           IconEditor      <== Empty icon editor if clicked
  DROP_ACTION           IconEditor      <== Load icon if dropped
  HELP_TOPIC            FPOnItemIconEditor
  HELP_VOLUME           FPanel
}


CONTROL Style                           <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        Bottombox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        4
  LABEL                 Desktop Style
  ICON                  Fpstyle
  PUSH_ACTION           OpenStyleMgr
  PUSH_RECALL           true            <== Bring up only 1 Style Manager
  CLIENT_NAME           dtstyle         <== Its X name is dtstyle
  HELP_TOPIC            FPOnItemStyle
  HELP_VOLUME           FPanel
}


CONTROL Applications                    <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        Bottombox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        5
  ICON                  Fpapps
  LABEL                 Applications
```

```
  PUSH_ACTION           OpenApplicationManager
  HELP_TOPIC            FPOnItemAppMgr
  HELP_VOLUME           FPanel
}

CONTROL Help                            <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        Bottombox
  CONTAINER_TYPE        BOX
  POSITION_HINTS        6
  ICON                  Fphelp
  LABEL                 Help Manager
  PUSH_ACTION           OpenHelpManager
  HELP_TOPIC            FPOnItemHelpMgr
  HELP_VOLUME           FPanel
}

SUBPANEL HelpSubpanel                   <== A subpanel above the help
{                                           control
  CONTAINER_NAME        Help
  TITLE                 Help
  CONTROL_INSTALL       False             <== Don't include install icon area
}

CONTROL HelpOverview                    <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        HelpSubpanel
  CONTAINER_TYPE        SUBPANEL
  POSITION_HINTS        1
  ICON                  Dthover
  LABEL                 Desktop Introduction
  PUSH_ACTION           OpenDtIntro
  HELP_TOPIC            FPOnItemDtIntro
  HELP_VOLUME           FPanel
}

CONTROL FPHelp                          <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        HelpSubpanel
  CONTAINER_TYPE        SUBPANEL
  POSITION_HINTS        2
  ICON                  Fpfphlp
  LABEL                 Front Panel Help
  PUSH_ACTION           FPHelp
  HELP_TOPIC            FPOnItemFPHelp
  HELP_VOLUME           FPanel
}

#
# OnItem help uses a pseudo push action FPOnItemHelp.
# Dtwm is looking for an exact match on this push action string.
# Do no  t localize this push action.
#

CONTROL OnItem                          <== Icon control with only
{                                           a push action
  TYPE                  icon
  CONTAINER_NAME        HelpSubpanel
  CONTAINER_TYPE        SUBPANEL
  POSITION_HINTS        3
  ICON                  DthonFP
  LABEL                 On Item Help
  PUSH_ACTION           FPOnItemHelp
  HELP_TOPIC            FPOnItemOnItem
  HELP_VOLUME           FPanel
}


CONTROL Printer                         <== Icon control with both
{                                           push and drop actions
  TYPE                  icon
  CONTAINER_NAME        Bottombox
```

```
  CONTAINER_TYPE        BOX
  POSITION_HINTS        7
  LABEL                 Default
  ICON                  Fpprnt
  PUSH_ACTION           PrintManager     <== Print Manager if clicked
  DROP_ACTION           Print            <== Print file if dropped
  DROP_ANIMATION        PrinterDrop      <== If dropped animate
  HELP_TOPIC            FPOnItemPrinter
  HELP_VOLUME           FPanel
}

SUBPANEL PersPrintersSubpanel            <== A subpanel above the printer
                                             control; it is empty
{
   CONTAINER_NAME       Printer
   TITLE                Personal Printers
}

ANIMATION TrashDrop                      <== Trash drop animation sequence
{
   ANIMATION    Fptrsh1         100      <== 1st icon; Wait 100 milliseconds
   ANIMATION    Fptrsh2         200      <== 2nd icon; Wait 200 milliseconds
   ANIMATION    Fptrsh3         100      <== 3rd icon; Wait 100 milliseconds
   ANIMATION    Fptrsh4                  <== 4th icon; Default wait
   ANIMATION    Fptrsh5         800      <== 5th icon; Wait 800 milliseconds
   ANIMATION    Fptrsh6         200      <== 6th icon; Wait 200 milliseconds
   ANIMATION    Fptrsh7                  <== 7th icon; Default wait
   ANIMATION    Fptrsh                   <== Last icon; Default wait
}

ANIMATION PrinterDrop                    <== Printer drop animation sequence
{
   ANIMATION    Fpprnt1         100      <== 1st icon; Wait 100 milliseconds
   ANIMATION    Fpprnt2         100      <== 2nd icon; Wait 100 milliseconds
   ANIMATION    Fpprnt3         100      <== 3rd icon; Wait 100 milliseconds
   ANIMATION    Fpprnt4         100      <== 4th icon; Wait 100 milliseconds
   ANIMATION    Fpprnt5         100      <== 5th icon; Wait 100 milliseconds
   ANIMATION    Fpprnt6         100      <== 6th icon; Wait 100 milliseconds
   ANIMATION    Fpprnt7         100      <== 7th icon; Wait 100 milliseconds
   ANIMATION    Fpprnt8         100      <== 8th icon; Wait 100 milliseconds
   ANIMATION    Fpprnt9         100      <== 9th icon; Wait 100 milliseconds
   ANIMATION    FpprntA         100      <== 10th icon; Wait 100 milliseconds
   ANIMATION    FpprntB         800      <== Last icon; Wait 800 milliseconds
}
```

Chapter 5. Style Manager

The Style Manager component of the desktop allows the user to interactively customize the look and operation of their desktop environment. The Style Manager's main user interface is a graphical menu (pictured in Figure 15) which consists of icons that represent the available customization tools. These tools allow user customization in the following areas:

- Color - Sets the color palette and the color usage for all desktop windows and supporting applications
- Font - Sets the font size used by text and labels in the desktop and supporting applications
- Backdrop - Sets the backdrops(wallpaper) used by the various workspaces
- Keyboard - Sets keyboard characteristics for key press click volume and repeating characters when a key is held down
- Beep - Sets the system speaker characteristics for beep volume, tone and duration
- Screen - Sets the use and operation of a screen saver and the screen lock
- Window - Sets window management characteristics for input focus, window dragging visuals and icon placement for minimized applications
- Startup - Sets characteristics of Session Manager operation that control which session to restart at log in, log out confirmation usage and setting the home session

This section will discuss these environment customization tools in different depths. Some of the tools are very self-explanatory and will be briefly described, while others require a more in depth explanation in order for the user to fully benefit from their capabilities.

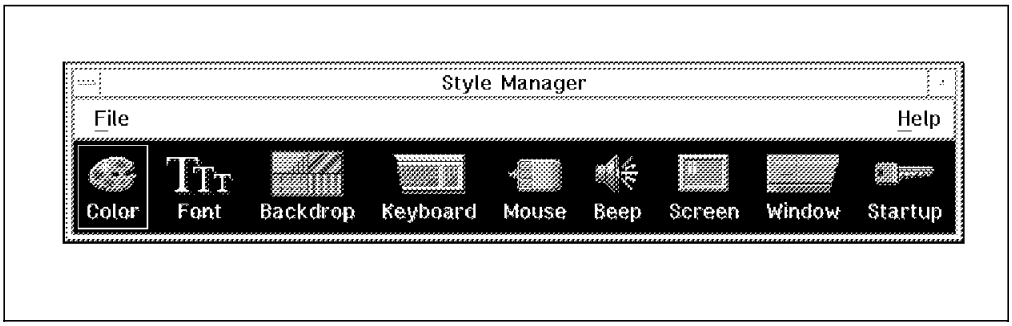


Figure 15. Style Manager Main User Interface

5.1 Customizing the Style Manager

The Style Manager can be customized by altering the X resources that define its behavior and appearance. Two of the available resources that are worth noting here are:

Dtstyle*componentList:

This resource specifies the customization tools that are available for the user. A system administrator can alter this resource to remove

customization tools from their user's Style Manager dialog. For example, to only allow your users to change their colors, font sizes and backdrops specify the resource as follows:

```
Dtstyle*componentList: Color Font Backdrop
```

Dtstyle*mainRC.orientation:

This resource specifies the orientation of the Style Manager. To orient the Style Manager vertically, comment out this resource in your local customized Dtstyle file.

These resources are specified in the /etc/dt/app-defaults/%LANG/Dtstyle file. If this file does not exist, copy the supplied default version from /usr/dt/app-defaults/%LANG/Dtstyle and modify it.

5.2 Color Customization

The desktop is an X Window application and thus is bound by the color limitations of X Windows for the particular hardware (display and display adapter) on which you are running the desktop. For example, If you are running on a monochrome system you may want to change the color scheme used by the desktop to one that is optimized to a monochrome environment. Similarly, if you are on a color display and adapter, you may want to choose a color scheme that is effective for the color environment.

The Style Manager's color customization tool (pictured in Figure 16) allows the user to customize the colors used by the desktop and supporting applications for displaying such things as window backgrounds, borders, and menus. This tool permits the user to:

- Select a named color palette from an available list
- Modify the colors in the selected color palette
- Save and delete color palettes from the available list
- Set the number of colors available in a color palette

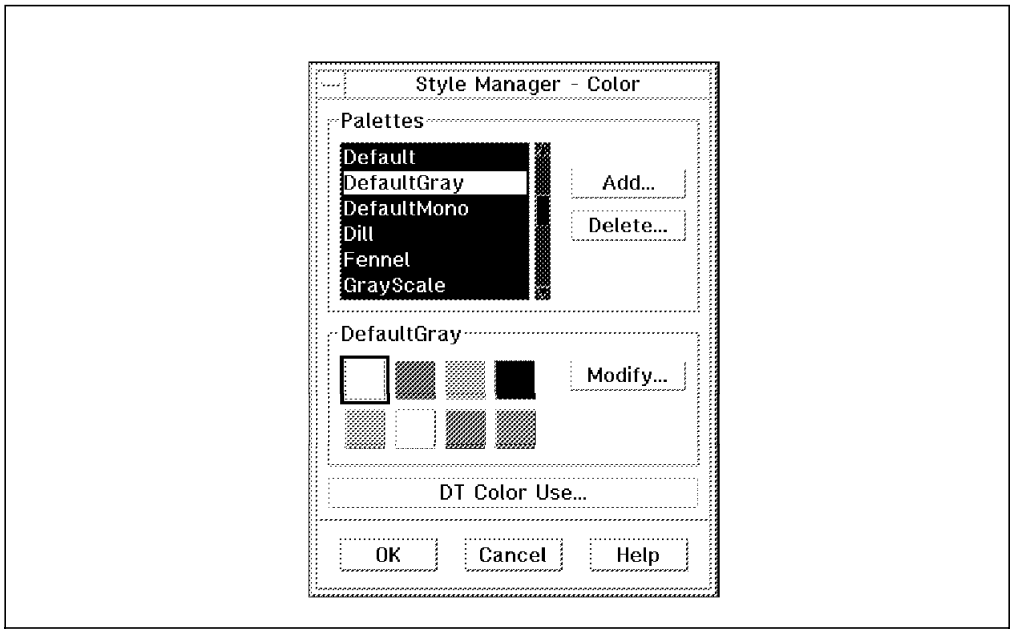


Figure 16. Style Manager Color Dialog

5.2.1 Desktop Use of Color

In order to fully utilize the color customization capabilities of the Style Manager, you need to first understand how the desktop uses colors. The desktop uses the concept of a color palette, made up of groups of five colors called color sets. Each color set has a user definable background color which used to color a window's background. The colors in a window such as foreground color, top and bottom shadow colors, and select color are the other four colors in a color set and are automatically chosen by the desktop to complement the selected background color. Each of the color sets in a palette (there can be up to eight of them) are assigned an ID. These color set IDs are used to assign which of the color sets is to be used to display a particular element of the user interface. For example, a color set can be assigned to color the frame around the window that currently has input focus. Each of the colors in the assigned color set are used to color a different part of the window frame. The background color is used as the frame background. The three dimensional shadows of the frame are colored from the top shadow and bottom shadow colors in the color set. The text in the title area of the frame is colored with the color set's foreground color. If the window frame had an editable input area or a scroll bar the color set's select color would be used to color these. The picture in Figure 17 illustrates this color concept.

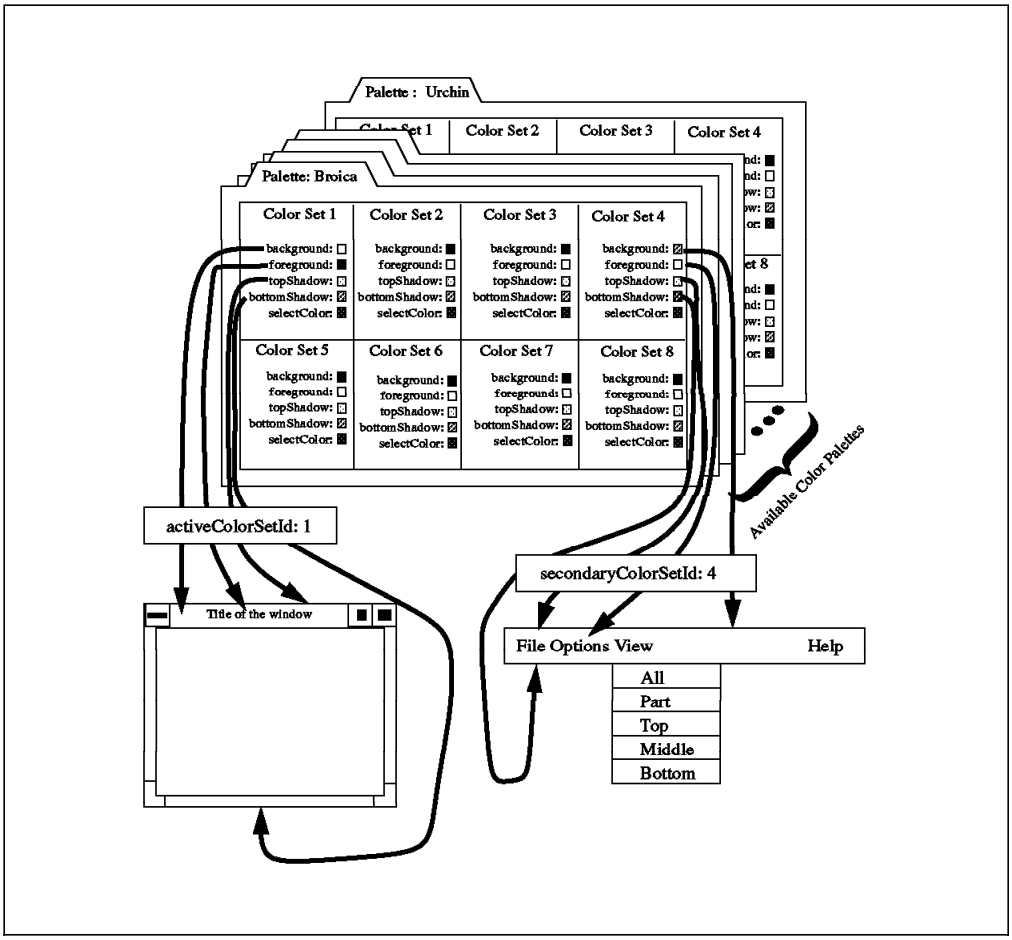


Figure 17. Desktop Color Concept

The colored buttons on the Style Managers color dialog (pictured in Figure 16 on page 74) are the available background colors for each available color set. In this picture, there are eight different buttons. These colored buttons correspond to

the different color set IDs. ID one is the button in the top row first column. The ID numbers continue sequentially from left to right, top to bottom, until the maximum number of color sets in the palette is reached. In this picture, ID eight is the button in the second row last column. This color set ID is used to reference which color set in the palette is to be used for a particular screen element. This mapping from screen element to color set ID is done using the following X Window resources:

Table 8. Mapping Color Sets to Screen Elements

Resource	Value	Default	Description
activeColorSetId	1 - Max available	1	Active window frame color
inactiveColorSetId	1 - Max available	2	Inactive window frame color
primaryColorSetId	1 - Max available	3	Main background areas
secondaryColorSetId	1 - Max available	4	Menu bar, menus and dialog boxes

The desktop and its clients all use color set ID three for their primaryColorSetId but vary the secondaryColorSetId.

Each of the background colors in a color set can be changed to a different color (thus creating a new palette). The following procedure illustrates how to do this:

1. Select the colored button on the color dialog that corresponds to the desired background color to change. The button should highlight with a box around it.
2. Select the **Modify...** button. This action will bring up the Modify Color pop window.
3. Use the sliders on this pop up window to alter the background color of the selected color set. The complementary colors in the color set (shadows, foreground and selection colors) are automatically selected by the system as you change the background color.
4. When you achieve the desired new color set combination (selected background plus supplied complementary colors) select the **OK** button to accept your new change.
5. Save the new color palette by selecting the **Add..** button and entering a unique name for the new palette.

The number of color sets to choose from in a color palette is dependent on the value of the Color Use parameter. This parameter can be set interactively through the Color Use dialog pictured in Figure 18 on page 77. To access this dialog, select the **Color Use...** button on the Style Manager color dialog.

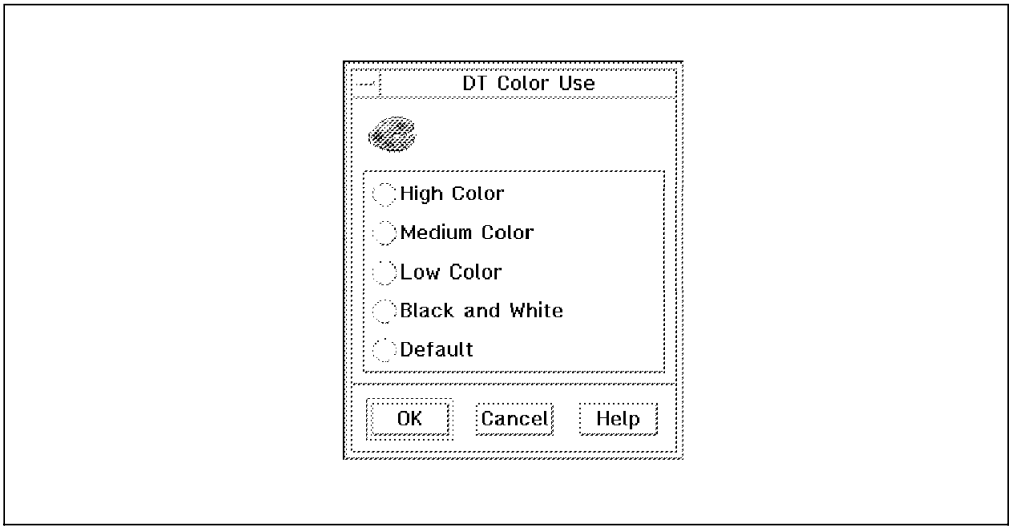


Figure 18. Color Use Dialog

The choices are available on this dialog are used to control how many color sets that are available in a desktop color palette. The choices are:

High Color
Eight color sets in the palette

Medium Color
Four color sets in the palette

Low Color
Two color sets in the palette

Black and White
Two color sets (black and white only) in the palette

Default
Allows the desktop to choose based on the current display being used

The choice to select is dependent on the following two factors:

- The type of display and display adapter that you are using. Monochrome systems should used Black and White. While more powerful color systems can use the High color option.
- The number of unique colors that you want to make available in the shared X Window color table for applications running under the desktop. Some applications may require a lot of unique colors to operate. In order to avoid having those applications load their own color map, thus avoiding the color map flashing effect when focus is changed from the desktop to the applications, you can restrict the number of colors that the desktop reserves. Since each color set is made up of five different colors, the more color sets that you make available through the setting of the color use parameter, the more colors that the desktop has to reserve for your desktop session. Therefore, if you have an application that requires a lot of colors, you may want to sacrifice some of the visual appeal of a full color desktop in order to free up as may colors as possible for your application.

5.3 Font Size Customization

The Style Manager's font size customization tool (pictured in Figure 19) allows the user to change the size of the font that is used by the desktop, its dialogs and applications that do not explicitly set a font in their program. Although this dialog only allows you to change the size of the font and not the font family, you can use the following workaround to overcome this limitation.

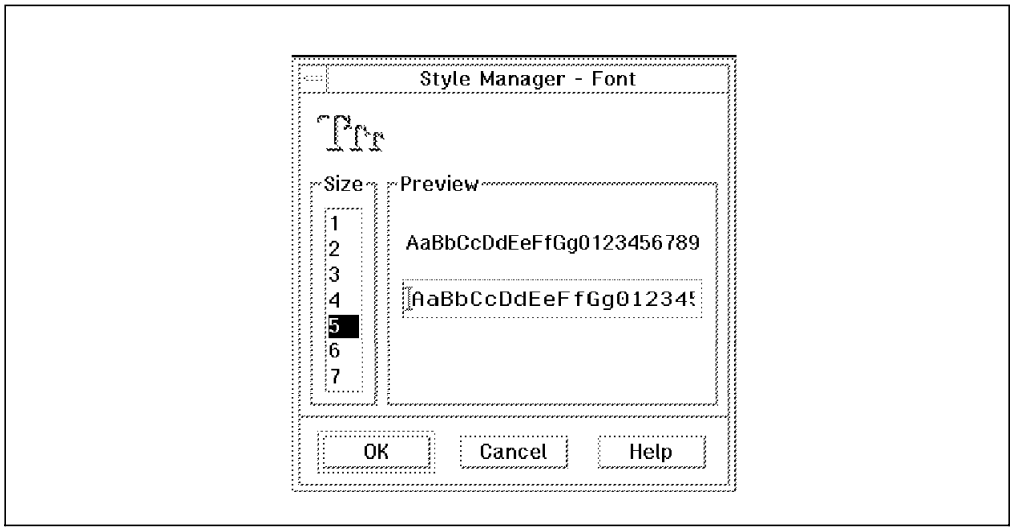


Figure 19. Style Manager Font Dialog

The font dialog lists the different font sizes as index numbers into font resources which are specified in the /usr/dt/app-defaults/%LANG/Dtstyle file. Instead of specifying different font sizes for the same font in the resources, as intended, you could specify different font families and sizes on these resources. There are actually two different types of fonts that can be set using these resources:

SystemFont This is the font that is used for system areas such as titles and text in and on menu bars, menu panes, push buttons, toggle buttons and labels

UserFont This is the font used for text entered in windows, for example, the default font used by the aixterm terminal emulator

In the /usr/dt/app-defaults/%LANG/Dtstyle file you will find the resources that look like the following from the default file:

```
Dtstyle*NumFonts: 7

Dtstyle*SystemFont1: -dt-interface system-medium-r-normal-xxs*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont2: -dt-interface system-medium-r-normal-xs*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont3: -dt-interface system-medium-r-normal-s*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont4: -dt-interface system-medium-r-normal-m*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont5: -dt-interface system-medium-r-normal-l*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont6: -dt-interface system-medium-r-normal-xl*-*-*-*-*-*-*-*-*-*:
Dtstyle*SystemFont7: -dt-interface system-medium-r-normal-xxl*-*-*-*-*-*-*-*-*-*:

Dtstyle*UserFont1: -dt-interface user-medium-r-normal-xxs*-*-*-*-*-*-*-*-*-*:
Dtstyle*UserFont2: -dt-interface user-medium-r-normal-xs*-*-*-*-*-*-*-*-*-*:
Dtstyle*UserFont3: -dt-interface user-medium-r-normal-s*-*-*-*-*-*-*-*-*-*:
Dtstyle*UserFont4: -dt-interface user-medium-r-normal-m*-*-*-*-*-*-*-*-*-*:
Dtstyle*UserFont5: -dt-interface user-medium-r-normal-l*-*-*-*-*-*-*-*-*-*:
```

```
Dtstyle*UserFont6: -dt-interface user-medium-r-normal-xl*-*-*-*-*-*-*-*-*:
Dtstyle*UserFont7: -dt-interface user-medium-r-normal-xxl*-*-*-*-*-*-*-*-*:
```

The first resource, Dtstyle*NumFonts, specifies the number of both the SystemFont and UserFont resources pairs that will be specified up to a maximum of seven. The SystemFont and UserFont resources specify the mapping between the index in Style Manager font dialog and the desired font. For example, font index four in the Style Manager font dialog is mapped in the above default file to SystemFont:

```
-dt-interface system-medium-r-normal-m*-*-*-*-*-*-*-*-*
```

And UserFont:

```
-dt-interface user-medium-r-normal-m*-*-*-*-*-*-*-*-*
```

You can easily see that you can alter the font definitions on these Dtstyle*SystemFont and Dtstyle*UserFont resource definitions to map to any available font size and/or family. For example, the following resource lines will define two different font families with three font sizes each for both the SystemFont and UserFont:

```
Dtstyle*NumFonts: 6
Dtstyle*SystemFont1: -ibm-serif-*-o-normal-*-19-*-*-*-*-*-*-*:
Dtstyle*SystemFont2: -ibm-serif-*-o-normal-*-22-*-*-*-*-*-*-*:
Dtstyle*SystemFont3: -ibm-serif-*-o-normal-*-25-*-*-*-*-*-*-*:
Dtstyle*SystemFont4: -ibm-typewriter-*-o-normal-*-19-*-*-*-*-*-*-*:
Dtstyle*SystemFont5: -ibm-typewriter-*-o-normal-*-22-*-*-*-*-*-*-*:
Dtstyle*SystemFont6: -ibm-typewriter-*-o-normal-*-25-*-*-*-*-*-*-*:
Dtstyle*UserFont1: -ibm-serif-*-r-normal-*-19-*-*-*-*-*-*-*:
Dtstyle*UserFont2: -ibm-serif-*-r-normal-*-22-*-*-*-*-*-*-*:
Dtstyle*UserFont3: -ibm-serif-*-r-normal-*-25-*-*-*-*-*-*-*:
Dtstyle*UserFont4: -ibm-typewriter-*-r-normal-*-19-*-*-*-*-*-*-*:
Dtstyle*UserFont5: -ibm-typewriter-*-r-normal-*-22-*-*-*-*-*-*-*:
Dtstyle*UserFont6: -ibm-typewriter-*-r-normal-*-25-*-*-*-*-*-*-*:
```

Note: This customization is stored in the /etc/dt/app-defaults/C/Dtstyle file. The version of the file in /usr/dt is the system supplied default and should not be modified. All customizations should be put in the /etc/dt directory. This protects your customizations from being overwritten when you update the desktop software.

5.4 Backdrop Customization

The Style Manager's backdrop customization tool (pictured in Figure 20 on page 80) allows the user to select the backdrop that is displayed in a workspace. This tool will set the backdrop for the current workspace only. To set the backdrop for other workspaces, switch to the desired workspace, open the Style Manager and then select a backdrop for that workspace.

The NoBackdrop option in the list specifies that the workspace is not to use a backdrop. With this set, the X Window's root window will be displayed as the backdrop. This is useful if you want to display an image as the backdrop using a command such as xsetroot.

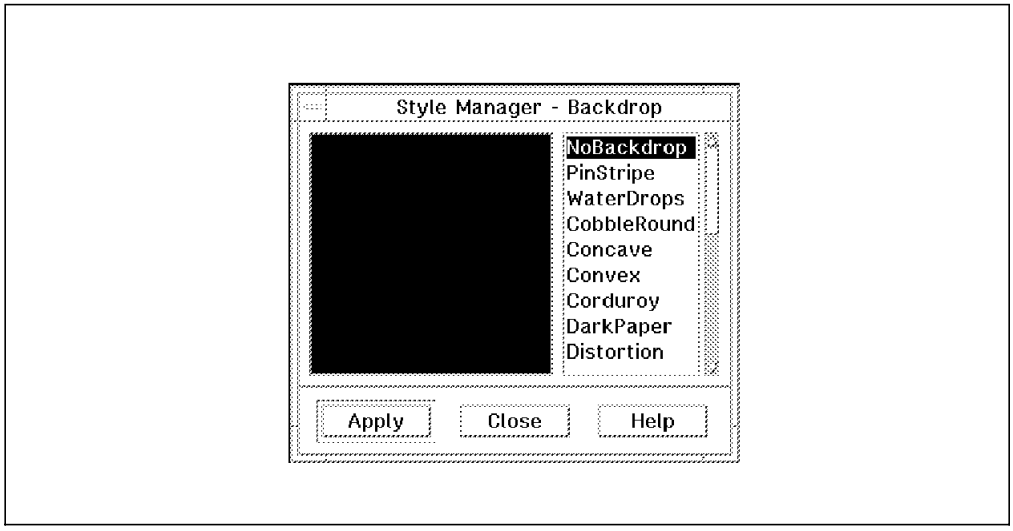


Figure 20. Style Manager Backdrop Dialog

You can add your own backdrops to the desktop. All backdrops must be XPM or XBM format. The desktop icon editor tool can be used to create the correct format backdrops.

To add a custom backdrop to the desktop where all users can use the backdrop:

1. Create the backdrop with a tool that can output either XPM or XBM format
2. Save the backdrop in a file with the extension of .pm or .bm depending on whether the file is an XPM or XBM format respectively
3. Put the .pm or .bm backdrop file in the /etc/dt/backdrops directory

The next time that you log out and then back in you will find your new backdrop available in the backdrop list of the Style Manager's backdrop dialog.

To add a custom backdrop to the desktop that only some users can use. For example, only yourself:

1. Create the backdrop with a tool that can output either XPM or XBM format
2. Save the backdrop in a file with the extension of .pm or .bm depending on whether the file is an XPM or XBM format respectively
3. Put the .pm or .bm backdrop file in a directory of your choice
4. Add or edit the following X resource to include the above directory in the backdrop search path:

   ```
   *backdropDirectories: yourpath
   ```

   Where yourpath is the full path name of the directory in which you stored your custom backdrop. The system default: /usr/dt/backdrops and the system administrator: /etc/dt/backdrops backdrop directories will be automatically appended to this search path. For example, we created an XPM format backdrop called Mybackdrop.pm. We put it in the /u/demo/backdrops directory. We then put the following line in our .Xdefaults file:

   ```
   *backdropDirectories: /u/demo/backdrops
   ```

   Note: Be very careful that the path name specified does not contain any trailing spaces.

The next time that you log out and then back in you will find your new backdrop available in the backdrop list in the Style Manager's backdrop dialog.

You can specify more than one path name on the *backdropDirectories resource as long as you separate the path names with a colon.

5.5 Keyboard Customization

The Style Manager's keyboard customization tool (pictured in Figure 21) allows the user to:

- Turn on and off Auto Repeat. This feature, when turned on, will repeatedly input the value of a held down key on the keyboard until the key is released.
- Change the volume of the keyboard key press feedback feature. By default, the volume is set to zero. As you increase the volume, the keyboard clicker will become louder.

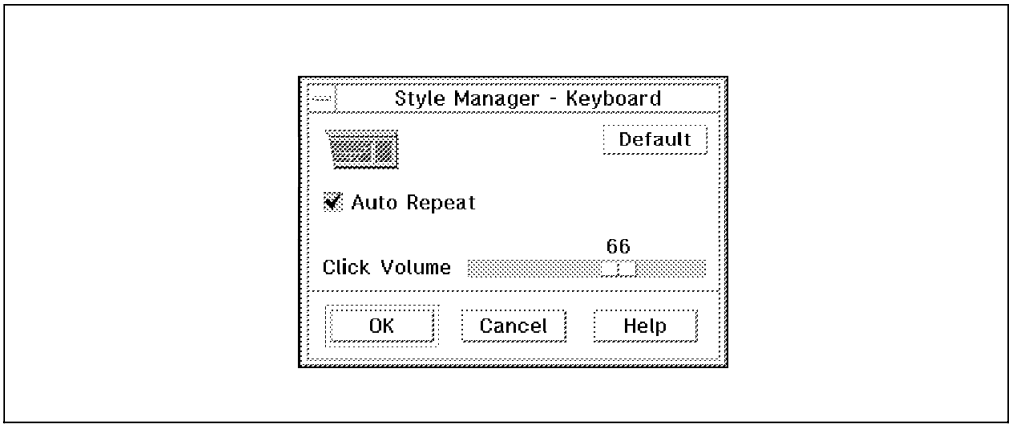


Figure 21. Style Manager Keyboard Dialog

5.6 Mouse Customization

The Style Manager's mouse customization tool (pictured in Figure 22 on page 82) allows the user to:

- Change the handedness of the mouse buttons for either right handed or left handed users.
- Switch the functionality of the middle mouse button to either:

  Transfer Mouse button two is used to drag and drop list or text items. This action must be supported by the application.

  Adjust Mouse button two is used to extend list selections in a multiple-select list or extend the text selection in text fields. With this set, mouse button one is used to drag and drop list and text items.
- Set the speed at which your must click a mouse button twice for the system to sense a double-click. The new speed setting can be tested by double-clicking the mouse picture in the upper-left corner of the dialog box.
- Specify a mouse movement acceleration value which controls how fast the pointer moves across the screen when the mouse is moved fast enough to

switch the pointer into acceleration mode. A setting of two, for example, causes the pointer to move twice as fast as the mouse moves.

- Specify a mouse movement threshold value which controls how fast you must move the mouse before the pointer moves at the above set accelerated speed. The threshold and acceleration adjustments allow the mouse to be used for precise alignment when it is moved slowly, or moved quickly across the screen at the accelerated speed.

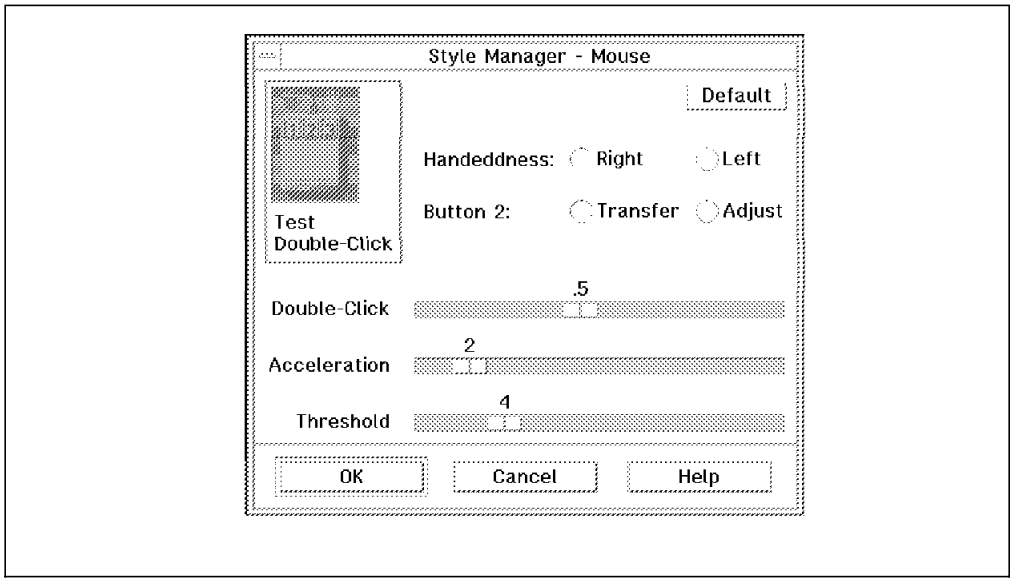


Figure 22. Style Manager Mouse Dialog

5.7 Beep Customization

The Style Manager's beep customization tool (pictured in Figure 23) allows the user to:

- Specify the beep volume in percent of total possible volume. The range is 0 to 100%, where zero is off.
- Specify the frequency or pitch of the system beep, from 82 to 9000 Hertz.
- Specify the duration of the system beep, from .1 to 2.5 seconds.

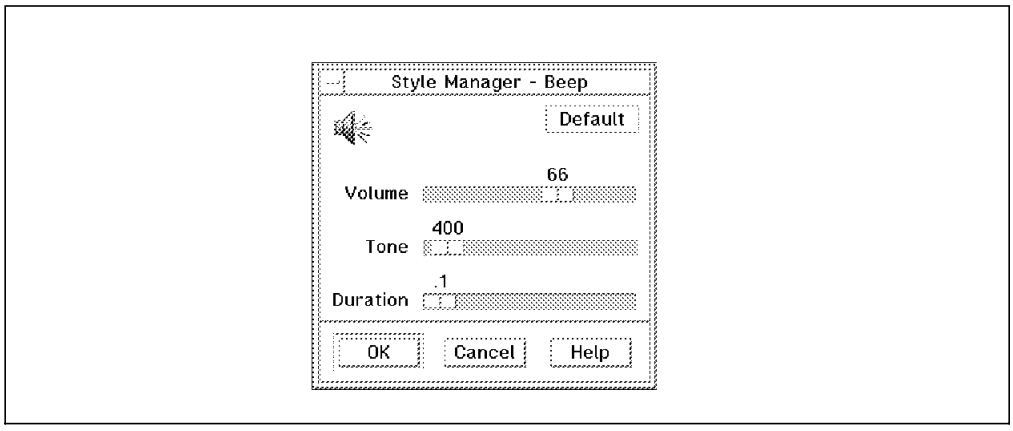


Figure 23. Style Manager Beep Dialog

5.8 Screen Customization

The Style Manager's screen customization tool (pictured in Figure 24) allows the user to:

- Set the characteristics of a screen saver program. The user is able to:
  - Turn on and off the usage of a screen saver program
  - Specify the number of minutes of mouse and keyboard inactivity to wait before invoking a screen saver program
  - Specify one or more screen saver programs to invoke when the mouse and keyboard are inactive for the specified number of minutes
  - Specify the number of minutes each of the selected screen saver programs is to remain active before switching to the next selected screen saver program
  - Preview a screen saver program
- Set the characteristics of the screen lock. The user is able to:
  - Turn on and off the use of a screen saver program while the screen is locked
  - Specify one or more screen saver programs to invoke when the screen is locked
  - Specify the number of minutes each of the selected screen saver programs is to remain active before switching to the next selected screen saver program

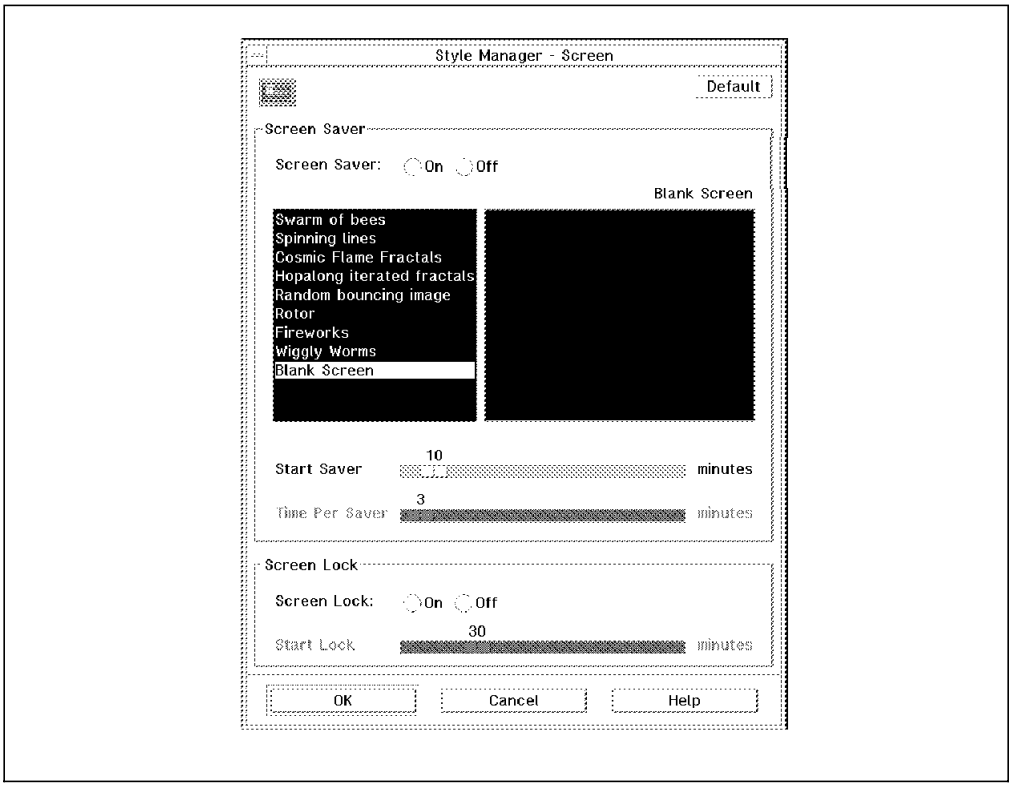


Figure 24. Style Manager Screen Dialog

5.9 Window Customization

The Style Manager's window customization tool (pictured in Figure 25) allows the user to:

- Specify an input focus behavior. The focus can be set to either:

  Focus Follows Mouse
  With this option set, the desktop automatically moves input focus to the window directly underneath the pointer

  Click In Window For Focus
  With this option set, the input focus will only switch to the window directly underneath the pointer when the select mouse button is clicked

- Specify if the window that has input focus is to be automatically raised to the foreground of the screen.
- Specify what a window is to look like when it is moved. A window being moved can either appear as an outline of the window (**Opaque Move** option not selected) or as an image of the window (**Opaque Move** option selected). If you are on an Xstation you may experience a performance degradation if you choose to move windows as images and not outlines.
- Specify if icons for minimized windows are to be placed in an icon box or placed directly on the workspace surface.

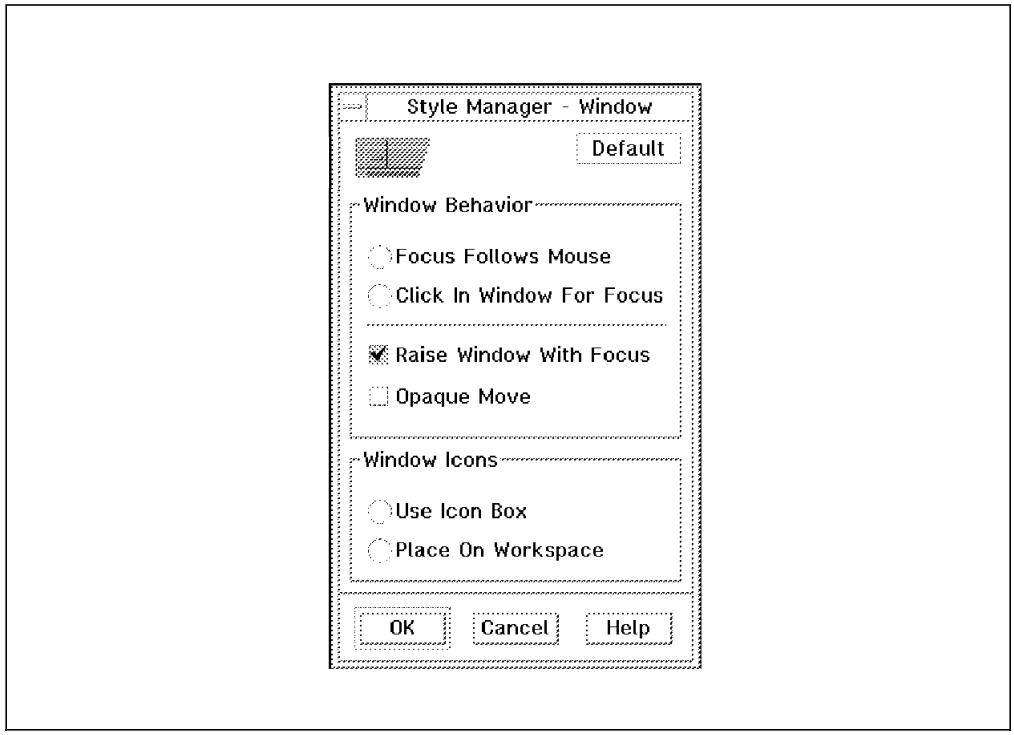


Figure 25. Style Manager Window Dialog

5.10 Login and Logout Customization

The Style Manager's Startup customization tool (pictured in Figure 26) allows the user to:

- Customize which session the Session Manager should start when the user logs in. For more information on the Session Manager and sessions see Chapter 3, "Session Manager" on page 21.
- Turn on and off the logout confirmation dialog. When on, the user must confirm that they want to log out of the desktop before they exit.
- Set the home session used by the Session Manager. For more information on the Session Manager and sessions see Chapter 3, "Session Manager" on page 21.

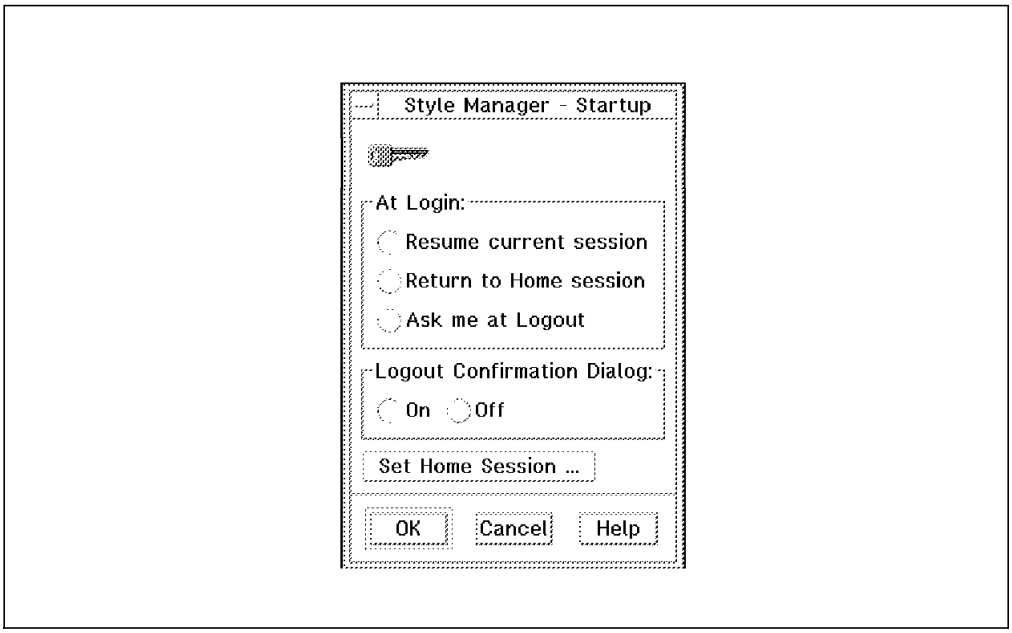


Figure 26. Style Manager Startup Dialog

Chapter 6. Workspace Manager

The Workspace Manager controls how items on the screen look and behave, and how they respond to input from the mouse and/or keyboard. This component of the desktop, known as dtwm, is an X-Window System window manager based on the OSF/Motif window manager, mwm (Version 1.2.3). It provides mwm compatible window management functionality in addition to some new functionality need to support the features of the desktop. This new functionality includes:

- Support for desktop workspaces
- Support for existing OPEN LOOK applications (OPEN LOOK hints are mapped to the nearest Motif behavior)
- Support for the front panel
- Visual enhancements to the frame placed around client windows in order to provide a distinctive look

Many features of the Workspace Manager can be customized by the system administrator and individual users. These include:

- The number of workspaces and their appearance
- The appearance of the windows and the icons in the workspace
- The actions associated with mouse buttons (button bindings)
- The actions associated with particular key depressions (key bindings)
- Workspace and window pop-up or pull-down menu entries

Many of these customizations can be made interactively through the Style Manager. See Chapter 5, “Style Manager” on page 73 for more information. The remaining customizations are made by either modifying X resources or by editing configuration files and are covered in the following sections.

6.1 Customizing Through Configuration Files

Although the Style Manager can be used to interactively customize portions of the Workspace Manager, the majority of the customizations are done through two configuration files:

dtwmrc This file is used to define window and workspace menus along with key and mouse bindings

Dtwm This file is used to contain X resources that modify the look and behavior of the Workspace Manager

If you have previously worked with the X11 environment, perhaps using the Motif Window Manager, mwm, then you will be used to customizing the window manager by making changes to resources in configuration files like the .mwmrc and Mwm. The desktop Workspace Manager does not use these files. Instead, the Workspace Manager uses different configuration files that are very similar in format and content to these files. The reason that the configuration files are different, is similar to the reason that Workspace Manager is different than the Motif Window Manager. In essence, the additional functionality of the desktop dictated that changes had to be made to the configuration files. Additionally,

since the Motif Window Manager is not going away because of the introduction of the desktop and the Workspace Manager, the mwm style configuration files must still be available in case you decide to use the Motif Window Manager in place of the desktop. Because both tools must be able to coexist, a new set of configuration files specific to the desktop was created.

The following sections will describe how to:

- Convert existing mwm configuration files for use by the Workspace Manager
- Define window and workspace menus
- Define keyboard bindings
- Define mouse button bindings

6.1.1 Converting mwm Configuration Files For Use By dtwm

Your existing X11/Motif configuration and resource files can be converted for use with the Workspace Manager. Although it's necessary to change all occurrences of mwm to dtwm and Mwm to Dtwm, in the configuration files, the conversion process is a little more complicated than that. Basically, in addition to the mechanical changes of changing things like mwm to dtwm, it is important to look at configuration files to see just which resources and definitions are being set and make the necessary adjustments when taking into account the changed functionality of the Workspace Manager and the various client changes. For example, to convert an existing .mwmrc menu definition file for use by the Workspace Manager, it is necessary evaluate the clients that are called from the menu definitions to determine if they are still needed or valid under the desktop. An example of a button that can be deleted might be one that brings of a terminal window. Since the front panel has a control which does the same function, a menu button may be redundant and not needed. Additionally, it will be necessary to add menu items that support the new desktop functionality such as workspaces. It is for reasons like this, that the following recommended procedure for migrating your mwm configurations to dtwm configurations starts with the dtwm default definitions and adds your customizations to them.

1. Start with the desktop supplied default configuration files and copy them into a local customization directory: $HOME for personal customizations and /etc/dt for system wide customizations.
   - Menu and key/mouse binding configurations can either be in:

     $HOME/.dt/dtwmrc (Personal customizations)

     /etc/dt/config/$LANG/sys.dtwmrc (System wide customizations)

     /usr/dt/config/$LANG/sys.dtwmrc (System default file)
   - Workspace Manager resources can either be in:

     $HOME/.dt/Dtwm (Personal customizations)

     /etc/dt/app-defaults/$LANG/Dtwm (System wide customizations)

     /usr/dt/app-defaults/$LANG/Dtwm (System default file)
2. Next carefully merge the definitions from your mwm versions of the configuration files into the above customization files. Remember to take into account that some clients and some resources have changed. Use the information in the following sections to better understand how to define menus, along with mouse and key bindings in the desktop versions of the

configuration files. Refer to the dtwm man page information for a listing of the available X resources to customize dtwm.

6.1.2 The Syntax for Definitions Described in the dtwmrc Configuration File

The dtwm configuration specification file is a standard text file with the following syntax:

- Blanks, tabs, and new lines characters are valid word separators
- Blank lines are ignored
- Items or characters can be quoted to avoid special interpretation (for example, 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
- If a line ends with a back-slash, the next line is considered to be a continuation of that line
- All text from an unquoted # (hash, U.S. number symbol) to the end of the line is regarded as a comment and is not interpreted as part of the specification description
- If an exclamation mark (!) is the first character in a line, the line is regarded as a comment

6.1.3 Specifying a Window or Workspace Menu

You can define your own menus to be displayed and from which selections can be made. Examples of the activities that can be initiated are:

- Invoke an action, for details of defining one of these see the details in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128
- Display another menu
- Shuffle the window stack up or down
- Restart the Workspace Manager
- Exit from the desktop
- Refresh the windows in the workspace by repainting the windows
- Toggle between displaying the front panel and minimizing it

The general syntax for a menu specification is shown below:

```
Menu menuname
  {
   label [mnemonic] [accelerator] function [argument]
   label [mnemonic] [accelerator] function [argument]
              .
              .
              .
   label [mnemonic] [accelerator] function [argument]
  }
```

Where:

menuname

A unique name used to reference this menu definition.

label

A descriptive entry that appears in the menu when the menu is displayed. The label may be a character string or a bitmap file. The string encoding for labels must be compatible with the menu font that is used.

When the menu is displayed, you may notice that labels are grayed out and unusable for menu items that:

- Specify the f.nop function
- Specify an invalid function
- Specify a function that doesn't apply in the current context

mnemonic

An optional single character that can be used to select the menu item. The letter must be one of the characters in the descriptive text (label) area. The mnemonic is specified as _character (underline character followed by a character which appears in the label). When the menu line is displayed the first occurrence in the label text that matches the mnemonic character 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 does not execute if any modifier (such as Shift) is pressed with the character key. The mnemonic character is functional only when the menu is posted (displayed) and keyboard traversal applies.

accelerator

An optional keyboard shortcut for the menu button. The accelerator is effective at any time. Whether the menu is displayed or not. However, if the same key press specification is used for an accelerator and in a key binding, the accelerator is effective while the menu is displayed but not otherwise.

function

The function to be carried out if this menu item is selected. The function can be any one of the following that are listed as being valid for a menu. The context column in Table 9 is used exclusively for keyboard and mouse button bindings.

Table 9 (Page 1 of 5). Workspace Manager Functions Supported by dtwm

Function	Description	Context	Valid For
f.action	This function causes the specified action to be invoked by means of the message server.	root, icon, window	button, key, menu
f.beep	This function causes a beep.	root, icon, window	button, key, menu
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 (transient windows) are re-stacked with their associated primary window. Additionally, they 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 the icon argument is specified, then the function applies only to icons. If the window function argument is specified then the function applies only to windows.	root, icon, window	button, key, menu

Table 9 (Page 2 of 5). Workspace Manager Functions Supported by dtwm

Function	Description	Context	Valid For
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 (transient windows) are re-stacked with their associated primary window. If the icon argument is specified then the function applies only to icons. If a window argument is specified then the function applies only to windows.	root, icon, window	button, key, menu
f.create_workspace	This function creates a new workspace. The new workspace name is generated automatically and is of the form ws_nnnn where nnnn is an integer. However, if you have specified an X resource which sets a title to the workspace number being generated that title will be used as the workspace name. For example, if you have predefined the following X resource: Dtwm*ws5*title:Office. The name Office will be used when the workspace is created.	root, icon, window	button, key, menu
f.delete_workspace	This function deletes the current workspace. Windows that reside only in the workspace being deleted will be moved to the next workspace in sequence. If the last workspace in the sequence is being deleted, then windows will be moved to the first workspace. This function will not remove the last workspace - so you'll always be left with one.	root, icon, window	button, key, menu
f.exec command (or ! command)	This function causes command to be executed (using the value of the $MWMSHELL or $SHELL environment variable if set, otherwise /usr/bin/sh). The ! notation can be used in place of **f.exec** as the function name. If the specified command needs to be several words long, or its parameters need a multi-word entry, quotation marks will need to be employed. Usually, both single and double quotes will be needed. This can lead to difficulty, but generally you will find that you get the result you want if you put single (') inside double ("). The example here illustrates a successful specification: f.exec "aixterm -T 'Quick Aixterm' -bg 'cornflower blue' &"	root, icon, window	button, key, menu
f.focus_color	This function sets the colormap focus to a client window. If this function is used in a root context, then the default colormap (setup by the X Window System for the screen where dtwm 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.	root, icon, window	button, key, menu
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.	root, icon, window	button, key, menu
f.goto_workspace workspace	This function causes the workspace manager to switch to the workspace named by the workspace argument. If no workspace exists by the specified name, then no action occurs.	root, icon, window	button, key, menu
f.help [topic [volume]]	This function displays help on the specified topic and volume. If no volume is given, then the workspace manager volume is assumed. If no topic is given, then help on the front panel is shown.	root, icon, window	button, key, menu
f.help_mode	This function causes the workspace manager to enter help mode. In help mode, the pointer changes shape to indicate that the window manager is in control. Any help defined for the control is then shown in a help window.	root, icon, window	button, key, menu
f.kill	This function is used to close application windows. The actual processing that occurs depends on the protocols that the application observes. The application lists the protocols it observes in the WM_PROTOCOLS property on its top level window: • If the application observes the WM_DELETE_WINDOW protocol, it is sent a message that requests the window be deleted. • If the application observes both WM_DELETE_WINDOW and WM_SAVE_YOURSELF, it is sent one message requesting the window be deleted and another message advising it to save its state. • If the application observes only the WM_SAVE_YOURSELF protocol, it is sent a message advising it to save its state. After a delay (specified by the resource quitTimeout), the application's connection to the X server is terminated. • If the application observes neither of these protocols, its connection to the X server is terminated.	icon, window	button, key, menu
f.lower [-client | within | freeFamily]	This function lowers a primary window to the bottom of the global window stack (where it obscures no other window) and lowers the secondary window (transient window or dialog box) within the client family. The arguments to this function are mutually exclusive. The client argument indicates the name or class of a client to lower. The name or class of a client appears in the WM_CLASS property on the client's top-level window. If the client argument is not specified, the context in which the function was invoked indicates the window or icon to lower. The within argument lowers the secondary window within the family (staying above the parent) but does not lower the client family in the global window stack. The freeFamily argument lowers the window to the bottom of the global windows stack from its local family stack.	root, icon, window	button, key, menu

Table 9 (Page 3 of 5). Workspace Manager Functions Supported by dtwm

Function	Description	Context	Valid For
f.marquee_selection	This function is only useful in conjunction with the Common Desktop Environment file manager. It enables selection of file manager objects that have been placed on the root window. It must be bound to a button when used.	root	button
f.maximize	This function causes a client window to be displayed at its maximum size. Refer to the maximumClientSize, maximumMaximumSize, and limit_Resize resources on the dtwm man page for more information.	icon, window (normal)	button, key, menu
f.menu menu_name	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.	root, icon, window	button, key, menu
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.	window	button, key, menu
f.move	This function initiates an interactive move of a client window.	icon, window	button, key, menu
f.next_cmap	This function installs the next colormap in the list of colormaps for the window with the colormap focus.	root, icon, window	button, key, menu
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 argument is specified, then the function applies only to icons. If a window argument is specified, then the function applies only to windows.	root, icon, window	button, key, menu
f.next_workspace	This function causes the workspace manager to switch to the next workspace. If the last workspace is currently active, then this function will switch to the first workspace.	root, icon, window	button, key, menu
f.nop	This function does nothing.	root, icon, window	button, key, menu
f.normalize	This function causes a client window to be displayed at its normal (de-iconified) size. Secondary windows (transient windows) are placed in their normal state along with their associated primary window.	icon, window (maximized)	button, key, menu
f.normalize_and_raise	This function causes a client window to be displayed at its normal size and raised to the top of the window stack. Secondary windows (transient windows) are placed in their normal state along with their associated primary window.	icon, window	button, key, menu
f.occupy_all	This function causes the associated window to be placed in all workspaces.	icon, window	button, key, menu
f.pack_icons	This function is used to re-layout 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.	root, icon, window	button, key, menu
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.	root, icon, window	button, key, menu
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.	root, icon, window	button, key
f.prev_cmap	This function installs the previous colormap in the list of colormaps for the window with the colormap focus.	root, icon, window	button, key, menu

Table 9 (Page 4 of 5). Workspace Manager Functions Supported by dtwm

Function	Description	Context	Valid For
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 the icon argument is specified then the function applies only to icons. If the window argument is specified then the function applies only to windows.	root, icon, window	button, key, menu
f.prev_workspace	This function causes the workspace manager to switch to the previous workspace. If the first workspace is currently active, then this function switches to the last workspace.	root, icon, window	button, key, menu
f.quit_mwm	This function terminates dtwm (but NOT the X window system). At some time, this may be replaced by f.quit_dtwm.	root	button, key, menu (workspace only)
f.raise [client | within | freeFamily]	This function raises a primary window to the top of the global window stack (where it is obscured by no other window) and raises the secondary window (transient window or dialog box) within the client family. The arguments to this function are mutually exclusive. The client argument indicates the name or class of a client to lower. If the client is not specified, the context in which the function was invoked indicates the window or icon to lower. Specifying the within argument raises the secondary window within the family but does not raise the client family in the global window stack. Specifying the freeFamily argument raises the window to the top of its local family stack and raises the family to the top of the global window stack.	root, icon, window	button, key, menu
f.raise_lower [within | freeFamily]	This function raises a primary window to the top of the global window stack if it is partially obscured by another window; otherwise, it lowers the window to the bottom of the window stack. The arguments to this function are mutually exclusive. Specifying the within argument raises a secondary window within the family (staying above the parent window), if it is partially obscured by another window in the application's family; otherwise, it lowers the window to the bottom of the family stack. It has no effect on the global window stacking order. Specifying the freeFamily argument raises the window to the top of its local family stack, if obscured by another window, and raises the family to the top of the global window stack; otherwise, it lowers the window to the bottom of its local family stack and lowers the family to the bottom of the global window stack.	icon, window	button, key, menu
f.refresh	This function causes all windows to be redrawn.	root, icon, window	button, key, menu
f.refresh_win	This function causes a client window to be redrawn.	window	button, key, menu
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 occurs.	root, icon, window	button, key, menu
f.resize	This function initiates an interactive resize of a client window.	window	button, key, menu
f.restart	This function causes dtwm to be restarted (effectively terminated and re-executed). A Restart is necessary for dtwm to incorporate changes in both the dtwmrc file and X resources.	root	button, key, menu (workspace only)
f.restore	This function restores the previous state of an icon's associated window. If a maximized window is iconified, then f.restore restores it to its maximized state. If a normal window is iconified, then f.restore restores it to its normalized state.	icon, window	button, key, menu
f.restore_and_raise	This function restores the previous state of an icon's associated window and raises the window to the top of the window stack. If a maximized window is iconified, then f.restore_and_raise restores it to its maximized state and raises it to the top of the window stack. If a normal window is iconified, then f.restore_and_raise restores it to its normalized state and raises it to the top of the window stack.	icon, window	button, key, menu

Table 9 (Page 5 of 5). Workspace Manager Functions Supported by dtwm

Function	Description	Context	Valid For
f.screen [next | prev | back | screen_number]	This function causes the pointer to warp (be moved) to a specific screen number or to the next, previous, or last visited (back) screen. The arguments to this function are mutually exclusive. The screen_number argument indicates the screen number that the pointer is to be warped. Screens are numbered starting from screen zero. Specifying the next argument causes the pointer to warp to the next managed screen (skipping over any unmanaged screens). Specifying the prev argument causes the pointer to warp to the previous managed screen (skipping over any unmanaged screens). Specifying the back argument causes the pointer to warp to the last visited screen.	root, icon, window	button, key, menu
f.send_msg message_number	This function sends an XClientMessageEvent of type _MOTIF_WM_MESSAGES with message_type set to message_number. The client message is sent only if the specified 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 an f.send_msg of a message that is not included in the client's _MOTIF_WM_MESSAGES property.	icon, window	button, key, menu
f.separator	This function causes a menu separator to be put in the menu pane at the specified location (any label is ignored).	root, icon, window	menu
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). By default, this is bound to a key press of Shift Ctrl Meta <Key>!	root, icon, window	button, key, menu
f.title	This function inserts a title in the menu pane at the specified location.	root, icon, window	menu
f.toggle_frontpanel	If the front panel is in the normal state, this function causes it to be minimized. If the front panel is minimized, this function will change it to the normal state.	root, icon, window	button, key, menu
f.version	This function causes the workspace manager to display its release version in a dialog box.	root, icon, window	button, key, menu
f.workspace_presence	This function displays the workspace presence (Occupy Workspace) dialog box. This dialog allows you to view and set the workspace in which a particular window resides. The root context is disallowed for this function.	window	button, key, menu

arguments

Optional one or more function arguments

6.1.3.1 Example Workspace Menus

The menu pictured in Figure 27 with its definition given below is the default workspace menu which can be found in /usr/dt/config/$LANG/sys.dtwmrc (in this case LANG=C):

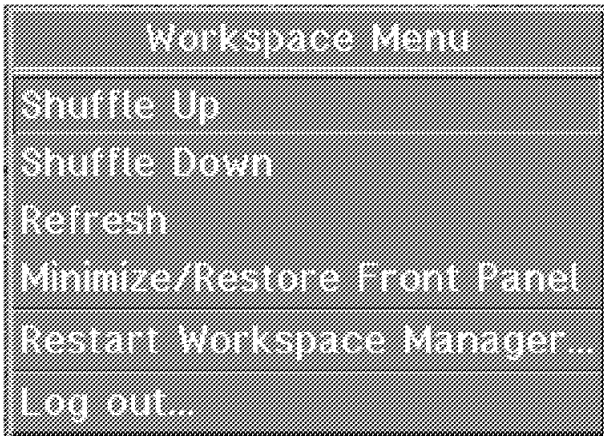


Figure 27. Default Workspace Menu

```
Menu DtRootMenu
{
    "Workspace Menu"                    f.title
    "Shuffle Up"                        f.circle_up
    "Shuffle Down"                      f.circle_down
    "Refresh"                           f.refresh
    "Minimize/Restore Front Panel"      f.toggle_frontpanel
     no-label                           f.separator
    "Restart Workspace Manager..."      f.restart
     no-label                           f.separator
    "Log out..."                        f.action ExitSession
}
```

The menu pictured in Figure 28 with its definition given below is an example of a menu which has been based on the default workspace menu, and customized:

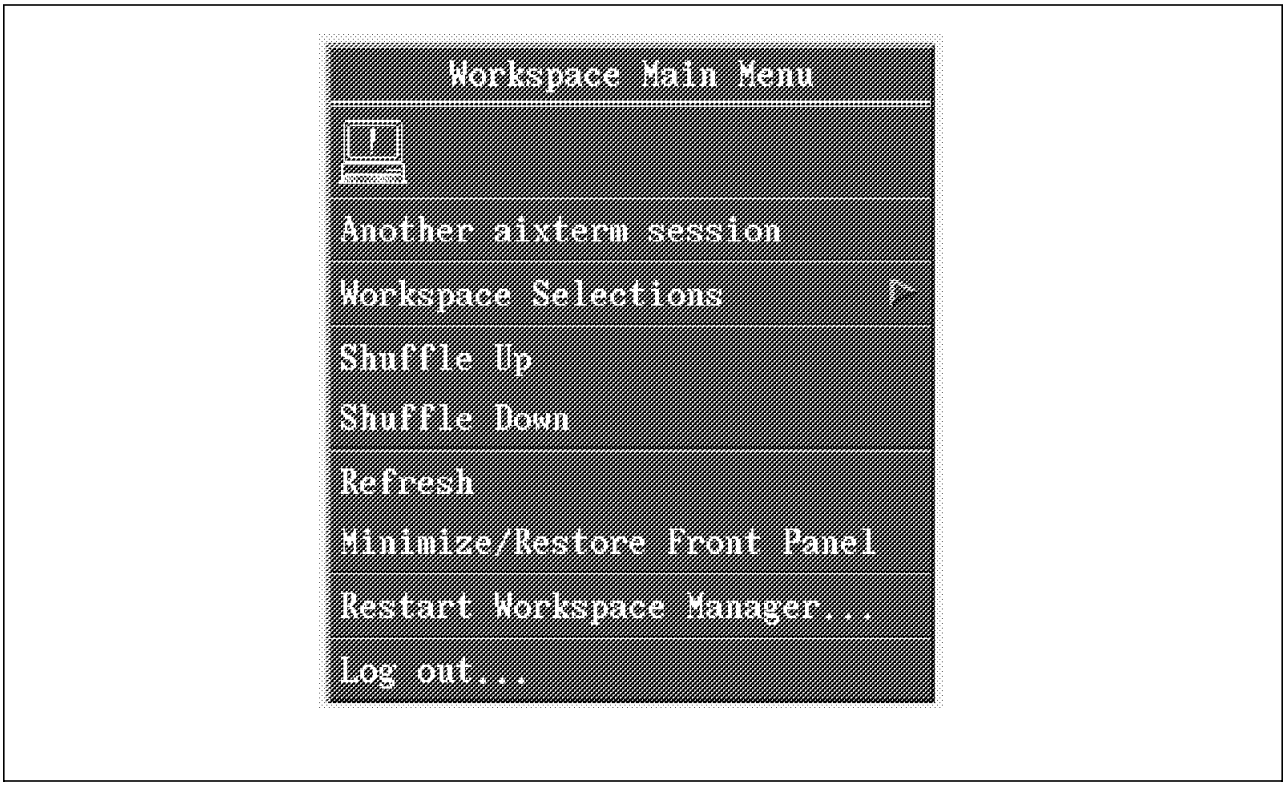


Figure 28. Workspace Menu Generated by the RBRootMenu Definition

```
Menu RBRootMenu
{
    "Workspace Main Menu"               f.title
    @/home/angelo/.dt/icons/Term.bm     f.action Aixterm
    no-label                            f.separator
    "Another aixterm session"           f.action Aixterm
    no-label                            f.separator
    "Workspace Selections"              f.menu WSSelections
    no-label                            f.separator
    "Shuffle Up"                        f.circle_up
    "Shuffle Down"                      f.circle_down
    no-label                            f.separator
    "Refresh"                           f.refresh
    "Minimize/Restore Front Panel"      f.toggle_frontpanel
    no-label                            f.separator
```

```
    "Restart Workspace Manager..."      f.restart
    no-label                            f.separator
    "Log out..."                        f.action EXIT_SESSION
}
```

In this example:

- The title of the menu is defined by:

  ```
  "Workspace Main Menu"                       f.title
  ```

- There are 2 different methods in this example for displaying the selection that results in initiating the Aixterm action. The first method, displays an icon in the menu:

  ```
  @/home/angelo/.dt/icons/Term.bm     f.action Aixterm
  ```

  The icon is defined in the file Term.bm - where bm indicates a bitmap. The @ (at) sign at the beginning of the line indicates that a bitmap is being used. The second method displays the text specified between the two "s (double quote marks):

  ```
  "Another aixterm session"           f.action Aixterm
  ```

- The no-label line draws a horizontal line across the menu to separate one part from another:

  ```
  no-label              f.separator
  ```

  You can use these where you like. They are a good way of signifying physical separation between associated functions in the menu.

- Another menu is to be called from this menu. The menu shown in Figure 29 is an example of this type of sub-menu.

Figure 29. Sub-Menu Accessed From the Workspace Menu

This is achieved by a statement like the following:

```
"Workspace Selections" f.menu WSSelections
```

In support of this, there is also a menu definition like the following for the sub-menu:

```
Menu WSSelections
{
    "Workspace Selection"   f.title
    "Change Workspace"      f.exec "dtksh /u/angelo/CDE/wkspace/dtwstest1"
    no-label                f.separator
    "Previous Workspace"    f.prev_workspace
    "Next Workspace"        f.next_workspace
}
```

The workspace menu with its sub-menu posted and ready for a selection is shown in Figure 30 on page 97.

Figure 30. Root Menu With its Sub-Menu Posted

To illustrate the use of a mnemonic and accelerators, the sub-menu has been modified to include a mnemonic and two accelerators. This can seen in Figure 31 along with its definition below it.

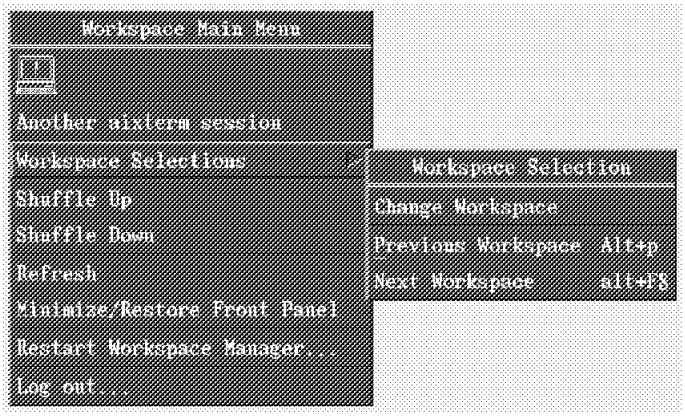


Figure 31. Workspace Menu Plus its Sub-Menu With Mnemonic and Accelerators

```
Menu WSSelections
{
    "Workspace Selection"               f.title
    "Change Workspace"                  f.exec "dtksh /u/angelo/CDE/wkspace/dtwstest1"
    no-label                            f.separator
    "Previous Workspace"  _P  Alt<key>p  f.prev_workspace
    "Next Workspace"          Alt<Key>F8 f.next_workspace
}
```

6.1.4 Defining Keyboard Bindings

A keyboard binding, also known as a key binding, associates a key or combination of keys with Workspace Manager functions or commands to be executed. For example, you want to be able to press the Shift key and Escape key together and display the window menu - that is, to perform the function f.post_wmenu. Key bindings apply across all workspaces. They are not workspace specific.

Key bindings are defined using the following syntax in the dtwmrc file along with the workspace and window menus.

```
Keys GroupName
{
 Modifier(s)<Key>key_name function_context function arguments
 Modifier(s)<Key>key_name function_context function arguments
   .
   .
   .
 Modifier(s)<Key>key_name function_context function arguments
}
```

Where:

GroupName

A unique name for the key binding group being defined.

Modifiers(s)

Optionally one or more of the following modifier keys (separated by a space) can be specified as part of the key press combination.

Ctrl The Control key

Shift The Shift key

Meta The Extend, Meta or Alt key

Alt The Extend, Meta or Alt key

Lock The Lock key

Mod1 Modifier1

Mod2 Modifier2

Mod3 Modifier3

Mod4 Modifier4

Mod5 Modifier5

Note: The Alt and Meta keys can be used interchangeably. Not all keyboards will have all of the listed modifier keys.

key_name

This is the specification of the key that is to be included in the key press combination. Keys with letters, numbers, or names on the key cap are specified by what is on the key cap. For example, the key with an a on it is specified a. Similarly, the 2 key is 2, the Tab key is Tab and the F3 is F3. Keys that do not have letters, numbers or names on the key cap, but have symbols, are specified by spelling out the name of the symbol. For example, the + key is specified plus. Special key combinations, keypad keys, and non-ASCII keys must be spelled out. There is a list of key names in a header (.h) file keysymdef.h in the /usr/include/X11 directory. This directory is

available when the X-Windows development environment has been installed. The entries in keysymdef.h will precede the name of a key with XK_. In the binding definition, the name of the key is specified without the XK_ prefix. For example, the : key is shown as XK_colon. In the key binding definition it is specified as colon.

function_context

Constrains the operation of the key binding to function only when the pointer is positioned in, at or on the specified context. More than one of the following contexts can be specified by separating them by the vertical bar symbol.

root Any part of the backdrop of the workspace. The space must not be occupied by a client window and its frame, an icon, or the front panel. This is known in the Motif world as the root window.

window Any part of the Client window.

icon Any icon representing a minimized Client window.

function

The Workspace Manager function that is to be executed when the specified keyboard combination is pressed. A list of valid functions can be found in Table 9 on page 90. Only those functions that are listed as being valid for key may be used.

arguments

Optional function arguments.

There can only be one group of key bindings active at any given time. The X resource Dtwm*keyBindings is used to specify which of the defined key binding groups are to be active. For example:

```
Dtwm*keyBindings:      RBKeyBindings
```

Activates the key binding group defined with the name RBKeyBindings.

6.1.4.1 Example Set of Key Bindings

The example below is the default set of key bindings taken from /usr/dt/config/$LANG/sys.dtwmrc (in this case, LANG=C):

```
Keys DtKeyBindings
{
#   Alt<Key>Menu                root|icon|window        f.toggle_frontpanel

    Shift<Key>Escape            icon|window             f.post_wmenu
    Alt<Key>space               icon|window             f.post_wmenu
    Alt<Key>Tab                 root|icon|window        f.next_key
    Alt Shift<Key>Tab           root|icon|window        f.prev_key
    Alt<Key>Escape              root|icon|window        f.next_key
    Alt Shift<Key>Escape        root|icon|window        f.prev_key
    Alt<Key>Down                root|icon|window        f.circle_down
    Alt<Key>Up                  root|icon|window        f.circle_up
    Alt Ctrl Shift<Key>exclam   root|icon|window        f.set_behavior
    Alt<Key>F6                  window                  f.next_key transient
    <Key>SunFront               icon|window             f.raise_lower
    <Key>SunOpen                window                  f.minimize
    <Key>SunOpen                icon                    f.normalize
}
```

The following example is a customized personal set of key bindings stored in $HOME/.dt/dtwmrc..

```
Keys RBKeyBindings
{

  Shift<Key>Escape       icon|window     f.post_wmenu
  Meta<Key>space         icon|window     f.post_wmenu
  Meta<Key>Escape        root|icon|window f.next_key
  Meta Shift<Key>Escape  root|icon|window f.prev_key
  Meta<Key>Down          root|icon|window f.circle_down
  Meta<Key>Up            root|icon|window f.circle_up
  Meta<Key>0             root|icon|window f.exec "dtksh /u/rogerb/Dt/dtwx1 ws0"

}
```

6.1.5 Defining Mouse Button Bindings

Just as you can assign particular functions/actions to key combinations, so too can you for mouse buttons. For example, you might wish to be able to bring up an aixterm terminal window when a particular mouse button is pressed while the pointer is on, say, the backdrop.

A Button Binding is an association of a Mouse button operation, possibly in conjunction with a keyboard modifier key (for example Ctrl, Shift, Alt), with a Workspace (or window) Manager function.

Button bindings are defined using the following syntax in the dtwmrc file along with the workspace/window menus and key bindings.

```
Buttons GroupName
{
 Modifier(s)<button&state> function_context function arguments
 Modifier(s)<button&state> function_context function arguments
    .
    .
    .
 Modifier(s)<button&state> function_context function arguments
}
```

Where:

GroupName

A unique name for the button binding group being defined.

Modifiers(s)

Optionally one or more of the following modifier (separated by a space) keys can be specified as part of the key and button press combination.

Ctrl The Control key

Shift The Shift key

Meta The Extend, Meta or Alt key

Alt The Extend, Meta or Alt key

Lock The Lock key

Mod1 Modifier1

Mod2 Modifier2

Mod3 Modifier3

Mod4 Modifier4

Mod5 Modifier5

Note: The Alt and Meta keys can be used interchangeably. Not all keyboards will have all of the listed modifier keys.

button&state

This is the specification of the mouse button and its state that is to be included with the optional modifier(s) to initiate the function.

The following abbreviations are used in the button&state table.

Btn1 The left button

Btn2 The middle button on a 3-button mouse; the right button on a 2-button mouse

Btn3 The right button on a 3-button mouse; both buttons on a 2-button mouse

Btn4 Buttons 1 and 2 together on a 3-button mouse; not valid for a 2-button mouse

Btn5 Buttons 2 and 3 together on a 3-button mouse; not valid for a 2-button mouse

Down Holding down a button

Up Releasing a button

Click Pressing and releasing a button; also referred to as the Extend character key

Click2 Pressing and releasing a button twice in rapid succession (double-click). The Style Manager′s mouse customization tool can be used to set the double click speed. For more information see 5.6, “Mouse Customization” on page 81

The following button&state values are valid:

Table 10 (Page 1 of 2). Valid button&state values

button&state value	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

Table 10 (Page 2 of 2). Valid button&state values	
button&state value	**Description**
Btn5Down	Button 5 Press
Btn5Up	Button 5 Release
Btn5Click	Button 5 Press and Release
Btn5Click2	Button 5 Double Click

function_context

Constrains the operation of the key binding to function only when the pointer is positioned in, at or on the specified context. More than one of the following contexts can be specified by separating them by the vertical bar symbol.

root Any part of the backdrop of the workspace. The space must not be occupied by a client window and its frame, an icon, or the front panel. This is known in the Motif world as the root window

window Any part of the client window

icon Any icon representing a minimized client window

title The title area of the window frame

app The client window (excluding the window management frame)

function

The Workspace Manager function that is to be executed when the specified mouse button and optional modifier key combination is pressed. A list of valid functions can be found in Table 9 on page 90. Only those functions that are listed as being valid for button may be used.

arguments

Optional function arguments

There can only be one group of mouse button bindings active at any given time. The X resource Dtwm*buttonBindings is used to specify which of the defined button binding groups is to be active. For example the following line activates the button binding group defined with the name MYButtonBindings:

```
Dtwm*buttonBindings:      MYButtonBindings
```

6.1.5.1 Example Set of Mouse Button Bindings

The example below is the default set of mouse button bindings taken from /usr/dt/config/$LANG/sys.dtwmrc (in this case, LANG=C):

```
Buttons DefaultButtonBindings
{
    <Btn1Down>          frame|icon              f.raise
    <Btn3Down>          icon                    f.post_wmenu
    Mod5<Btn1Down>      root                    f.marquee_selection
    Mod5<Btn3Down>      root                    f.menu  DtRootMenu
    Mod5<Btn1Down>      icon|frame|window       f.focus_key
    Mod5<Btn3Down>      icon|frame              f.post_wmenu
}
```

The following is an example of a customized personal set of mouse button bindings stored in $HOME/.dt/dtwmrc..

```
Buttons MYButtonBindings
{
    <Btn1Down>      root             f.menu  RBRootMenu
    <Btn3Down>      root             f.menu  DtRootMenu
    <Btn1Down>      frame|icon        f.raise
    <Btn3Down>      frame|icon        f.post_wmenu
    Meta<Btn1Down>  icon|window       f.move
    Meta<Btn3Down>  window           f.minimize
}
```

6.1.6 Configuration Error and Warning Messages

Errors that occur during the processing of the configuration files are recorded in the $HOME/.dt/errorlog. file.

Be sure to check this file if the appearance or behavior of dtwm is not what you expect. The following is an example of some of the error messages that you might find in the errorlog:

```
Mon Oct 03 09:15:02 1994
dtwm: The action definition "PrinterInfonn" in the file
  "cde:/usr/dt/appconfig/types/en_US/printerNN.dt"
has the illegal value "OUTPUT_ONLY" in the "WINDOW_TYPE" field.

Mon Oct 03 09:15:04 1994
dtwm: The action definition "PrinterInfonn" in the file
  "cde:/usr/dt/appconfig/types/C/printerNN.dt"
has the illegal value "OUTPUT_ONLY" in the "WINDOW_TYPE" field.

Mon Oct 03 09:15:05 1994
Workspace Manager: Key bindings MyNewKeyBindings not found, using built
in key bindings
```

6.2 Moving, Copying and Removing Windows From and Between Workspaces

You might need to move a window from its present workspace to another, copy it to other workspaces or remove it from a workspace. All of these things can be achieved by either:

- Interactively using options on the window menu
- Assigning an appropriate Workspace Manager function to a key and/or mouse button binding

Note: Copying a window into a workspace means the window will simply show up in another workspace. Another copy of the application is not started as a result of the copy. Similarly, removing a window from a workspace simply prevents it from being displayed in a particular workspace. The application is not terminated as a result of the removal.

For more information on the Workspace Manager Functions see Table 9 on page 90.

The following sections will describe how to:

- Interactively move and copy windows between workspaces
- Interactively copy a window into all workspaces
- Interactively remove a window from a workspace

- Move or copy windows between workspaces with key or button bindings
- Copy a window into all workspaces with key or button bindings
- Remove a window from a workspace with key or button bindings

6.2.1 Interactively Moving or Copying a Window Into Another Workspace

There are two aspects to this:

- You may want to move the window from where it is now into another workspace and remove it from its current workspace
- You may want to keep the window in the workspace it is now, and copy it into other workspace(s)

The procedure to doing either aspect is basically the same:

1. Start by displaying the window menu for the window that you want to move or copy into another workspace. The window menu is displayed by clicking on the left mouse button while the pointer is over the button at the top left corner of the window frame. The following menu will be displayed.

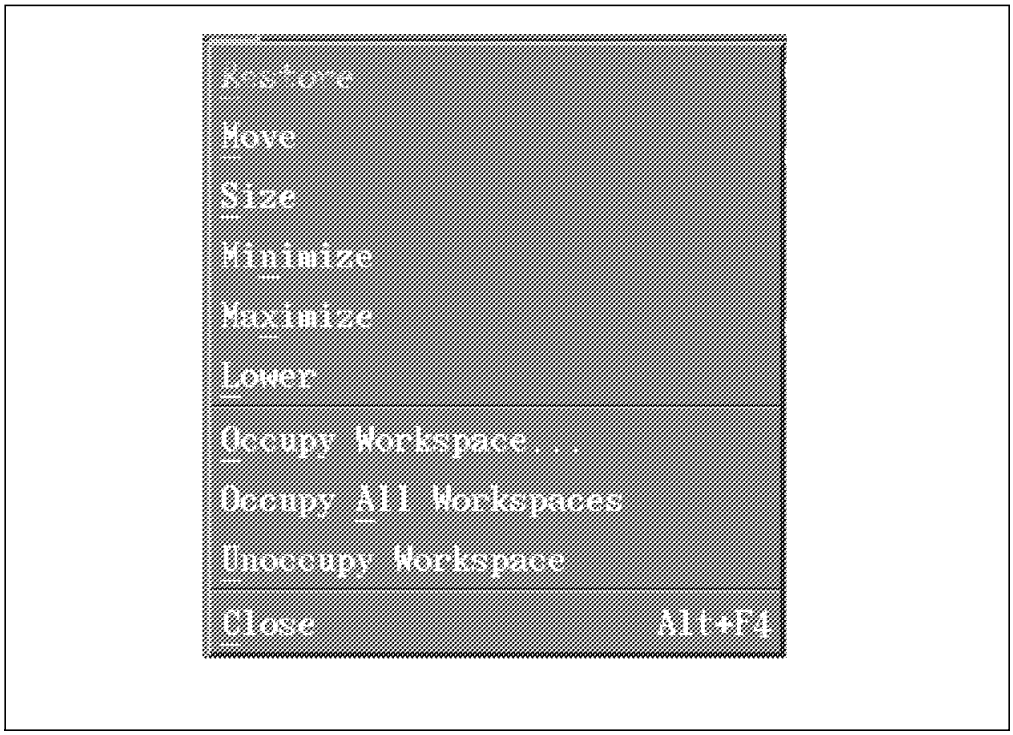


Figure 32. Window Menu

2. Select the **Occupy Workspace...** option. The following dialog will be displayed.

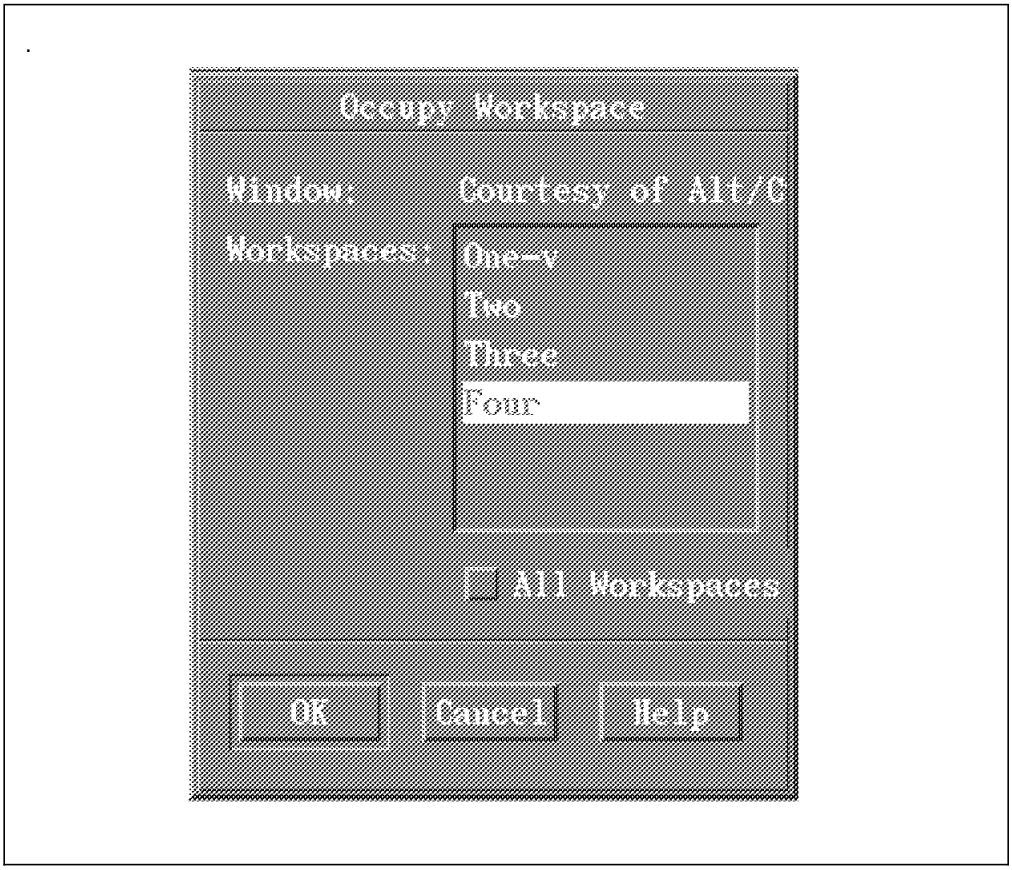


Figure 33. Occupy Workspace Dialog

3. The window in the Occupy Workspace dialog lists all of the available workspaces. The workspaces which are highlighted are the ones that currently contain the window you are trying to move or copy. From this list, you can:
   - Move the window into another workspace by:
     a. Putting the mouse pointer on the name of the workspace you want to move the window into.
     b. Click with the left button so that a workspace name is highlighted as in the window shown above. Notice that highlighting it removed it from all the workspaces except the one you clicked on.
     c. Select **OK** to activate the window move. Otherwise select **Cancel** to cancel the window move.
   - Copy the window into another workspace by:
     a. Putting the mouse pointer on the name of a workspace in which you want the window.
     b. Press and hold the **Ctrl** key then click the left mouse button to highlight the new desired workspace name.
     c. Repeat the above two until all of the desired workspaces are highlighted.
     d. Select **OK** to activate the window copy. Otherwise select **Cancel** to cancel the window copy.

6.2.2 Interactively Copying a Window Into All Workspaces

To interactively copy a window into all workspaces:

1. Start by displaying the window menu for the window that you want to copy into all workspaces. The window menu is displayed by clicking on the left mouse button while the pointer is over the button at the top left corner of the window frame. The menu shown in Figure 32 on page 104 will be displayed.
2. Select the **Occupy All Workspaces** option to copy the window into all workspaces.

6.2.3 Interactively Removing a Window From a Workspace

A window can be removed from any workspace as long as it exists in more than one workspace. You are not allowed to remove a window that exists in only one workspace. To interactively remove a window from a workspace:

1. Use the workspace switch to move to the workspace from which you want to remove the window.
2. Display the window menu for the window that you want to remove from the workspace. The window menu is displayed by clicking on the left mouse button while the pointer is over the button at the top left corner of the window frame. The menu shown in Figure 32 on page 104 will be displayed.
3. Select the **Unoccupy Workspace** option to remove the window from the workspace. If the option is grayed out, this means that the window only exists in the current workspace. Remember removing is not allowed.
4. Repeat the above three steps to remove the workspace from any other desired workspaces.

6.2.4 Moving or Copying Windows Between Workspaces With Key or Button Bindings

You use the Workspace Manager function `f.workspace_presence` to move or copy a window between workspaces. This is a function when invoked, is applied to the window currently in focus. The procedure shown in 6.1.4, “Defining Keyboard Bindings” on page 98 or 6.1.5, “Defining Mouse Button Bindings” on page 100 can be used to define a keyboard or mouse button binding that calls this Workspace Manager function. This function, when called, brings up the Occupy Workspace dialog shown in Figure 33 on page 105. Step 3 of the procedure in 6.2.1, “Interactively Moving or Copying a Window Into Another Workspace” on page 104 can be used to move or copy the focused window between the workspaces.

6.2.5 Copying a Window Into All Workspaces With Key or Button Bindings

You use the Workspace Manager function `f.occupy_all` to copy a window into all workspaces. This is a function when invoked, is applied to the window currently in focus. The procedure shown in 6.1.4, “Defining Keyboard Bindings” on page 98 or 6.1.5, “Defining Mouse Button Bindings” on page 100 can be used to define a keyboard or mouse button binding that calls this Workspace Manager function. This function, when called, will copy the focused window into all workspaces.

6.2.6 Removing a Window From a Workspace With Key or Button Bindings

You use the Workspace Manager function `f.remove` to remove a window from a workspace. This is a function when invoked is applied to the window currently in focus in the active workspace. The window can be removed from any workspace as long as it exists in more than one workspace. You are not allowed to remove a window that exists in only one workspace. The procedure shown in 6.1.4, “Defining Keyboard Bindings” on page 98 or 6.1.5, “Defining Mouse Button Bindings” on page 100 can be used to define a keyboard or mouse button binding that calls this Workspace Manager function. This function, when called, will remove the focused window from the active workspace.

6.3 Directing an X Window Client to a Specific Workspace

At present it is not possible to initiate an X Window client and direct it to display in a particular workspace. The client will be displayed in which ever workspace is active when the application finishes initializing. However, you are able to move the window from its present workspace to another, or copy it to other workspaces - as described in 6.2, “Moving, Copying and Removing Windows From and Between Workspaces” on page 103.

6.4 Switching Workspaces Using the Keyboard

You are able to switch workspaces using the keyboard if you create some customized key bindings to associate the appropriate Workspace Manager function to a particular key press combination. For more information on defining key bindings see 6.1.4, “Defining Keyboard Bindings” on page 98.

The Workspace Manager functions that are directly concerned with switching workspaces are:

f.goto_workspace `workspace`

This function causes the workspace manager to switch to the workspace named by the `workspace` argument.

If no workspace exists by the specified name, then no action occurs.

A workspace name is of the form wsn where n is a number. Remember that the text on the workspace button on the front panel is the workspace′s title, not its name.

Note that adding and deleting workspaces dynamically can affect this function.

f.next_workspace

This function causes the workspace manager to switch to the next workspace.

If the last workspace in the list of workspaces is currently active, then this function will switch you to the first workspace in the list.

f.prev_workspace

This function causes the workspace manager to switch to the previous workspace.

If the first workspace in the list of workspaces is the active one, then this function switches to the last workspace in the list.

6.4.1 Example Key Bindings That Switch Workspaces

Below is an example of some key bindings that switch workspaces.

```
Keys RBKeyBindings
{

  ...
  ...

  Meta <Key> Left         root|icon|window    f.prev_workspace
  Meta <Key> Right        root|icon|window    f.next_workspace

  Meta Ctrl <Key> 0       root|icon|window    f.goto_workspace ws0
  Meta Ctrl <Key> 1       root|icon|window    f.goto_workspace ws1
  Meta Ctrl <Key> 2       root|icon|window    f.goto_workspace ws2
  Meta Ctrl <Key> 3       root|icon|window    f.goto_workspace ws3

  ...
  ...

}
```

6.5 Using a Multiple Screen Display With the Desktop

The desktop supports multiple screen displays by default. If you have more than one display adapter in your system, the desktop will automatically configure itself to support use the other display. However, the following restrictions apply:

- You will not be able to drag anything from one screen to another.
- The front panel will only exist on the first screen.
- X Window clients can be started on the other screens as long as their display resource is set appropriately. For example:

  ```
  aixterm -display starlight:0.1
  ```

  Will start an aixterm on the second screen of the display named starlight.

6.6 Starting the Workspace Manager

Normally, the Session Manager automatically starts up dtwm as the Workspace Manager. You can alter the command line that is used to start the window manager via the wmStartupCommand resource. See on page 24 for more about the wmStartupCommand resource.

Chapter 7. File Manager

The File Manager is a tool to navigate through the filesystems and manipulate files and directories in a user friendly way. By pushing the appropriate control in the front panel (see Figure 34), the File Manager will be invoked and a graphical interface (window) of the user's home directory will be represented (see Figure 34).

Each File Manager interface shows the contents of a directory. In this interface, files and directories have an iconic representation. This representation of files, directories, actions, simply all UNIX files, are called objects in the desktop. For more information about objects see 7.1, "Object" on page 110.

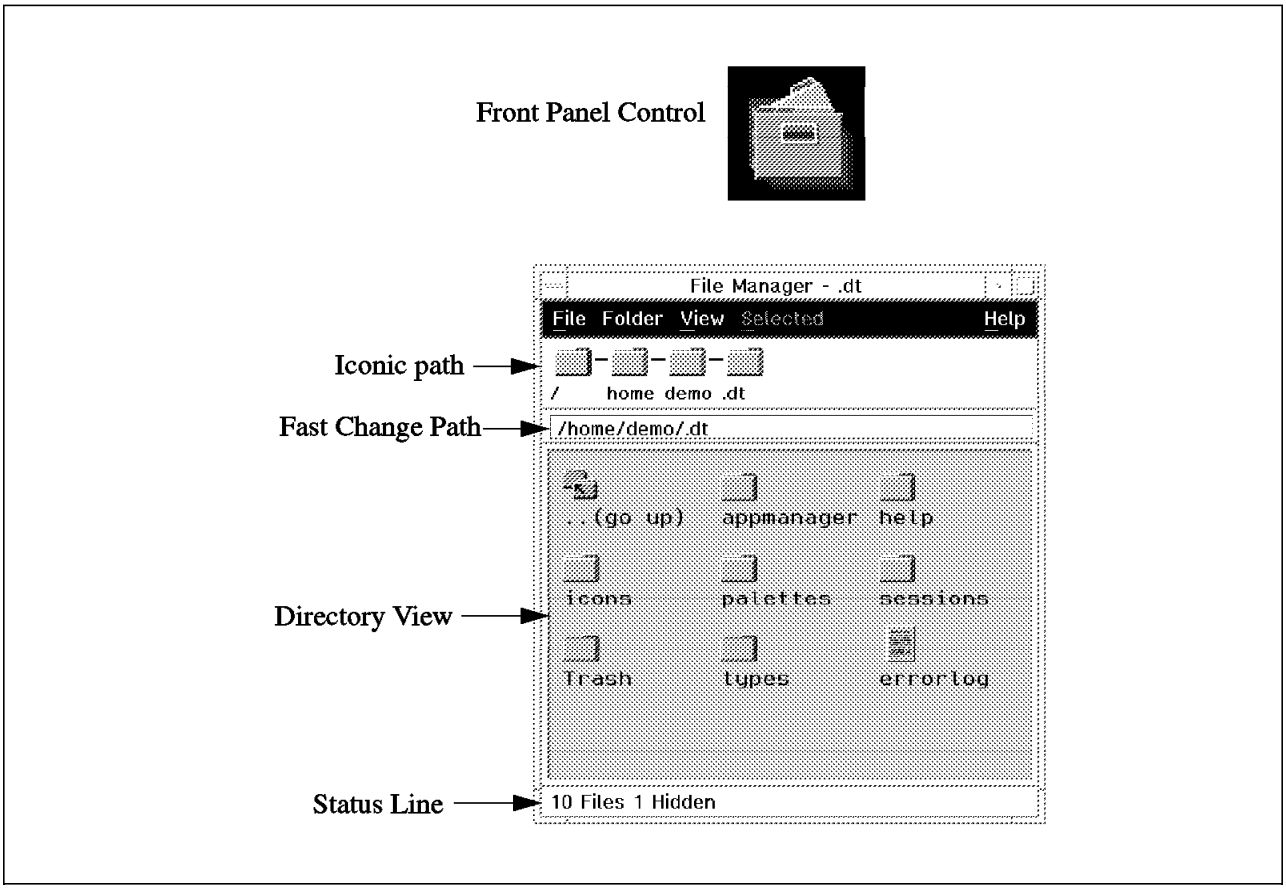


Figure 34. File Manager Front Panel Control and the File Manager Interface

With the File Manager you can interactively perform the following functions:

- View different directories
- View the contents of directories by different representations
- Move, copy and link files
- Rename files
- Delete files
- Execute executable files
- Edit files
- Search for files

7.1 Object

The desktop is organized around objects. An object consists of:

- An icon which is used to visually represent the object
- A set of predefined actions that define the behavior of the object
- A title or a name that identifies the object

The object concept is used to represent all entities in the filesystem. These entities can be:

- A data file
- An executable file
- A directory
- A file that represents a desktop action

In essence, an entity in the filesystem that is represented by an icon, has a predefined set of actions and has an associated title is an object. For example, your home directory is a object because, it is represented in the File Manager by a folder icon. It has some predefined actions associated with it, specifically OpenInPlace and OpenNewView. And finally, it has a title. In the File Manager, your home directory has the name of the directory displayed along with its folder icon.

In fact, every entity in the filesystem is represented by an object. The icons, actions and titles associated to each entity have either been specified by you or your system administrator or most likely defaulted based on the type of entity. Back to the home directory example, it is unlikely that you or your system administrator have established a type definition which describes the icon and actions available for your home directory. Consequently, the system supplied default type definition is used to make an object out of your home directory.

This process of classifying entities by their type is called data typing. The desktop is delivered with many predefined data types which classify several different types of entities into objects. In addition to the type definition used for your home directory, the default desktop contains a data type definition which specifies that all data files with an extension of .c have the same icon (pictured in Figure 35 on page 111) and have three actions associated with them:

- Open - Open the file in the editor
- Make - Compile and link the file

- Print - Send the file to the printer

The titles of the .c files use the names of the files themselves. We now have the required things that make up an object (filesystem entity, icon actions and title). Therefore the data type definition for .c files classifies all .c files as objects.

For more information on data typing see 8.3, “Data Types” on page 124 For more information on actions see 8.2, “Actions” on page 122

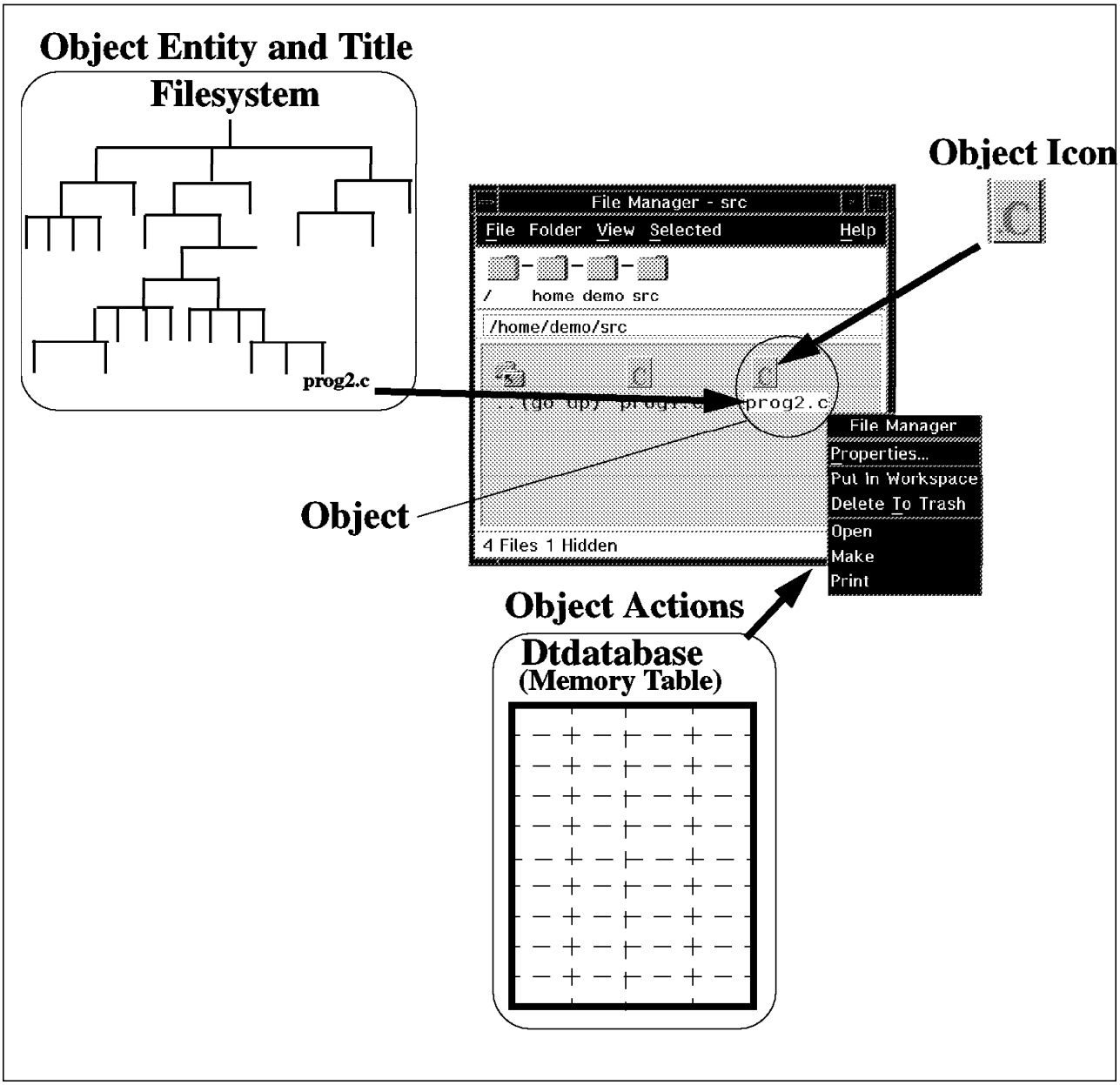


Figure 35. Different Parts of an Object

7.2 Interactively Customizing the File Manager

The File Manager provides an interactive tool that can be used to customize the way it presents the filesystem information. To access this tool, select **Set Preferences** from the View menu bar option.

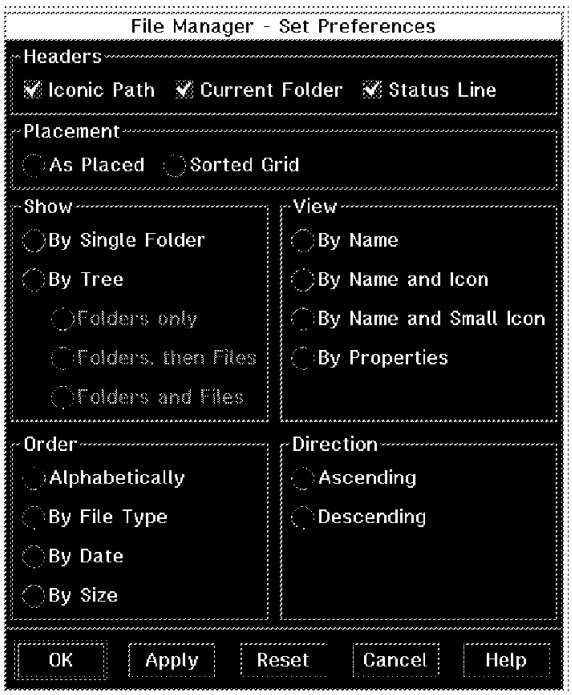


Figure 36. File Manager Preferences

As shown in Figure 36 this tool has six different sections: Headers, Placement, Show, View, Order, Direction.

Headers:

- Iconic Path: This will enable the iconic path.
- Current Folder: This will enable the area for the fast change path.
- Status Line: If status line is enabled the total number of hidden files and, if **Show** is set to **Single Folder**, the number of files will be shown here.

By default all three items are enabled.

Placement: Changes the objects placement criteria

- As Placed: This will show the objects in a single column.
- Sorted Grid: This will only be effective as long as the **Show** item is set to **Single Folder**. The objects will be shown in as many columns as fit in the interface. Figure 34 on page 109 shows an example of the File Manager with the preferences Placement: Sorted Grid and Show: By Single Folder.

Show: Changes the type of view

- By Single Directory:(default) This will show one directory per interface only.

- By Directory Tree: Presents the filesystem as a tree structure. The tree representation can further be customized by displaying:
  - Directories only
  - Directories then files
  - Directories and files

  Figure 37 shows an example of the File Manager directory tree representation.

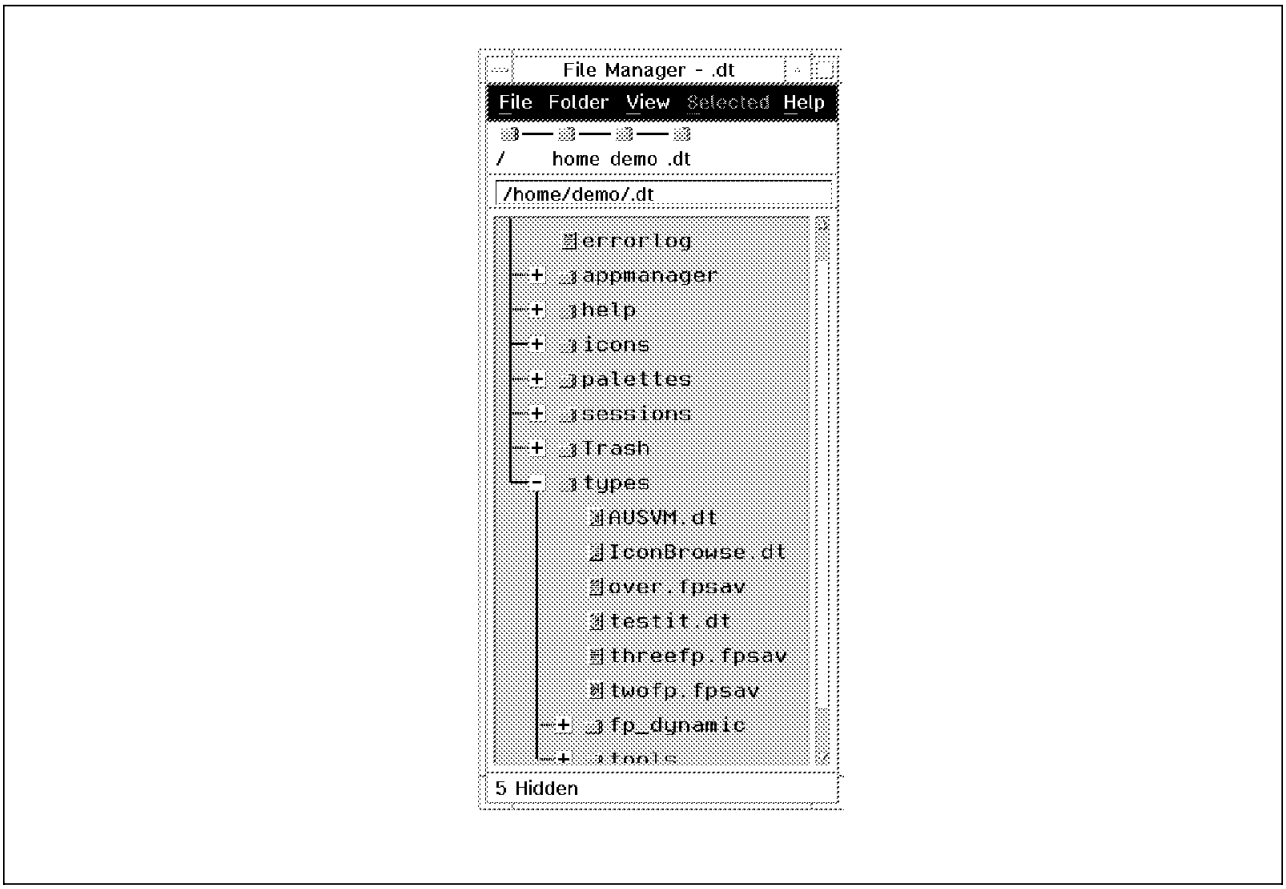


Figure 37. Directory Tree Structure. This Figure shows the File Manager preferences Show: By Tree and Directories then Files.

View: Shows the content of the directory in different fashions

- By Name: No icons will be displayed.
- By Name and Icon: This is the default for the view of a single directory. An example of this selection is shown in Figure 34 on page 109.
- By Name and Small Icon: This is the default for the view by tree. See Figure 37 for an example of this selection.
- By Properties: This is similar to an output from the AIX `ls -l` command combined with names and small icons. Figure 38 on page 114 shows you an example of this selection.

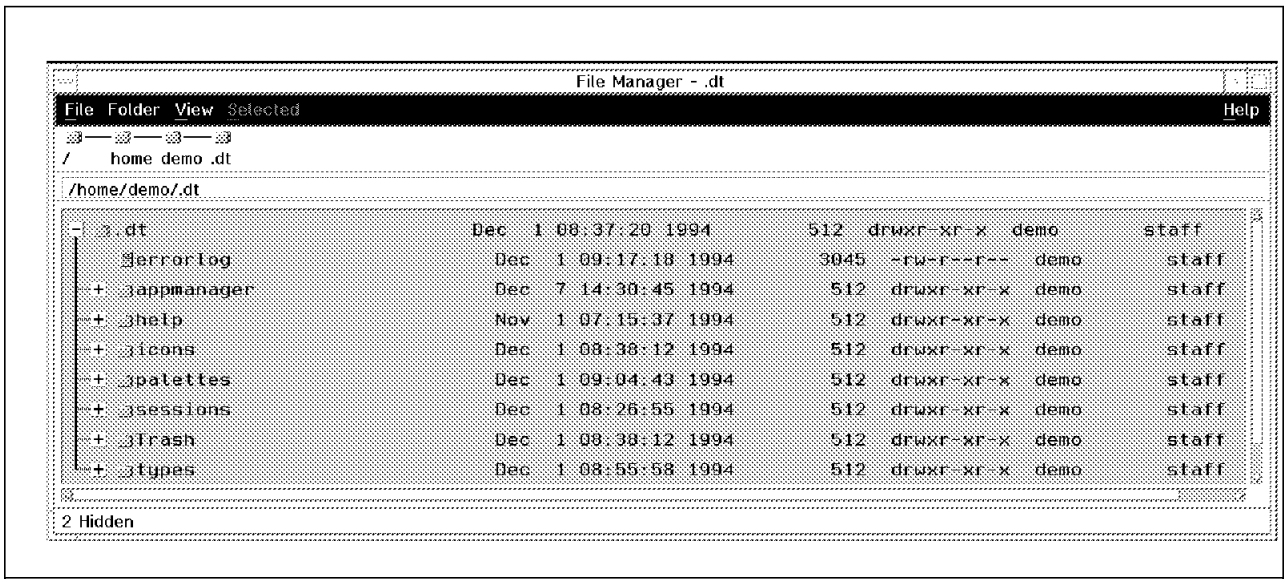


Figure 38. File Manager Interface. The Preferences are set to: Placement: As Placed; Show: By Tree with Folder and Files; View: By Properties

Order: Sorts the contents of the directory in different fashions

- Alphabetically (default)
- By File Type
- By Date: the date of last update
- By Size

Direction: Changes the method of ordering

- Ascending (default)
- Descending

7.3 Manipulating the Filesystem With the File Manager

The following sections will illustrate how to manipulate the entities in the filesystem with the File Manager.

7.3.1 Viewing and Changing the File Properties

There are two different ways to see the file properties:

1. You can see the file properties in a similar way as the `ls -l` command will show you. To do this, the **By Properties** option has to be set in the Set Preferences Interface. For more information about this option see 7.2, “Interactively Customizing the File Manager” on page 112.
2. Move the mouse cursor in the File Manager above the object whose properties you want to see. Press and hold the right mouse button. A menu will pop up. Select **Properties** from the menu and an interface displaying the file properties will be shown. As long as you have write permission to the file, you will be able to change the file properties by selecting the appropriate buttons. The file permissions are set if the check symbol is present inside the button.

7.3.2 Viewing Hidden Files

To see the hidden files you have click on **Show Hidden Files** in the View menu bar option or you can use the keyboard accelerator Ctrl-s while in a directory.

7.3.3 Viewing Only the Needed Files

File Manager can filter the objects in a directory before presenting them. By default, the File Manager filters out all of the hidden objects. Object filtering is based on an object's data type. For more information about data types see 8.3, "Data Types" on page 124. You can modify the File Manager's filter criteria by selecting **Modify Filter List** option from the View menu bar item. This will pop up the interface shown in Figure 39.

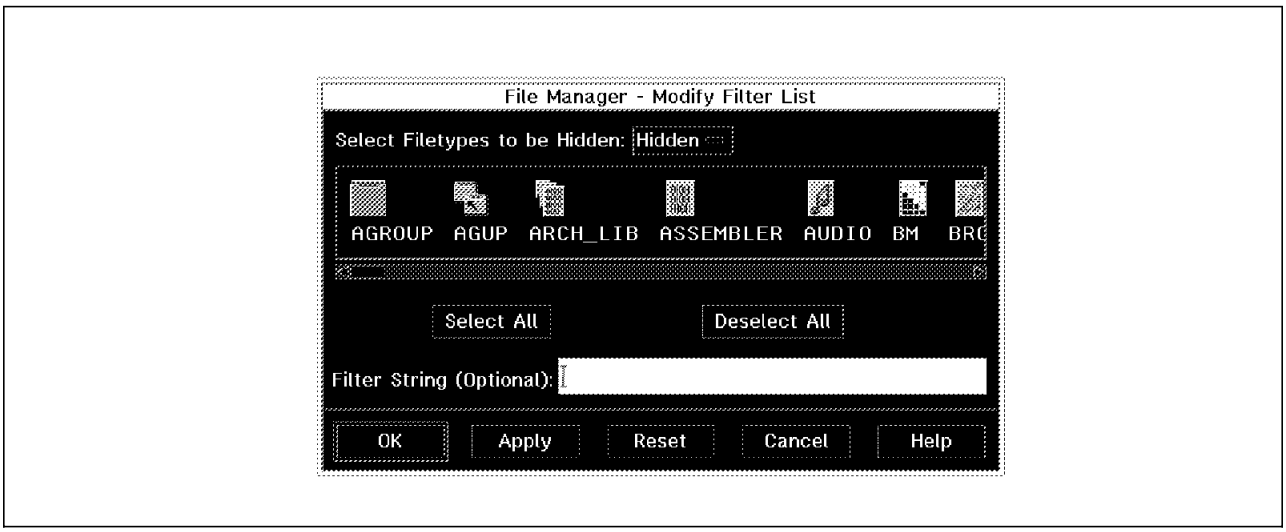


Figure 39. Modifying the Filer List

A list of data types which are defined on your system will be presented. This includes any of your user defined data types.

In order to filter objects, you have to first choose whether the data types you are going to select will be hidden or shown. Then select those data types that you want to hide or show. It is also possible to filter by a filename. To do this, enter the filename in the **Filter String** field. You may combine filtering by data types and by filter string. Figure 39 is an example where the filter criteria have been set to show only objects with the data type: DATA.

7.3.4 Moving/Copying/Linking Files

Moving, copying and linking files in the File Manager is done interactively by dragging and dropping the objects that represent the files to be manipulated.

To move, copy or link a file from one directory (source) to another directory (target):

1. Display both the source and target directories using the File Manager. If both the target and source directories are not immediate sub-directories of one another, you can either:

   Open the File Manager twice. This will give you two File Manager windows. Then change directory views in one of the File Manager windows to be a view of the source directory. And change directory

views in the other File Manager window to be a view of the target directory.

- Set the File Manager preferences to display the tree representation described in 113. Then open the tree structure to display both the target and source directories.

2. Drag and drop the object that represents the file to be manipulated:
   - To move the file, click and hold the left mouse button while the mouse pointer is over the icon of file to be moved. Then while holding the left mouse button down drag the icon from the source directory window into the target directory window. Release the mouse button.
   - To copy the file, press and hold down the keyboard Ctrl key and then follow the above step for moving the file. The addition of the Ctrl key into the drag and drop operation, tells the File Manager to copy the file instead of moving it.
   - To link the file, press and hold down both the keyboard Ctrl key and Shift key then follow the above step for moving the file. The addition of the Ctrl and Shift keys into the drag and drop operation, tells the File Manager to link the file instead of moving it.

During the drag and drop operation the mouse pointer changes to a representation displaying the type of operation being performed. Figure 40 shows the different pointer representations.

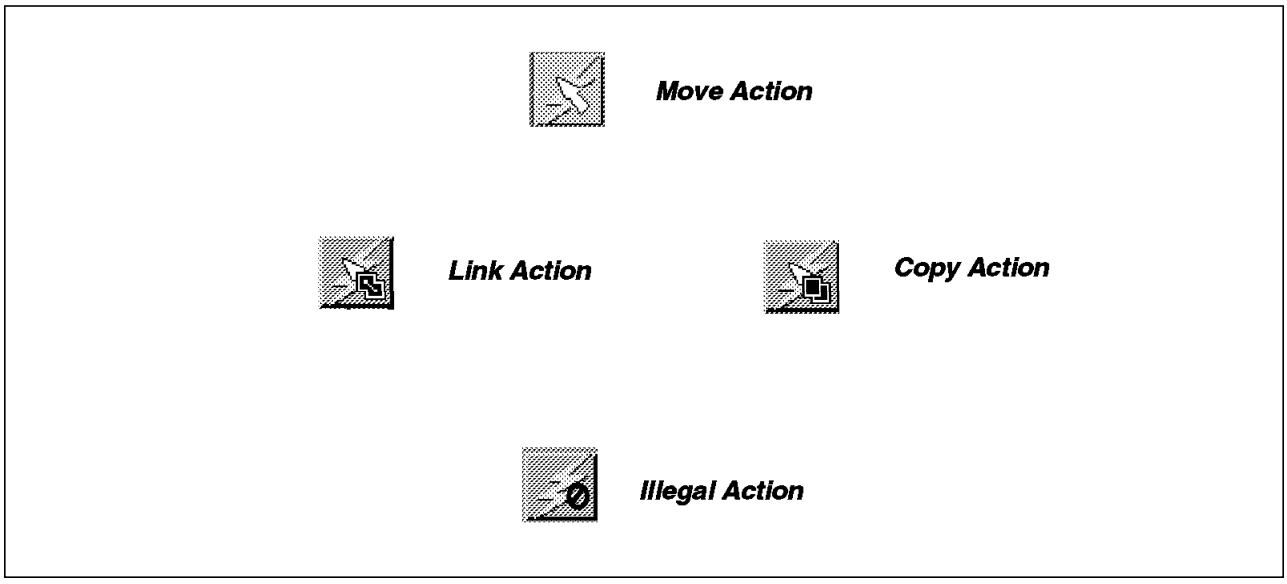


Figure 40. Different Cursor Representations

You may also note that areas of the screen where file dropping is allowed will change shape when dragging over the area. This feature is intended to help the user identify where a valid drop area is located. Where dropping it is not allowed, the pointer changes to the illegal action shape.

7.3.5 Renaming a File

To rename a file in the File Manager, click the left mouse button once while the mouse pointer is over the filename in the directory window. The filename will change into an editable box. Using this box, you can edit or enter a new name for the file. When finished, press the keyboard Enter key to save your new name, otherwise the change is lost.

7.3.6 Deleting a File

To delete a file with the File Manager, you can either:

- Drag the icon of the object that represents the file to be deleted and drop it on the trash can control in the front panel.
- Move the mouse pointer over the icon of the object that represents the file to be deleted and click the right mouse button. A pop up menu will appear. Select the **Delete to Trash** option from the menu.

With either method, the file is not actually deleted. Instead, it is moved into the directory $HOME/.dt/Trash. This is done so that the file can easily be recovered (picked out of the trash) if it was unintentionally deleted. It is best to occasionally go in and empty the trash by:

1. Opening the trash can by selecting the control in the front panel.
2. Selecting all (or just the desired) objects to be removed from the trash. The **Select All** option from the File menu bar option can be used to select all objects in the trash.
3. Select the **Remove** option from the File menu bar option. This action will delete the objects from the system.
4. Close the trash can.

7.3.7 Editing or Executing a File With the File Manager

You can edit a file, execute a program or invoke an action that is represented by an object in the File Manager by simply double clicking on the object. The operation that is preformed as a result of the double click depends on the objects data type and associated actions. For example, by default, an ASCII readable file is an object with the data type: DATA. The action defined for the data type: DATA is to invoke the Text Editor and load the file. Similarly, an executable is an object with the data type: EXECUTABLE. The action for data type: EXECUTABLE is to execute the file when selected. For more information about data typing see 8.3, “Data Types” on page 124.

7.3.8 Searching for a File by Filename or File Contents

The File Manager allows you to find a file in the filesystems by searching for it by name or its contents. To do this:

1. Select the **Find...** option from the File menu bar option. The interface shown in Figure 41 on page 118 will be displayed.
2. In the upper section of the interface you can enter your search criteria. You can either search for a specific name or the contents of a file. You can also combine those two search criteria by entering a filename and a file contents string. Using the wild card symbol (*) will allow you to specify only parts of the name or contents. For example,

   ```
   myfile*
   ```

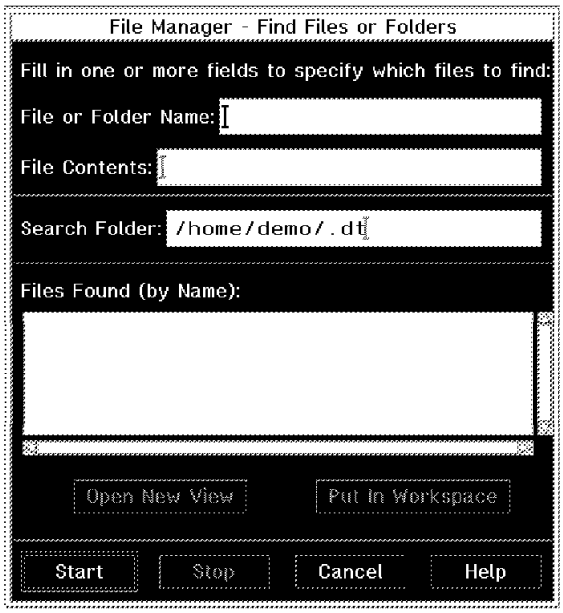


Figure 41. Find Files or Folders

Will find all the files that start with `myfile`.

3. In the next section of the interface, enter the search directory. If nothing is entered, the current directory of the file manager will be used.
4. Start the search by pressing the **Start** button. As files are found that match your specified criteria, they will be listed in the scrolled list: Files Found (by Name) You can stop the search at any time you want by pressing the **Stop** button.

7.3.9 Opening the File Manager on a Specific Directory

When you invoke the File Manager by clicking on the File Manager control in the front panel, the File Manager opens up and displays your home directory. There are several methods that can be used to tell the File Manager to display other directories:

- If you want to go to a directory which is a sub-directory of your current directory, you can double click on the directory object in the current directory that represents the desired directory you want to change to.
- If you want to go back in your directory path you can either:
  - Click on the desired directory in the iconic path.
  - Double click on the **..(go up)** object. This option is not available if you are using the tree representation.
- You can use the fast change path to enter the directory path you want to go to. Do this by:
  - Clicking once on the fast change path area in the File Manager. This area is the window that is displaying the full path name of your current directory. The path name will now become editable.

- Type the full path name of the directory you want to go and press Enter File Manager will change to the directory you entered.

- Under the **Folder** option of the menu bar you will find the following facilities to change to another directory:
  - **Fast Change To:** This gives you the same possibilities as the fast change path.
  - **Change To...:** This displays a list of all directories you have visited during this File Manager session. You can select one of the directories on the list or you can enter a new directory.
  - **Home:** This will change to your home directory.
  - **Up:** This will go up one directory level.

7.3.10 Moving File Manager Objects to the Workspace

You might want to execute an object without opening the File Manager or you may want to execute it on a workspace other than the one where your File Manager is currently running. This can be achieved by placing the object directly on the backdrop of a workspace. This can be done by either:

- Dragging the object from the File Manager and dropping it on the workspace.
- Moving the mouse pointer over the object and clicking the right mouse button. A pop up menu will appear from which you can select the **Put in Workspace** option.

A copy of the object will now exist on the workspace backdrop. The copy can be operated just like the original in the File Manager. Additionally, the copy will stay on the backdrop until you remove it. The object can be removed by moving the mouse pointer over the object on the backdrop and clicking the right mouse button. A pop up menu will appear, from which you can select the **Remove From Workspace** option.

7.3.11 Copying a File Manager Object to the Front Panel

You can copy a File Manager object to a subpanel of one of the front panel controls by dragging the object and dropping it on the **Install Icon** in the subpanel. The icon will be added at the bottom of the subpanel. For more information about the subpanel see 4.2.3.1, “Adding Controls to a Subpanel Using Interactive Customization” on page 52.

If you want to add the icon as a front panel control see 4.2.1.1, “Adding Controls Using Interactive Customization” on page 46.

Chapter 8. Launching Applications From the Desktop

An application is made up of executable programs/shell scripts and data files. The desktop offers several ways in which an application and its data files can be integrated into the desktop so that the user can launch the application. By default, the user can:

- Double click on the icon of the object which represents the application. All files including executable files and shell scripts are represented as objects under the desktop. The default action for an executable file is to execute it when selected. Therefore, if the executable for an application resides in the filesystem, it will be an object and thus have at least the default execute icon and the execute action associated with it. The icon for the executable can be double clicked from the File Manager to execute the application.
- Drag and Drop the application′s data files (represented by other objects) onto the icon of the executable. This will invoke the application and pass to it the name of the dropped data file(s).

These are the default launch possibilities. The desktop also provides facilities that allow an application developer, system administrator or the application end user to further integrate the application into the desktop. These facilities include:

- The ability to specify a specific icon to use when the object that represents the application is displayed.
- The ability to prompt the application user for information that will be passed as parameters to the application.
- The ability to associate an application executable with a specific type of data file. For example, all files with the .doc extension can be setup to bring up Framemaker and load the file when the object that represents the .doc file is selected.

A number of operating system commands, utility programs and basic applications are already integrated into the desktop using these facilities.

When a new application is installed in the system, two scenarios are possible:

- The application is desktop aware. This means it has been written to operate in the desktop environment and to cooperate with the desktop components. In this case, the application′s installation procedure should integrate it automatically into the desktop.
- The application is not desktop aware. In this case, you or your system administrator must perform a specific activity to integrate it into the desktop in order to go beyond the default launch capabilities mentioned above.

There are actually several levels to which an application can be integrated into the desktop. Level 0, as described in 9.1, “Integrating an Application into the Desktop” on page 144, contains the default launch capabilities mentioned above. Level 1 uses the other launch facilities mentioned above to integrate the application into the desktop in a more efficient and elegant manner. Level 1 integration and these facilities is the focus of this chapter.

8.1 Launch Integration Overview

Generally, in order to integrate an application into the desktop at a level 1 integration, the application developer, system administrator or the end user must supply:

1. Actions to provide the desktop with a general mechanism to launch the application. See 8.2, “Actions” and 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128.
2. Data Types for the application’s data files. This is not always strictly necessary but it’s strongly recommended, especially if your application works on data files with a specific format (graphic viewers, publishers and others). See 8.3, “Data Types” on page 124 and 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132.
3. An icon which is used for the iconic representation of the application. See 11.1, “Creating Your Own Icons” on page 161.

By providing these items and placing them in the appropriate directories, you will be able to launch your application from the File Manager (in other than the default manner) or add it to a subpanel or even the front panel. More activity is required, however, to launch your application from the Application Manager. See 9.1, “Integrating an Application into the Desktop” on page 144 for more information.

8.2 Actions

An action provides a general mechanism to invoke applications or procedures. In the most general case actions are represented as objects in the desktop so that the user can start the application just by double clicking on its iconic representation, in say, the File Manager. Actions can also be used in the desktop in many other ways, which do not always need an iconic representation. For instance actions can be called from other actions (map actions), the root menu, the command line, or even inside a user’s program.

There are several different types of actions:

1. **Command Actions** are the most common type, they are used to start a command, a program, a shell script or an application. They generally have an iconic representation, and the specified command will be invoked when the user double clicks on the action’s object.
2. **Map Actions** map the current action to another existing action. These are most commonly used in the data type’s action definition and they generally do not have an own iconic representation. See 8.3, “Data Types” on page 124 for more information about data types.
3. **ToolTalk Message Actions** are used to send ToolTalk messages by applications supporting this messaging system. These actions generally have no iconic representation.

Actions are defined using the following syntax:

```
ACTION ActionName
{
     TYPE    COMMAND | MAP | TT_MSG
     KEYWORD   value
     ...
}
```

Where:

ActionName
: A name for the action being defined. This name does not have to be unique. In fact there are circumstances when you want a non-unique name. See 8.8, “Creating Complex Actions or Data Types” on page 138 for such an example.

TYPE
: The type of the action being defined (COMMAND, MAP or TT_MSG).

KEYWORD/value
: Action parameters and values.

Many different keywords exist to specify the action’s behavior. The most commonly used keywords for the most common action type (command actions) are:

- EXEC_STRING - Specifies the command to execute
- ICON - Specifies which icon will be used by the object to represent the action

See *AIXWindows Desktop Advanced User’s and System Administrator’s Guide*, Chapter 9 for more information on the available keywords for action definitions.

Each action definition must be placed in a Dtdatabase definition file located in one of the directories listed in the DTDATABASESEARCHPATH variable value. At login the desktop will search in these directories for all the Dtdatabase definition files and will load all of action definitions from the files into the online Dtdatabase. See 8.4, “Action and Data Type Database” on page 126 for more information on the Dtdatabase its definition files and search paths.

As mentioned earlier, actions are generally represented as objects in the desktop. All actions that are objects meet the following requirements:

1. The action definition must exist in a Dtdatabase definition file. For example,

   ```
   ACTION ActionExample
   {
        TYPE    COMMAND
        ICON     Dtxmpicon
        EXEC_STRING myexample
   }
   ```

   The definition must include the ICON statement as shown in the example above. Notice that you should specify the icon file without mentioning the extension of the icon file name. The desktop will select the appropriate icon file (different size) to match the desktop component where the object will be displayed (File Manager, front panel, and son on). See 11.2, “Desktop Icon Name and Size Conventions” on page 163 for the information which icon file is used in which desktop component. The icon used to represent the action should be indicative of the behavior of the action itself. This will make it easy for the user to understand its purpose.

2. At least one icon file matching the name referenced on the ICON statement in the action′s definition must exist in one of the directories listed in the DTICONSEARCHPATH. For more information about icon file naming conventions and icons search path see Chapter 11, “Icons” on page 161
3. If the action is to be called from the File or the Application Manager, an action file must exist, be named with the same basename as the action definition and must be executable. This file acts as a flag in the filesystem that tells the desktop where to display the icon for the action. What this file contains is not important. It can even be empty, although it is recommended to contain at least a comment which describes this file to be an action file. The action file can be placed anywhere in the filesystem. When the File Manager is opened to the chosen directory, the action will be visible. If the action is defined so that it is part of the front panel the action file is not needed. This is only true if you are manually customizing the front panel though front panel definitions. Interactively adding the action to a subpanel and/or the front panel requires that you drag and drop the object from the File or Application Manager onto a subpanel. Since the object must first reside in either the File or the Application Manager the action file must exist, as stated above. For more information about objects see 7.1, “Object” on page 110.

For more information about the online action database see 8.4, “Action and Data Type Database” on page 126. See 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128 and 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132 for examples of action definitions. The *AIXWindows Desktop Advanced User′s and System Administrator′s Guide*, Chapter 7 has more information on search paths and Chapter 8 more information on Actions.

8.3 Data Types

Data typing allows you to:

- Represent similar files with the same iconic representation
- Define common actions to work on similar files

Data typing is used extensively throughout the desktop, and many of the commonly used data types are already defined in the system. With the File Manager it is easy to see that similar objects share the same icon. For example, all objects representing text files share the same icon. The same goes for all the C files and pixmap files. This is because all of these files files have been classified with data typing so that they are represented by similar objects in the desktop. Figure 42 on page 125 shows some of the iconic representations for some data typed objects.

Data typing not only specifies an icon to use for a similar group of files, it also specifies a set of actions that are available for the group. By default, all data types have at least one action defined. This action is named open and is invoked by double clicking on the object, in say, the File Manager. Since open is an action, each data type can define what open means for the files classified by the data type. For example, double clicking on a text file object in the File Manager will open the default editor on the selected file. Double clicking on a PostScript file, multimedia file or image file should call an appropriate application (PostScript viewer, media player, or image display tool) to handle the file. In

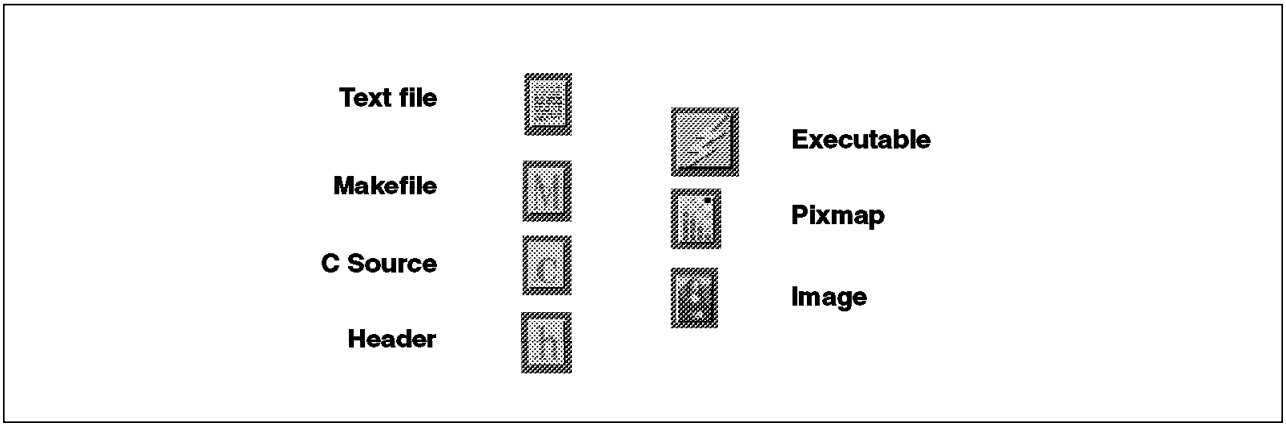


Figure 42. Data Type Icons

addition to the default open action, most data types also have a print action defined. This action is used to print the content of the file. Like the open action, the data type specifies the appropriate printer command to handle the data in the file. Other actions can be defined for a data type if needed.

Data typing is done through data type definitions. These definitions must be stored in a Dtdatabase definition file located in the search path defined by the DTDATABASESEARCHPATH variable. At login the desktop will follow this search path and load the definitions found in the various Dtdatabase definition files into the Dtdatabase. See 8.4, “Action and Data Type Database” on page 126 for more information on Dtdatabase files and search paths.

Data types are defined by first defining the classification. This is called DATA_ATTRIBUTES. Then each group of files which fall under the classification are defined. These are called DATA_CRITERIA. For example, you could define a datatype called SoundFiles. This would give a common icon and actions to all of the various types of sound files(.wav, .au, .snd, ...). This definition would be the DATA_ATTRIBUTES. Then you can create groups of DATA_CRITERIA statements which specify which sound file types are part of the SoundFile classification. One DATA_CRITERIA group is created for each different type of sound file. The following example illustrates this:

```
DATA_ATTRIBUTES SoundFile
{
     ACTIONS       Open, Edit
     ICON          Dtaudio
     DESCRIPTION   Sound files
}
DATA_CRITERIA SoundFile_AU
{
     DATA_ATTRIBUTES_NAME SoundFile
     MODE          !d&r&w
     NAME_PATTERN  *.au
}
DATA_CRITERIA SoundFile_WAV
{
     DATA_ATTRIBUTES_NAME SoundFile
     MODE          !d&r&w
     NAME_PATTERN  *.wav
}
```

In this example,

DATA_ATTRIBUTES defines the classification for the data type. There can only be one DATA_ATTRIBUTES statement for each classification and it must have a unique name (like SoundFile) and at least the ICON specification. Remember the open action is supplied by default. In this example, we are saying that a sound file can either be opened or edited by invoking the Open or Edit actions. These actions are defined elsewhere to play the sound file for Open and bring up a soundfile editor for Edit.

DATA_CRITERIA defines the criteria to be matched in order for a file to belong to the data type. Although there can be more than one DATA_CRITERIA statement for any given classification, each must have a unique name(SoundFile_AU and SoundFile_WAV).

- The DATA_ATTRIBUTES_NAME keyword identifies which classification the files specified by this classification belong to. In our example, all DATA_CRITERIA definitions specify files for the SoundFile classification.
- MODE, specifies the file type and permissions to match, values for this attribute are:
  - d to select only directories
  - !d to select only files
  - r,w,x to specify the file permissions
  - The & operator can be used to AND more of the conditions above
  - !d&r&w means that we want to select files with read and write permissions
- NAME_PATTERN specifies the file name pattern to match. In our example any files that have the .au or .wav file extension are part of the SoundFile classification.

With these definitions in place, whenever the File Manager opens a directory for you to view its contents, it will try and classify the files into objects in the directory according to the defined data types. In this example, all .au and .snd files will be displayed with the same speaker icon. Additionally, you can select one of the sound objects and have it played or edited.

See 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128 and 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132 for examples of actions and data type definitions.

For the actual syntax, available keywords and more information on DATA_ATTRIBUTES and DATA_CRITERIA refer to *AIXWindows Desktop Advanced User's and System Administrator's Guide* chapters 8,9,10 and appendix B.

8.4 Action and Data Type Database

The action and data type database is a memory table built out of all action definitions, data type definitions and front panel definitions known by the desktop. This database is called Dtdatabase.

The action definitions and data type definitions have to be stored in files with the extension .dt. For the purpose of this manual we call these files Dtdatabase definition files. A single Dtdatabase definition file can contain one or more

action definitions and/or one or more data type definitions. The front panel definitions have to be stored in files with the extension .fp.

The Dtdatabase definition files and the front panel definitions can be located in different directories. By default these directories are the following:

- $HOME/.dt/types
- /etc/dt/appconfig/types/%L
- /etc/dt/appconfig/types/C
- /usr/dt/appconfig/types/%L
- /usr/dt/appconfig/types/C

The desktop knows to search these directories, because the desktop is directed there by the DTDATABASESERARCHPATH environment variable.

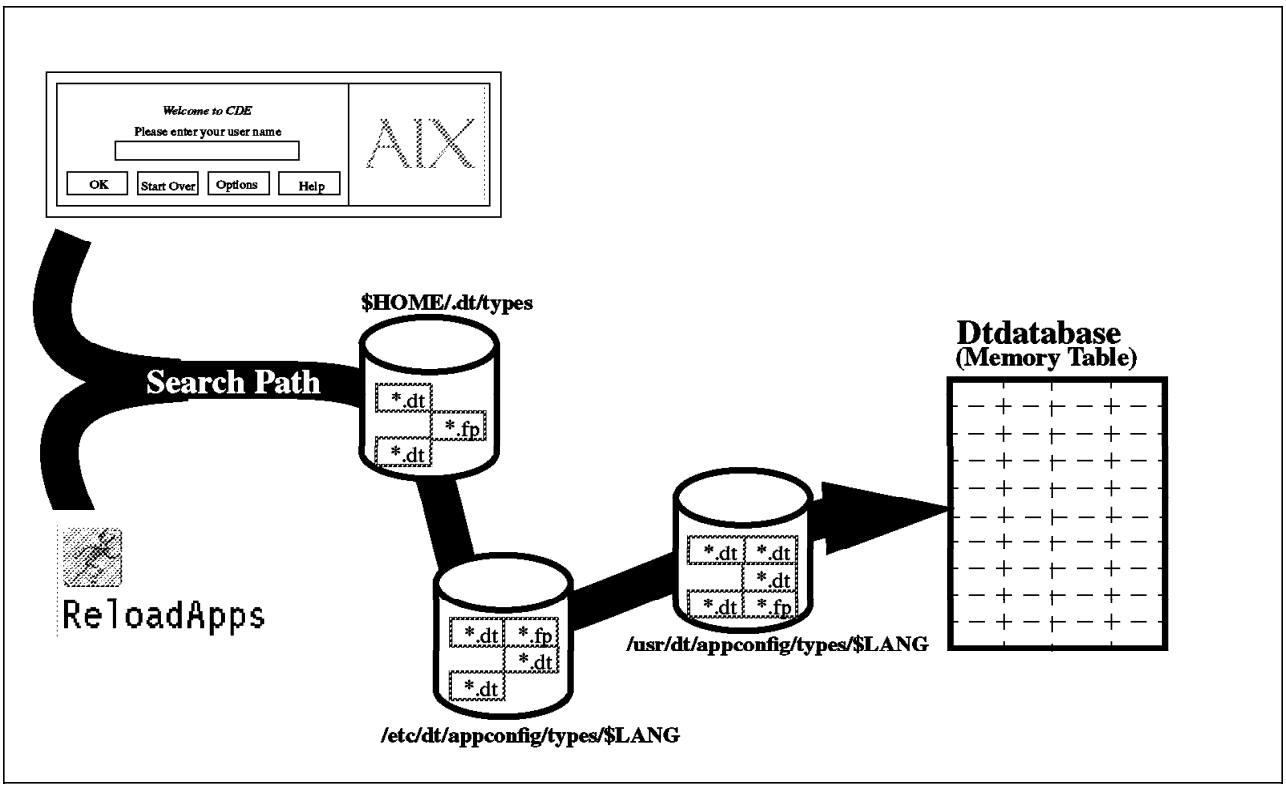


Figure 43. How the Dtdatabase is Built

At login or when the ReloadApps Action is executed the desktop will search in the directories listed in the DTDATABASESERARCHPATH variable for all the Dtdatabase definition files and the front panel definition Files. All action definitions, data type definitions and front panel definitions found will be loaded into the Dtdatabase in memory. The definitions in the $HOME/.dt/types directory have the highest priority. This means that if another definition with the same name is found in either the /etc/dt/appconfig/types/$LANG or the /usr/dt/appconfig/types/$LANG directories, it will not be read. Only the definition in $HOME/.dt/types will be loaded. The /etc/dt/appconfig/types/$LANG directory has a higher priority than the /usr/dt/appconfig/types/$LANG directory.
Figure 43 presents a graphical explanation of this process.

Only the action definitions, data type definitions and front panel definitions actually loaded in the Dtdatabase will be known and usable by the desktop. When a new action or data type is created, it must be placed in one of the directories defined in the DTDATABASESERARCHPATH variable, and the ReloadApps action must be run to force a reload of the Dtdatabase. ReloadApps can be found in the Application Manager's DesktopTools folder.

It is possible to add new locations to the search path. To do this on a single user basis, edit your $HOME/.dtprofile and add the following line to define the new search path location:

```
DTSPUSERDATABASEHOST=/newusrlocation
```

The DTSPUSERDATABASEHOST environment variable adds directories to the DTDATABASESERARCHPATH variable for a single user.

To add new locations to the search path on a system wide basis you must get root authority, create or edit the /etc/dt/config/Xsession.d/0010.dtpaths file, to add the following line:

```
DTSPSYSDATABASEHOST=/newsyslocation
```

The DTSPSYSDATABASEHOST environment variable adds directories to the DTDATABASESERARCHPATH variable on a system wide basis.

See 12.3, "Migrating Desktop Customizations to Another Desktop Host" on page 171 and *AIXWindows Desktop Advanced User's and System Administrator's Guide* chapter 7 for more information on search paths.

8.5 Creating a Simple Action Using the CreateAction Tool

CreateAction is an easy-to-use tool that allows users to create simple actions to start their applications. CreateAction can also be used to create simple data types, this capability is covered in 8.7, "Creating a Simple Data Type Using the CreateAction Tool" on page 132.

The following procedure explains how to create a simple action to start an application by double-clicking on its object in the File Manager. As an example we'll create an Action to start a host terminal emulator.

1. Open the Application Manager.
2. Open the **DesktopTools** folder.
3. Double click on the **CreateAction** object, to open the tool. Figure 44 on page 129 shows the CreateAction main interface.
4. Enter the name you want to give to your action in the **Action Name** field. In addition to the name of your action, this name will be used for both the action file filename, and the Dtdatabase definition file filename(appended with the extension .dt). In our example we used AUSVM.
5. Choose the icon to represent your action in the **Action Icons** area. You can click on **Find Set** to select one of the existing icons, or on **Edit Icon** to create a new one. You can also use the default icon (man running).
6. Enter the command that your action will execute when it is double clicked in the **Command** field. This is the most important part. Write the command as you would do on the command line. The full command path name is not

Figure 44. CreateAction Tool

required if the command is in your path. The command can also include arguments. For information on argument passing see 8.6, “Passing Arguments to Actions” on page 130. In our example we could enter: `xant ausvmr`.

7. Enter a brief description of the actions goal in the **Help Text** field. This text will be displayed by the Help Manager when a user ask for help on this action.

8. Indicate the type of application that the action invoke with **Window Terminal** options. There are four choices:

   - **Graphical (X-Windows):** This is the default, and must be used any time the application is X-Window based. Our example chooses this.
   - **Terminal (Auto Close):** This must be used when the application is not X-Window based, and therefore needs a terminal in order to run in an X-Window environment. Auto Close means that the terminal will be closed automatically when the application exits.
   - **Terminal (Manual Close):** The only difference with this and the previous option is that the terminal remains open after the application exits. This is useful when running batch commands. If you selected the auto close

terminal the window would disappear before you could read the command's output.

- **No Output:** This option is used for commands that do not produce output, such as background programs.

9. Choose the **Save** option from the File menu bar option. A dialog box will pop up to inform you that the action file has been created and placed in your home directory and that the Dtdatabase definition file has been created and placed in $HOME/.dt/types. In our example the tool created $HOME/AUSVM as our action file and $HOME/.dt/types/AUSVM.dt as the Dtdatabase definition file. The CreateAction tool will also reload the Dtdatabase automatically, so that you can see your new object in your Home folder using the File Manager.
10. Double click on the new object to check the result. Figure 45 on page 131 shows our new action in operation.

The action definition created by CreateAction for our example looks like this:

```
ACTION AUSVM
{
     TYPE          COMMAND
     EXEC_STRING   xant ausvmr
     ICON          Dtactn
     WINDOW_TYPE   NO_STDIO
     DESCRIPTION   Start VM emulator
}
```

There are a few limitations to using the CreateAction tool that you should be aware of:

- If you edit and change a Dtdatabase definition file created by CreateAction with any editor, CreateAction would not be able to reload it for further changes.
- CreateAction can only create command actions; Map or ToolTalk Message actions are not available.
- The CreateAction tool is designed for creating simple actions. Like the one created in the example. However the desktop is capable of handling far more complex actions. Unfortunately at this time it is only possible to create more complex actions by manually editing the Dtdatabase definitions files.

See 8.8, "Creating Complex Actions or Data Types" on page 138 for more information. See *AIXWindows Desktop Advanced User's and System Administrator's Guide* chapters 8 and 9 for more information on actions.

8.6 Passing Arguments to Actions

Actions can either accept one or more dropped objects as parameters or prompt the user for information.

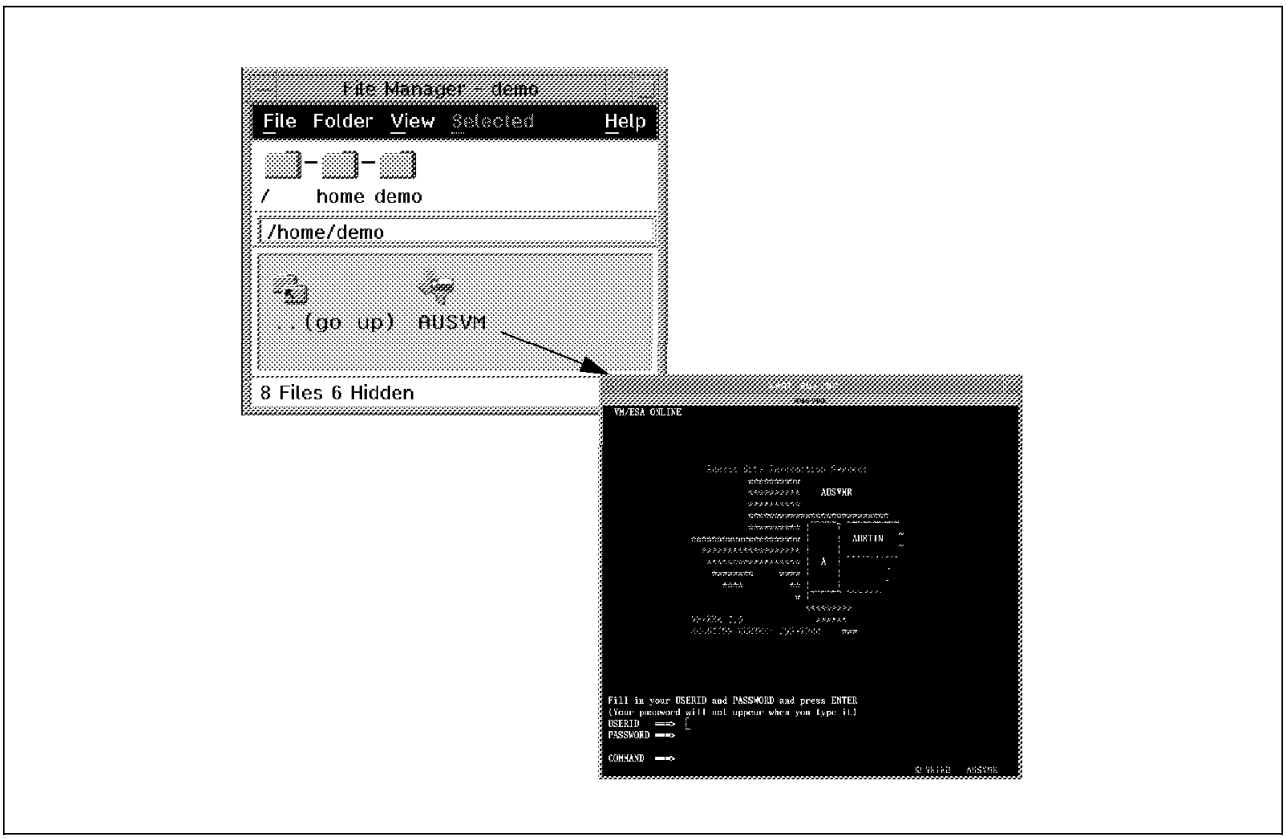


Figure 45. Using a New Action from the File Manager

8.6.1 Accepting Dropped Objects as Parameters

An action can accept parameters through a drag and drop operation by using one of the following syntaxes on the **Command** field in the CreateAction tool or on the EXEC_STRING keyword if manually editing the Dtdatabase definition files.

- `command %Arg_n%`

  The nth parameter is passed to the command. If you are using the CreateAction tool you can specify the arguments as $1 $2 ... CreateAction will substitute the string %Arg_n% for $n.

  Note: If you are qualifying the argument, as in the following options or adding a prompt, the $n syntax may not be used.

- `command %(File)Arg_n%`

  The nth parameter is passed to the command, and the parameter is interpreted as a file; this means that if it is specified as a remote file, the path is translated to the form hostname:/path.

- `command %(String)Arg_n%`

  The nth parameter is passed to the command, and the parameter is interpreted as a string; if a file name is passed, the hostname part is stripped off.

For example:

- `dtpad %(File)Arg_1%`

  The first parameter passed to the action is interpreted as a file name and passed to the dtpad editor.

- `lp -t%(String)Arg_1% %(File)Arg_1%`

  The parameter passed to the action is first interpreted as a string for the -t flag and then as the file to be printed.

- `diff %Arg_1% %Arg_2%`

  If no attribute is specified arguments are treated as files. The user must select the two files to compare and drop them together on the action.

8.6.2 Prompting the User For Information

An action can prompt the user for information by using one of the following syntaxes on the **Command** field in the CreateAction tool or on the EXEC_STRING keyword if manually editing the Dtdatabase definition files.

- `command %Arg_1"prompt"%`

  When the user double clicks on the action a dialog window prompts for an argument, that is then passed to the command as the first argument.

- `command %(File)Arg_1"prompt"%`

  When the user double clicks on the action a dialog window prompts for an argument, that is then interpreted as a file and passed to the command as the first argument.

- `command %(String)Arg_1"prompt"%`

  When the user double clicks on the action a dialog window prompts for an argument, that is then interpreted as a string and passed to the command as the first argument.

For example:

- `dtpad %(File)Arg_1"File to Edit:"%`

  If a object representing a file is dropped over the action, it works just as the previous example. If the user double clicks on the action a dialog window prompts for the file name to edit.

- `lp -t%(String)Arg_1% %(File)Arg_1"File to Print%`

  When the user double clicks on the action a dialog window prompts for the file name to print.

- `diff %Arg_1"File1"% %Arg_2"File2"%`

  In this case the dialog box has two text fields in which the two file names to be compared can be entered.

See *AIXWindows Desktop Advanced User's and System Administrator's Guide* chapter 9 for more information.

8.7 Creating a Simple Data Type Using the CreateAction Tool

CreateAction is an easy-to-use tool that allows even an inexperienced user to create simple actions to start their applications and simple data types to classify their data files.

The CreateAction tool creates simple data types which call actions like the one we created in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128. Using the CreateAction tool to create data types requires that you create at least one new action to be called by the data type. You can find a

more detailed description of how to create a new action using the CreateAction tool in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128.

The following procedure describes how to create a simple data type and an action to open the files classified by the data type. As an example we’ll create a data type for the FrameMaker Version 4 document files. To start we know that all valid FrameMaker Version 4 document files begin with the string <MakerFile 4.0K>. For the purposes of this example we are also assuming that all FrameMaker document files are always named with the extension .fmdoc. To create our Framemaker Version 4 data type:

1. Open the **DesktopTools** group in the Application Manager.
2. Double click on the **CreateAction** object to open the tool.
3. From the Options menu select **Show Advanced Functions**. The optional fields will appear in the CreateAction interface. Figure 46 on page 134 shows the CreateAction main window including the optional fields.
4. In the **Action Name** field enter the name you want to give to your action, for instance Maker.
5. In the **Action Icons** area choose the icon set to represent your action. You can either create a new icon or choose one of the existing icons.
6. In the **Command** field enter the command the action will execute if selected, for our example: maker4 -f $1. We want to open FrameMaker on a specific file, that’s why we use the -f flag and the $1 variable. The $1 symbol will be automatically translated into the correct action syntax by the tool. See 8.6, “Passing Arguments to Actions” on page 130 for more information on argument passing.
7. In the field **When Action Opens Ask User for** you can enter for example the text: Insert Frame Filename. This text will be used to prompt the user for a file name if none is provided. No quotes are needed around the text.
8. The Datatypes That Use This Action section lists the data types created for the current action so far. Of course, initially the list is empty. Press **Add** to create the first data type.
9. The Edit Datatype window is opened. From this window you’ll be able to define DATA_ATTRIBUTES and DATA_CRITERIA for your new Data Type.

   This window is shown in Figure 47 on page 135, you can operate on it as follows:

   a. By default the new data type will be named ActionName_FILE_1. In our case Maker_FILE_1. You can change this name if you want, for instance change it to MakerDataType.

   b. In the Identifying Characteristics section you must define the data type’s criteria. Press the **Edit** button to open the Identifying Characteristics window.

      This interface is shown in Figure 48 on page 136. You can operate on it as follows:

      1) Select the **Files/Folders** (directory) toggle, in our case we’ll select **Files**.

      2) Insert a name pattern in the **Name Pattern** field. In our example we entered *.fmdoc.

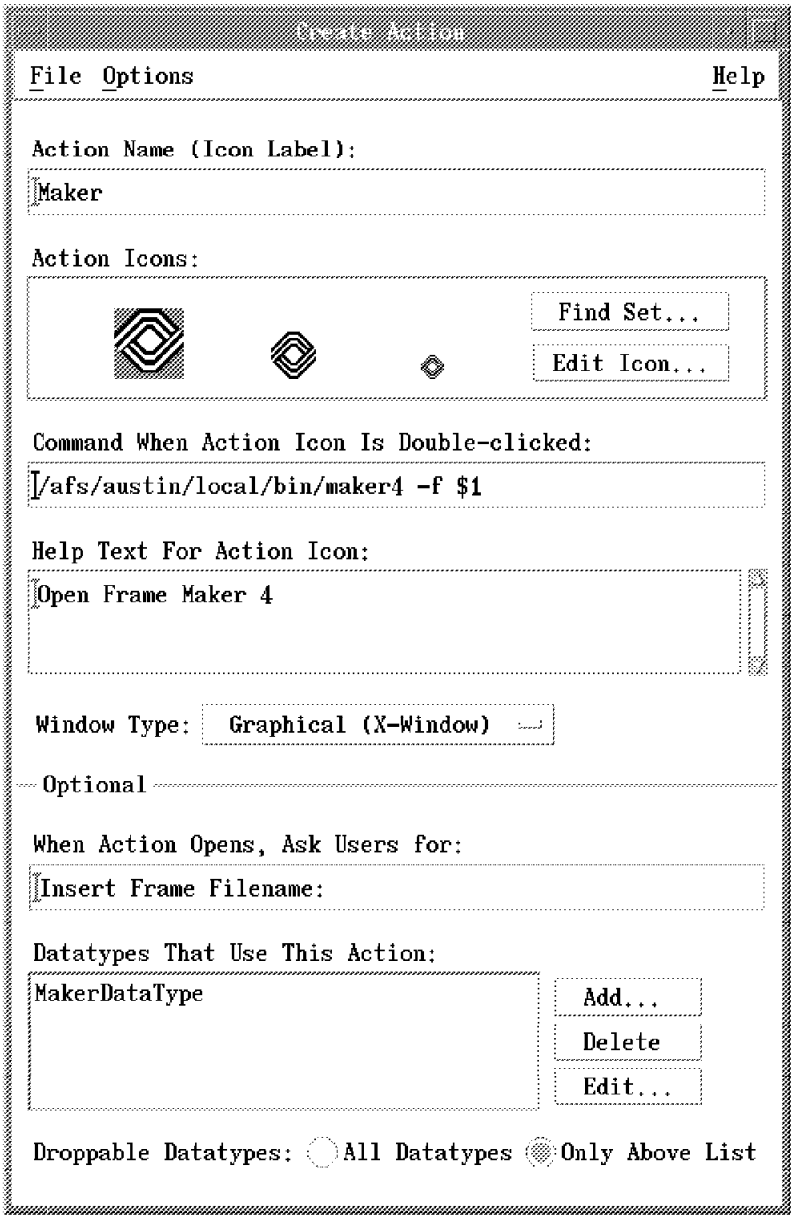


Figure 46. CreateAction Interface with Optional Fields

3) Use the **Permission Pattern** field to specify specific file permission pattern to match. Our example did not use this specification.

4) Use the **Content** area to specify the file contents to match. In our example, we specified <MakerFile 4.0K> in the **Pattern** field. **Type:** was String and 0 was specified in the **Start Byte** field.

5) Press **Ok** when you have entered all the characteristics that your files have to match.

Now you're back to the Edit Datatype interface.

c. Insert some Help Text for the Data Type. For instance Data Type for FrameMaker 4 files.

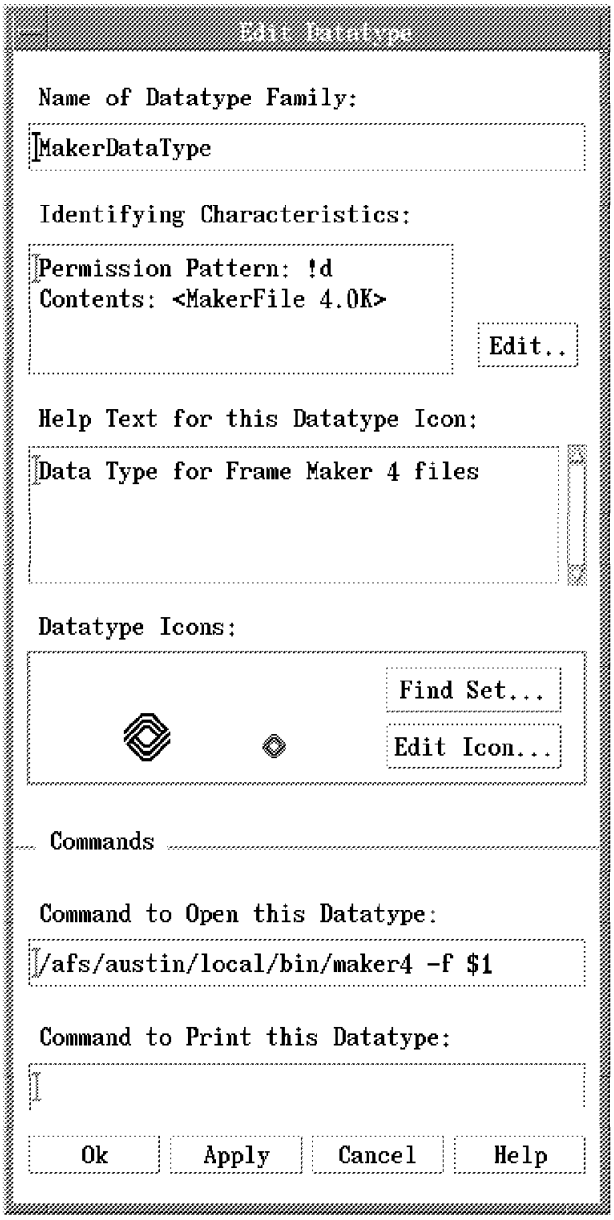


Figure 47. CreateAction - Edit Datatype Interface

d. Choose a Data Type icon. This can be the same as the one you chose for the Action or a different one.

e. In the **Command section** you have two text fields. The first contains the open action command string, this is the string that you previously entered in the action's command text field, and you cannot change it. The second text field is editable, and there you can enter a command string for the print action.

f. When you have entered all the information required, press **Ok**.

Now you're finally back to the main window.

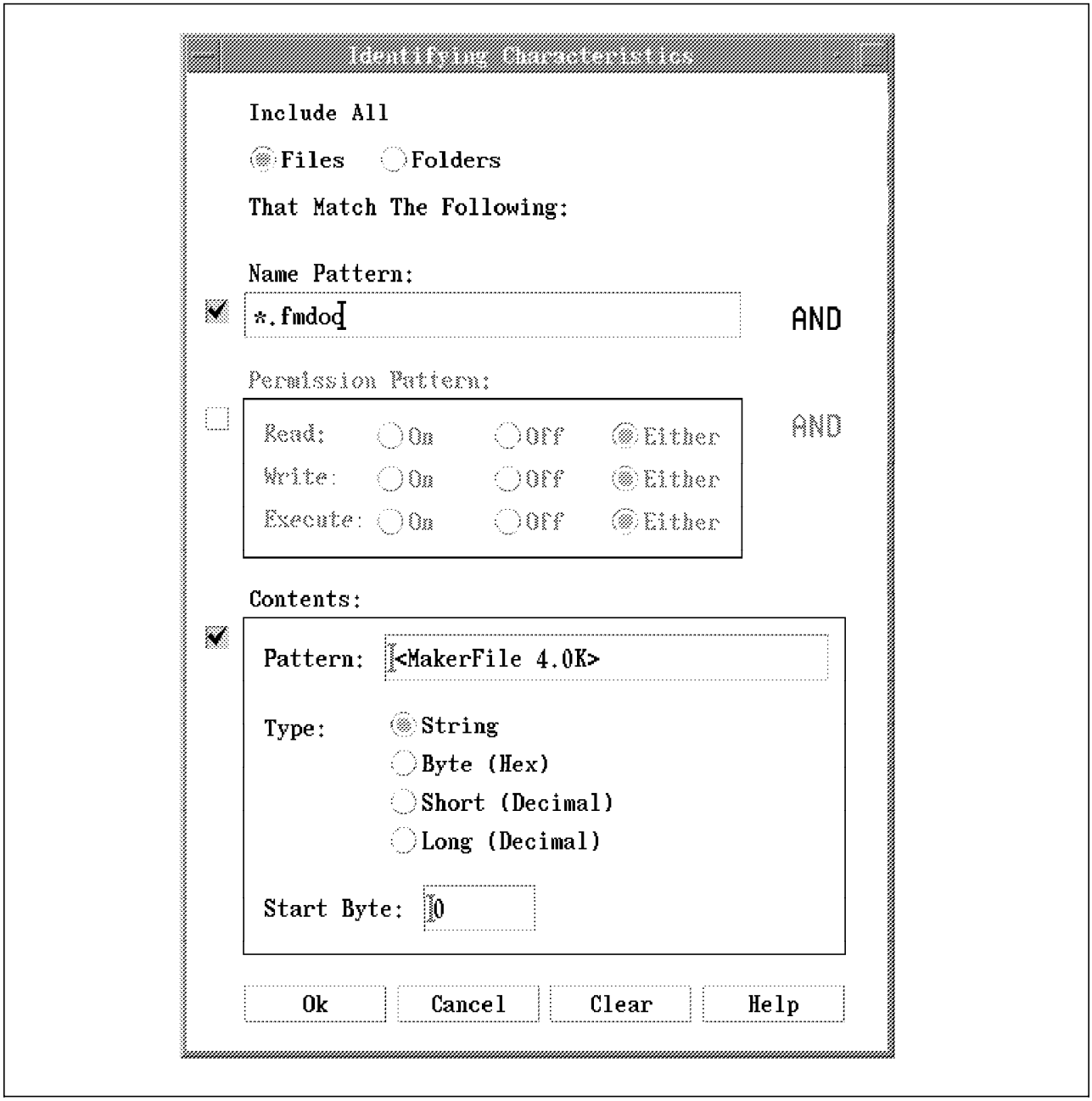


Figure 48. CreateAction - Identifying Characteristics Interface

10. The Datatypes That Use This Action list will now show the name of the data type that you just created. You can now add another data type, delete the one that you just created or edit it to make some change.

11. The Droppable Datatypes toggle button allows you to choose between **All Datatypes**, in which case all the files will be droppable on your action's object, or **Only Above List**, in which case only the files belonging to your data type will be droppable. The difference between the two cases is clear if you drop a non-FrameMaker file on the Maker object: in the first case (All Datatypes) the error will be intercepted by the FrameMaker application, in the second case (Only Above List) the error will be intercepted by the desktop which will realize that the file you have dropped is not of the right type, and the application won't even be called.

12. From the File menu choose the **Save** option. A dialog interface will inform you that two files have been created. In our example these will be: $HOME/Maker and $HOME/.dt/types/Maker.dt. CreateAction also reloads automatically the Dtdatabase, so that you can see your new object in your Home folder. Further more, if you navigate with the File Manager into the directory where you keep your FrameMaker files, you will see that they all have different iconic representations than before. Now they all use the icon that you set for the data type that you have just created.
13. You can now check the new data type in different ways:
    - Double click on one of your FrameMaker files: FrameMaker should start and open this file.
    - Drag and drop one of your FrameMaker files over the Maker action in your $HOME folder: again FrameMaker should start and open this file.
    - Double click on the Maker object in your $HOME folder:
      - If you selected **All Datatypes** then a dialog window will prompt you for a Frame Filename. Enter a valid file name and press **Ok**, FrameMaker should start and open this file.
      - If you selected **Only Above List**, you will get the error message `Action Maker was not found.` Actually the Maker action does exist but can be accessed only by a MakerDataType file. In fact you can still drag and drop a FrameMaker data file on it and it will work.

The action and data types definitions created by CreateAction look like this:

```
ACTION Maker
{
     TYPE          COMMAND
     EXEC_STRING   maker4 -f %Arg_1"Insert Frame Filename:"%
     ICON          frame
     WINDOW_TYPE   NO_STDIO
     ARG_TYPE      MakerDataType # if you selected "Only Above List"
     DESCRIPTION   Open Frame Maker 4
}

DATA_ATTRIBUTES MakerDataType
{
     ACTIONS       Open
     ICON          frame
     DESCRIPTION   Datatype for Frame Maker 4 files
}

DATA_CRITERIA MakerDataTypeA
{
     DATA_ATTRIBUTES_NAME MakerDataType
     MODE          !d
     PATH_PATTERN  *.fmdoc
     CONTENT       0 string <MakerFile 4.0K>
}

ACTION Open
{
     ARG_TYPE      MakerDataType
     TYPE          MAP
     MAP_ACTION    Maker
```

```
    LABEL           Open
}
```

It is interesting to notice that two Actions are actually created. One is called Maker, this is the command Action that starts the application, and it is called directly when you double click on the Maker action. The other one is called Open, and it is a map action. When you double click on a MakerDataType file the Open action is invoked, that turns the call to the Maker action. In this way the standard Open action, that every data type has, is remapped to a specific action valid only for the MakerDataType files.

CreateAction limitations:

- If you edit and change a file created by CreateAction with any editor, CreateAction would not be able to reload it for further changes.
- CreateAction can only create command actions. Actually we have just seen that the Open Action is a map action, which is created automatically by CreateAction and the user has no way to instruct the tool to create another or a different map action.
- CreateAction cannot create actions or data types more complex than the one shown above (except for the execution string complexity). For example only one DATA_CRITERIA can be created for each DATA_ATTRIBUTE. On the other hand, actions and data types can be much more complex than that. If you need to create more complex actions or data types you must create them manually.

See 8.8, “Creating Complex Actions or Data Types” for more information. See *AIXWindows Desktop Advanced User's and System Administrator's Guide* chapters 8,9 and 10 for more information on actions and data types.

8.8 Creating Complex Actions or Data Types

As discussed in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128 and 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132 the CreateAction tool has several limitations. In many circumstances you may create the basic actions and data types for your application with CreateAction and then edit the Dtdatabase definition file and manually add the extra functionality that you need.Remember, once you edit this file manually you won′t be able to load it in CreateAction anymore.

The example in 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132 gives us a good opportunity to explain why and how to add new functionality to the action and data types definition created by CreateAction.

In that example we created an action called Maker and a data type called MakerDataType. In that procedure (step 11) we explained how you can choose either the value All Datatypes, or Only Above List for the attribute Droppable Datatypes. If you chose Only Above List you may have noticed something strange:

- You can double click on a data file to open the application on that file.
- You can drag and drop a data file onto the Maker object to open the application on that file.

- But if you double click on the Maker object itself you get the error message: `Action Maker was not found`. The reason you get this error is that you can access the Maker Action only through a MakerDataType file.

Since you generally still want to access the application via its object, you can either switch back to use All Datatypes, in which case you have no filtering on the droppable data types, or you can try something more complex, which involves the manual editing of the Dtdatabase definition files created by CreateAction.

The desktop allows you to have more than one action with the same name. So the idea here is that you could write another action called Maker, similar to the one you already have, but which is not bound to any particular data type, and just let the desktop decide when to use one or the other. To do this you can follow this procedure:

- Change directory to $HOME/.dt/types and change the name of Maker.dt file using:

  ```
  mv Maker.dt MakerData.dt
  ```

- Open CreateAction to create an action called Maker. For simplicity we'll refer to this one as the new Maker action and to the one we created in the previous example as the old Maker action. It must be clear though that we want to use both these two actions.
- Use the same command string you used for the old Maker action:

  ```
  maker4 -f %Arg_1"Insert Frame Filename:"%
  ```

- Choose the same icon that you chose for the old Maker action.
- Select **All Datatypes**
- Save. The tool generates a new Dtdatabase definition file called Maker.dt. At this point it should be clear why you had to change the Dtdatabase definition file name to MakerData.dt before creating the new Maker Action. Remember that CreateAction always creates a Dtdatabase definition file with the same basename as the action you create. It doesn't let you choose a different Dtdatabase definition file filename, and it doesn't warn you that another Dtdatabase definition file with the same filename already exists.

Now you can double click on the Maker object to start FrameMaker, but you have still no filtering on the droppable data types. To get this function you can now edit the new Maker.dt Dtdatabase definition file and force the filtering of unwanted data types by adding a new attribute to the action definition:

```
ARG_COUNT   0
```

This attribute specifies that the new Maker Action will not accept any direct argument passing, but you can still prompt the user for an argument. Save the change in Maker.dt and run the **ReloadApps** action from the Application Manager.

The new Maker Action definition looks like this:

```
ACTION Maker
{
 TYPE          COMMAND
 EXEC_STRING   /frame4/bin/startmaker4 %Arg_1"Insert FrameMaker File"%
 EXEC_HOST      cde
 ICON          frame
```

```
 WINDOW_TYPE   NO_STDIO
 ARG_COUNT     0
 DESCRIPTION   Open FrameMaker4 double clicking on Maker icon
}
```

What really happens is:

- When you double click on the Maker object the new Maker Action gets called. It will prompt you for a file name and when you press **OK** the application will be invoked.
- When you drag and drop a file on the Maker object, the desktop will realize that the new Maker Action cannot be called because it has ARG_COUNT set to zero. The desktop will then look in its Dtdatabase to see if there is any other Maker action applicable and will still find the old Maker Action. This action defines the MakerDataType as the only droppable data type, so the action will be called only if the object being dropped is a FrameMaker file.
- When you double click on the object representing a FrameMaker document, the desktop will use the file name as the argument for the Maker action. So just as in the previous case the old Maker Action will get called.

Notice that our action and data type definition for the FrameMaker application is now spread across two Dtdatabase definition files: Maker.dt and MakerData.dt. This is perfectly fine. On the other hand we could have also edited the old Maker.dt and manually added the action definition shown above. The only advantage in having two files in this case is that at least one of them (MakerData.dt) can still be loaded into CreateAction for further changes.

For more information on how to write complex actions or data types refer to *AIXWindows Desktop Advanced User's and System Administrator's Guide*.

8.9 Invoking an Action from the Command Line

The `dtaction` command allows you to invoke an action from the command line.

The dtaction syntax is:

```
dtaction [-user username] [-execHost hostname] action [argumentlist]
```

The -user option allows a user name to be specified; if dtaction is not currently being run by that user, then a prompt dialog will be used to collect either the password for the indicated user, or the root user password. Once a valid password has been entered, the dtaction client will change so that it will be run by the requested user and then initiate the requested action. Anytime the dtaction client changes to run as a different user, it will be logged in the file: /usr/adm/sulog.

The -execHost option can be used to specify an alternate execution hostname, for a command action. If the action is not a command action, then this option is ignored. The action will be attempted on host_name before any of the hosts specified in the action's EXEC_HOST specification. An error dialog will be posted if it is not possible to invoke the specified Action on any eligible host.

For example:

If you want to call the default terminal program from the command line, enter the following command:

```
dtaction Terminal
```

If you want to get the terminal with root authority, enter:

```
dtaction -user root Terminal
```

If you want to open your HOME folder in the File Manager with root authority, enter:

```
dtaction -user root OpenHomeDir
```

See *AIXWindows Desktop Advanced User's and System Administrator's Guide* and the man pages of the command dtaction for more information.

Chapter 9. Application Manager

The Application Manager provides easy access to desktop tools and user applications. The user does not need to be aware of the actual location of the tools and applications in the file system or even on the network. The user simply selects the icon for their desired tool or application and the desktop does the rest.

The Application Manager is represented by a control in the front panel which is shown in Figure 49. The interface of the Application Manager itself is very similar to that of the File Manager. In fact the Application Manager is a special instance of the File Manager with the restriction that you cannot move outside its directory. For example, changing the visual representation of objects or executing an object is the same in the Application Manager as in the File Manager.

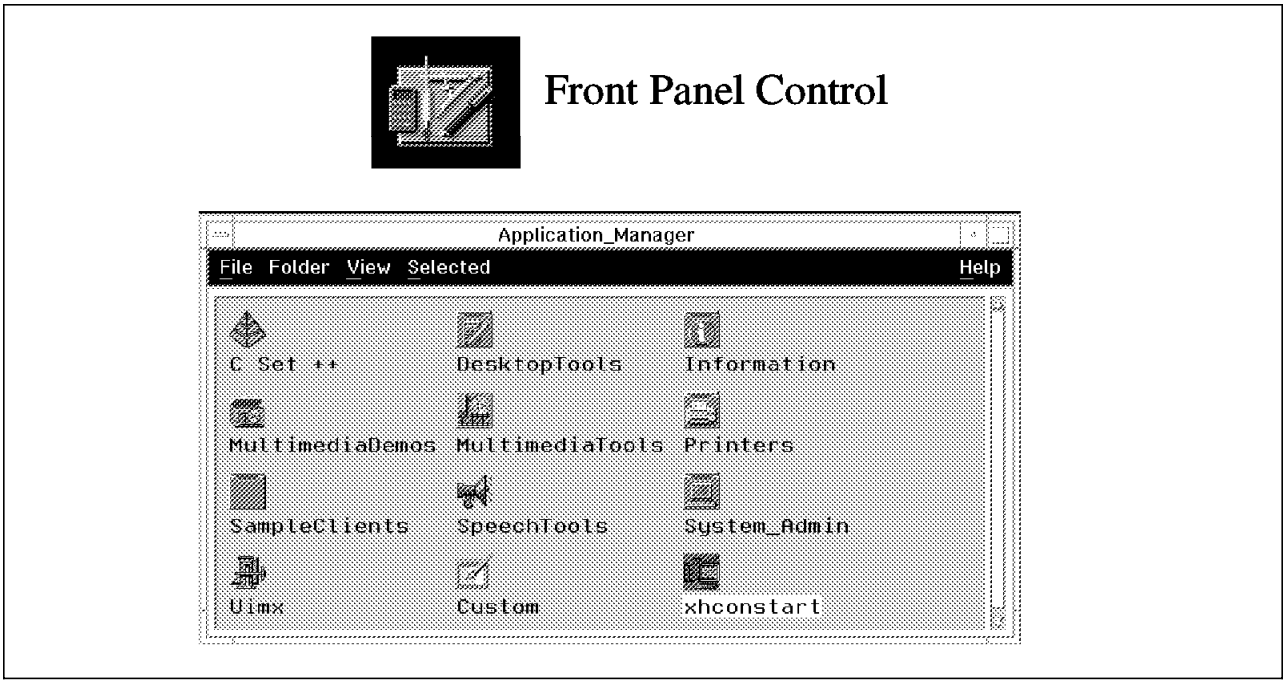


Figure 49. The Application Manager Front Panel Control and the Main Interface

As mentioned earlier, the Application Manager provides a repository of various applications and tools. These applications and tools are arranged in a fashion similar to the File Manager. Applications and tools are be stored in groups similar to directories. Which in turn can hold other groups or the icons for the applications themselves. The main difference between the File Manager and the Application Manager is that applications and tools are organized by functionality and not filesystem location. For example, the default supplied configuration of the Application Manager contains a group called DesktopTools. This group is a set of miscellaneous desktop tools. This group also includes another group which is a set of UNIX utilities which provide graphical interfaces to many of the more commonly used UNIX commands. The actual tools called by the objects in these groups reside in many places through out the filesystem. However, they can all be accessed from this single location.

Applications and tools can easily be added to the Application Manager. Since adding an application into the Application Manager is in essence a form of integrating it with the desktop, the following section will explain what it really means to integrate an application into the desktop. The subsequent sections will then explain how to add applications into the Application Manager for those applications and tools that don't automatically add themselves as part of their installation process.

9.1 Integrating an Application into the Desktop

Whether you are a desktop user, a system administrator or a developer, you may want your application to be integrated into the desktop. Applications that are integrated into the desktop are represented by objects in either the File or Application Manager. These objects have the ability to click to start and they can be dragged and dropped onto other desktop objects such as the printer or the text editor.

The desktop has four different levels of integrating an application into it, these are seen in Figure 50.

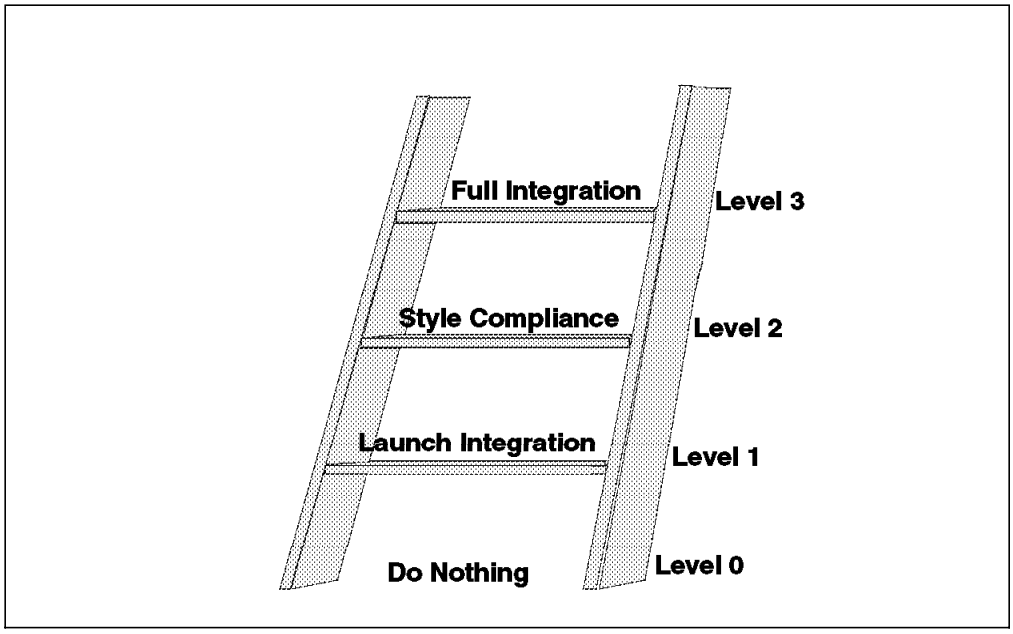


Figure 50. Four Levels of Integration in the Desktop

Level 0 is the lowest level. At level 0 an application is not integrated into the desktop at all. This means, the only way to start it is by issuing a command on the command line of a terminal window or selecting the default object supplied for the applications executable in the File Manager. Although this level of integration is not easy to use or elegant, the application can still be run with no additional effort.

Level 1 is probably the most common level to integrate an application. With this level of integration you will be able to start the application by double clicking on the representative icon for the application in the File Manager, the Application Manager, a subpanel or the front panel. You can additionally, start it through the drag and drop facilities of the desktop. For the most part, this level of integration is done through the facilities of actions and data typing. For more information

about actions and data types see Chapter 8, “Launching Applications From the Desktop” on page 121. Once an application is made into an object though actions and data typing, it can easily be added to a subpanel, the front panel or the Application Manager. For more information about subpanels and the front panel see Chapter 4, “Front Panel” on page 31. Adding an object into the Application Manager will be covered in the following sections of this chapter.

Level 2 is built on top of level 1, and covers all applications which are desktop style compliant. A desktop style compliant application must comply with the *OSF/Motif 1.2 Style Guide and the Common Desktop Environment Certification Checklist*. Applications should also follow the guidelines in the *Common Open Software Environment Internationalization Programming Guide* in order to be ready for non-US English environments. To integrate existing applications on this level it may be necessary to change the application’s source code.

Level 3 is built on top of level 2. It uses the facilities of the desktop through the desktop APIs. You will need to make source code changes to make an existing application integrated at level 3. The APIs for this environment are not available with this release of AIXwindows desktop. Therefore, it is not possible to do a level 3 integration at this time.

9.2 Adding System Wide Applications to the Application Manager

Integrating an application, system wide, into the Application Manager, is a two step process. First, a specific file structure with some icons, actions and data type definitions needs to be created. Then secondly, the command `dtappintegrate` has to be run in order to integrate the application into the Application Manager.

Generally applications that are desktop aware, will integrate themselves into the desktop and the Application Manager.

For applications that are not desktop aware, the system administrator is responsible for integrating them into the Application Manager. To do this, the system administrator has to create a special file structure and specific files under the application’s root directory. Then `dtappintegrate` needs to be run to complete the integration.

The specific files that are needed to integrate an application into the Application Manager are:

1. **Action file:** This file is the flag that tells the desktop where to display the icon which represents the action that launches the application. In other words, it allows the actions to be visible within the Application Manager or the File Manager. It is needed for each action you want to be represented in the Application Manager and has to have the following properties:
   - The name should be meaningful.
   - It must have execution permission.
   - The content is irrelevant. The file can be empty, but it is recommended that the file contain ASCII text stating it is an action file.

   For more information about the action file see 8.2, “Actions” on page 122.
2. **Dtdatabase definition file:** The provided Dtdatabase definition file must contain either action definitions, data type definitions or both which define

how the application is to be launched from the desktop. For more information on action definitions see 8.2, “Actions” on page 122 and 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128. For more information about data typing see 8.3, “Data Types” on page 124 and 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132. For more information on the Dtdatabase definition file refer to 8.4, “Action and Data Type Database” on page 126.

3. **Icon files:** They are used for the iconic representation of the application′s objects in the desktop. There should be icons for the executables as well as the various types of data files. For more information about icons, how to use them and how to create your own icons, please refer to chapter Chapter 11, “Icons” on page 161.

All these files have to exist in different sub-directories under the application′s root directory. They all have to contain language directories. At least, the directory for the locale C is recommended. If you want to have different data used by the different language environments, which might be used in your system, you must put this data in the associated directories. For example, your application may support different Help Volumes for several languages. Those directories are shown in Figure 51 and are described as follows:

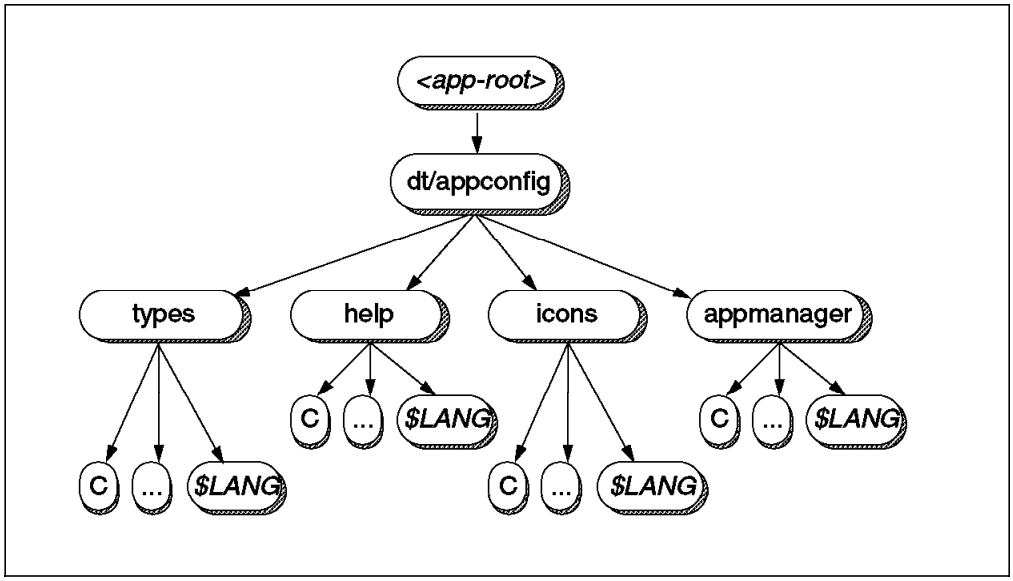


Figure 51. Desktop Specific Subdirectories of the Application′s Root Directory

1. **Types directory:** This directory contains, in the different language directories, the Dtdatabase definition files for your application. It may also contain front panel definition files, if there are any.
2. **Help directory:** This directory contains the help files for the application.
3. **Icons directory:** This directory contains, in the different language directories, all icons which are associated with this application.
4. **Appmanager Directory:** This directory contains, in the different language directories, all the associated action Files. They should be stored in a sub-directory below the language directory which will become the name of the Application Manager group. For example, /Frame4/dt/appconfig/appmanager/$LANG/FrameMaker will create a folder in the Application Manager called FrameMaker that will hold all of the objects that launch the FrameMaker application.

After creating the required files in this file structure under the application's root directory you need to run `dtappintegrate`. This command will register the application's actions, data types, help volumes, and icon files with the desktop and create the application's folder in the Application Manager.

The command syntax is as follows:

```
dtappintegrate -s <app_root> [ -t <target_path> ]
                [ -l <lang> ] [ -u ]
```

`-s <app_root>`
: This parameter specifies the path of the application's root directory. This parameter is required.

`-t <target_path>`
: This parameter specifies the path where the application's desktop configuration files will be linked rather than the default system location. This parameter is optional.

`-l <lang>`
: This parameter specifies the directory under which the location-specific files to integrate are located. This parameter is optional. If this parameter is not specified, then all languages which exist in this directory will be integrated.

`-u`
: This parameter un-integrates the application. This parameter is optional.

For example:

Suppose you want to integrate FrameMaker into the Application Manager and you are going to use the action which was created in 8.7, "Creating a Simple Data Type Using the CreateAction Tool" on page 132.

1. Create the dt/appconfig directory in the application's root directory. Let us say the applications root Directory is /Frame4. So the directory will be /Frame4/dt/appconfig.
2. In this directory create the directories: types/$LANG, help/$LANG, icons/$LANG, appmanager/$LANG.
3. Create a directory called FrameMaker in the appmanager/$LANG directory. This will give you a folder in the Application Manager interface named FrameMaker.

4. You should now have the following directories:

   /Frame4/dt/appconfig/types/$LANG
   /Frame4/dt/appconfig/help/$LANG
   /Frame4/dt/appconfig/icons/$LANG
   /Frame4/dt/appconfig/appmanager/$LANG/FrameMaker

5. Create the:

   - Icon files for all needed sizes and give them a base name for example, MakerIcon.
   - Dtdatabase definition file which describes how to launch the application from the desktop. In this example, the $HOME/.dt/types/Maker.dt file was previously created in 8.3, “Data Types” on page 124.
   - Action file, again, our $HOME/Maker file was previously created.

   For more information on how to create these files refer to 11.1, “Creating Your Own Icons” on page 161, 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132 and 10.2, “Adding Your Own Help Volume or Changing an Existing Volume” on page 159.

6. Copy the files into the directories:

   - MakerIcon in icons/$LANG
   - $HOME/.dt/types/Maker.dt in types/$LANG
   - $HOME/Maker in appmanager/$LANG/FrameMaker

   The preparation for the integration is complete.

7. Execute the command:

   ```
   dtappintegrate -s /Frame4
   ```

8. Run the **ReloadApps** object in the DesktopTools group of the Application Manager to activate the integration. You should now be able to open the Application Manager and see the new FrameMaker folder.

9.3 Adding User Specific Applications to the Application Manager

Users can create their own actions to launch applications and tools by using the tool called CreateAction. (See 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128 for more information.) You may want to group the objects for these actions together into a new folder in the Application Manager group. This can be done by adding your own group to the Application Manager and subsequently adding the desired objects into this new group. To do this:

Create a new directory:

1. Create a new directory as a sub-directory of $HOME/.dt/appmanager, which will become the new folder in the Application Manager. For example, $HOME/.dt/appmanager/MyTools.
2. Select **ReloadApps** in the DesktopTools group of the Application Manager to activate the new folder.
3. The new application folder will now be displayed in the Application Manager. This icon only represents another view of the /$HOME/.dt/appmanager/MyTools directory. The view under the File Manager still exists.

Adding actions:

1. If needed, create the desired actions for the applications and tools to be included in the new folder. You can do this with the CreateAction tool. For more information see 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128.
2. Open the File Manager and move, via drag and drop, the objects of the desired tools into the new folder in the Application Manager. For more information on copying, moving and linking files with the File Manager see 7.3.4, “Moving/Copying/Linking Files” on page 115.

Your tools will now be displayed in the MyTools group of the Application Manager. Figure 52 shows the Application Manager after adding the new folder.

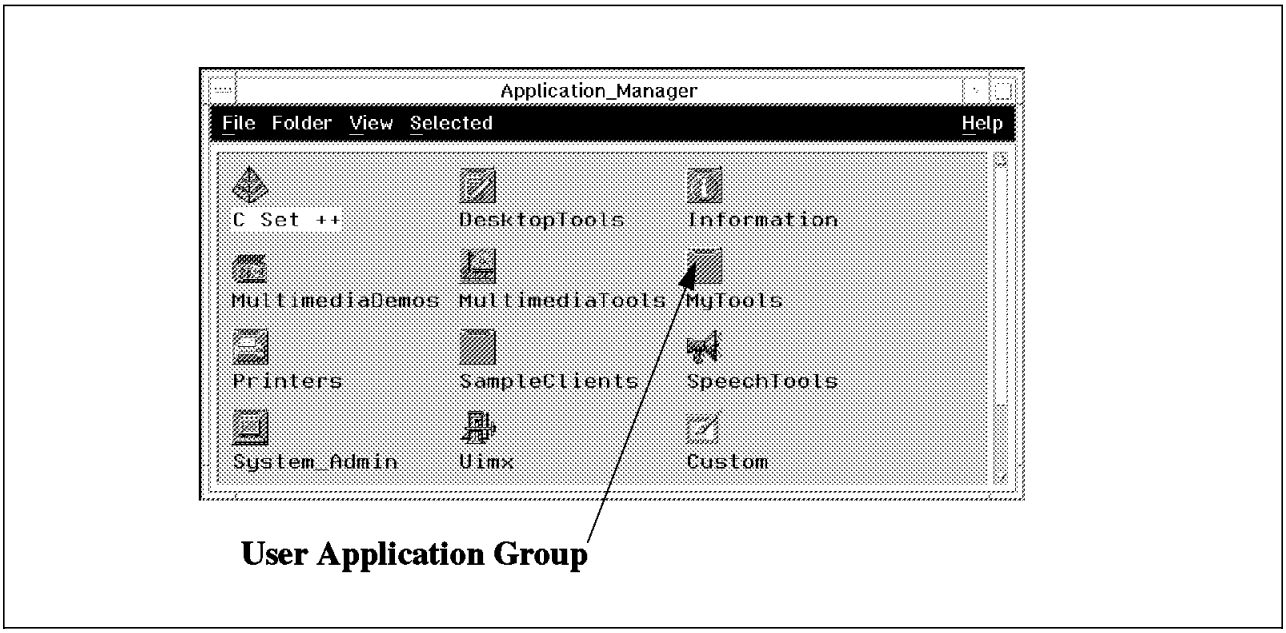


Figure 52. Application Manager with the Folder MyTools

For example:

Let’s say you want to integrate the host terminal emulator action (AUSVM), which was created in the example in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128, into your tools group in the Application Manager:

1. In the $HOME/.dt/appmanager directory create a new subdirectory. For example: MyTools.
2. Open the **DesktopTools** Group in the Application Manager and reload the applications by clicking on **ReloadApps**.
3. The Application Manager will now have your new folder called **MyTools**.
4. Double click on this folder to open it.
5. The action for the terminal emulator AUSVM was created with the CreateAction Tool in 8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128. To place this tool in your new folder, Drag the icon for this tool from the File Manager into the MyTool Group in the Application Manager.

This tool is now integrated in your personal Application Manager folder.

Chapter 10. Help Manager

The desktop Help Manager provides desktop and application help information in a hypertext format. The information is organized hierarchically by:

1. **Volume** - A help volume consists of groups of related help topics. Usually covering a particular product, tool or task.
2. **Topic** - A help topic can be either a subgrouping of information pertinent to a particular topic in the help volume or the entry containing information on the topic itself. Help topics are arranged hierarchy. The first topic in a help volume is called the home topic.

For example, a help volume for a specific application could be organized using the following main topics:

- Overview
- Quick Start
- Tasks
- References

Figure 53 shows an example on how the help volume could be organized for this application.

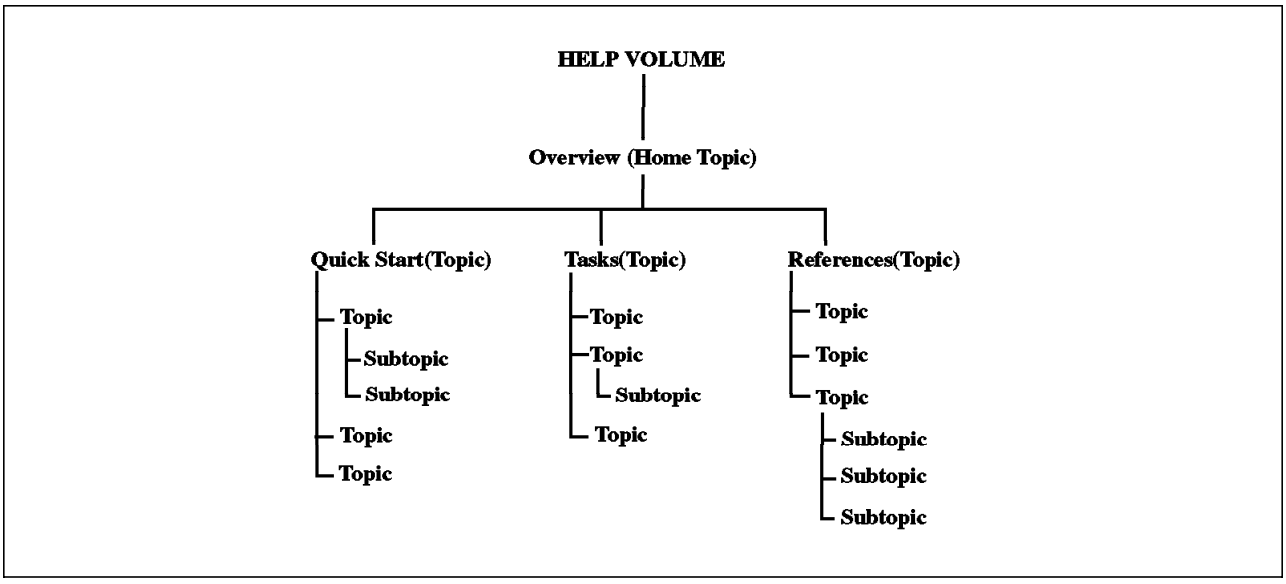


Figure 53. Help Volume Organization

The Help Manager allows you to navigate through the different topics and volumes via hyperlinks in the help information. For information about hyperlinks refer to 10.1.4, “Using Hyperlinks” on page 156.

This chapter will explain how to use the Help Manager.

10.1 Using the Help Manager

The desktop Help Manager provides online help information for the desktop and its facilities. Additionally, other applications installed on your system may use the desktop Help Manager instead of providing their own help facilities.

10.1.1 Accessing the Help Manager

The Help Manager can be accessed through:

- The help Manager control on the front panel or its subpanel
- Pop up menus of the front panel, subpanels or controls
- The Help menu in the menu bar in the application or tool
- The F1 key (help key) in an application or tool

Figure 54 shows the Help Manager front panel control, the help subpanel and the help menu from the menu bar of an application's interface.

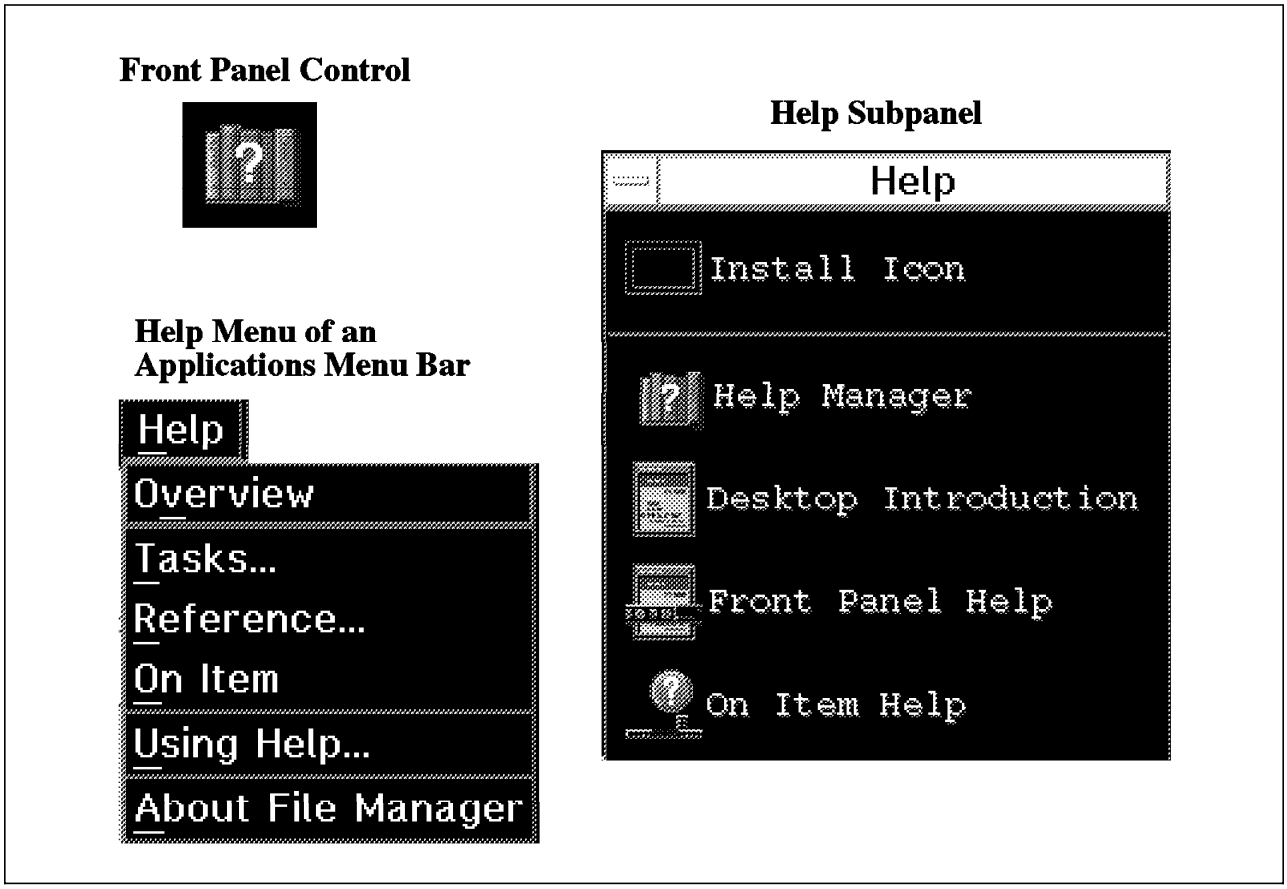


Figure 54. Some Different Options for Accessing Help

Selecting the Help Manager control in the front panel will bring up the Help Manager's top level interface. This interface is shown in Figure 55 on page 154.

The subpanel located above the front panel help control displays several different help options:

Help Manager

This option displays the top level interface of the Help Manager. This option is identical to selecting the help control in the front panel.

Desktop Introduction

Brings up the Help Manager directly on the topic: *Introducing the CDE Desktop*.

Front Panel Help

Displays a Help Manager interface containing the topic: *Front Panel Help*

On Item Help

This help option is an interactive facility that will supply the available help for a specific application component. To get on item help you will have to:

1. Choose **On Item Help** from the help subpanel. The pointer's shape will change to a ? (question mark).
2. Move the pointer over the component you require help on and click the left mouse button.

This will bring up the Help Manager directly on the help information for the selected component. This function is only available if the application's help volume supplies a help topic for this component. If a help topic does not exist for this component, an error message will be displayed.

The help pull down menu of an application's menu bar accesses help information related to the application you are using. If your application supports the desktop Help Manager the help pull down menu will look like the example menu shown in Figure 54 on page 152. The different topics on the menu are described below:

Overview

Displays the home topic for the application you are using. The home topic is the first topic of the application's help volume hierarchy. Figure 53 on page 151 shows an example of how an application's help volume is organized.

Tasks...

Displays a Help Manager interface with all available task instructions of the application.

Reference...

Displays a Help Manager interface with all available reference topics for the application, arranged into various categories.

On Item

This option supports the same type of help as the On Item Help option in the help subpanel. Please refer to the above description of On Item Help.

Using Help...

Provides help on how to use help. It will present a Help Manager interface, which contains the home topic of the Help Managers help volume.

About *Application*

Displays the version and copyright of the application.

10.1.2 Using the Help Manager Interfaces

No matter which way you choose to access the desktop help system, the Help Managers interface will appear.

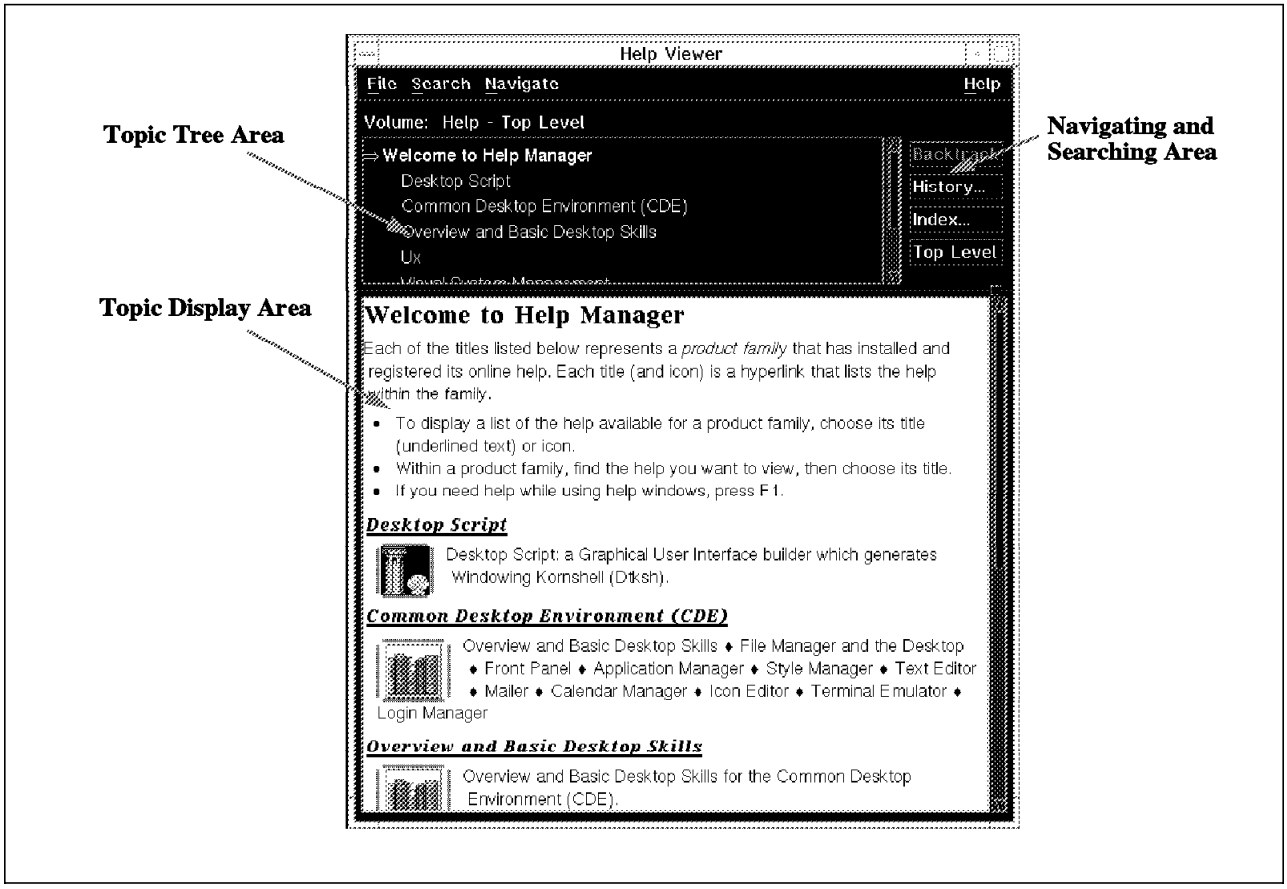


Figure 55. Help Manager Interface. This interface shows the top level of the Help Manager.

As shown in Figure 55 the Help Manager interface has several sections.

- **Topic Tree Area**
  The topic tree area lists the topic tree with its topics and subtopics. The current topic, which is displayed in the topic display area, is highlighted and has the ⇒ symbol. at the beginning of its row. If you click on another topic, this new topic will be displayed in the topic display area.

  The title of the topic tree area shows you the help volume name to which the current help topic belongs.

- **Navigating and Searching Area**
  This area presents four buttons:

  – **Backtrack**
    Gives you the opportunity to return to the help topics you have visited in reverse order.

  – **History...**
    Displays the help history browser interface. This interface is shown in Figure 56 on page 155. It lists the help topics you have visited during your current access of the Help Manager, sorted by the help volumes.

Figure 56. History Browser

The upper section displays all the previously visited help volumes. The lower section of the interface displays the previously visited help topics of the current help volume. The current help volume is highlighted in the upper section. If you want to return to a topic in another help volume, follow the steps below:

1. Click on the desired help volume in the upper section of the help browser interface. The visited topics of this help volume will be displayed in the lower section.
2. Click on the title in the lower section of the topic you want to be displayed.

The chosen topic will be displayed in the Help Manager interface.

- **Index...**
  Gives you the opportunity to search for an entry or a topic in the Help Manager index. Clicking on this button will display the index search interface shown in Figure 58 on page 158. See 10.1.5, “Searching for an Entry or a Topic” on page 157 for more information about this tool.
- **Top Level**
  Pressing this button will cause the Help Manager to display its top level topic shown in Figure 55 on page 154.

- **Topic Display Area**
  This section displays the contents of the current help topic including text, graphics and hyperlinks.

10.1.3 Moving Between Different Topics

There are several ways to move between different help topics.

- If the help topic you want to move to is displayed in the topic tree area you can double click on its row. The topic will be displayed in the topic display area.
- You can also use the tools in the navigating and searching area to switch to another topic.

- If you have already visited the topic during your current Help Manager session. You can use either the **Backtrack** or the **History...** buttons in the navigating and searching area. For more information about **Backtrack** and **History...** refer to 10.1.2, “Using the Help Manager Interfaces” on page 154.
- To search for a specific topic, use the **Index...** button in the navigating and searching area. See 10.1.5, “Searching for an Entry or a Topic” on page 157 for more information about searching for help.

• The hyperlinks in the topic display area are another way that you can to move to another topic. See 10.1.4, “Using Hyperlinks” for more information on hyperlinks.

10.1.4 Using Hyperlinks

Hyperlinks are active elements imbedded in the text which perform activities. They appear as underlined text or as a graphical object. Most hyperlinks jump to a related topic. Most of them take you deeper into the topic hierarchy by displaying subtopics. Other hyperlinks may jump across the hierarchy to display related topics in other help volumes. Some hyperlinks even execute commands instead of jumping to another topic.

Figure 57 on page 157 shows a help topic containing and describing all the different types of hyperlinks. This topic belongs to the help volume: Help and is a subtopic of the help topic: Concepts.

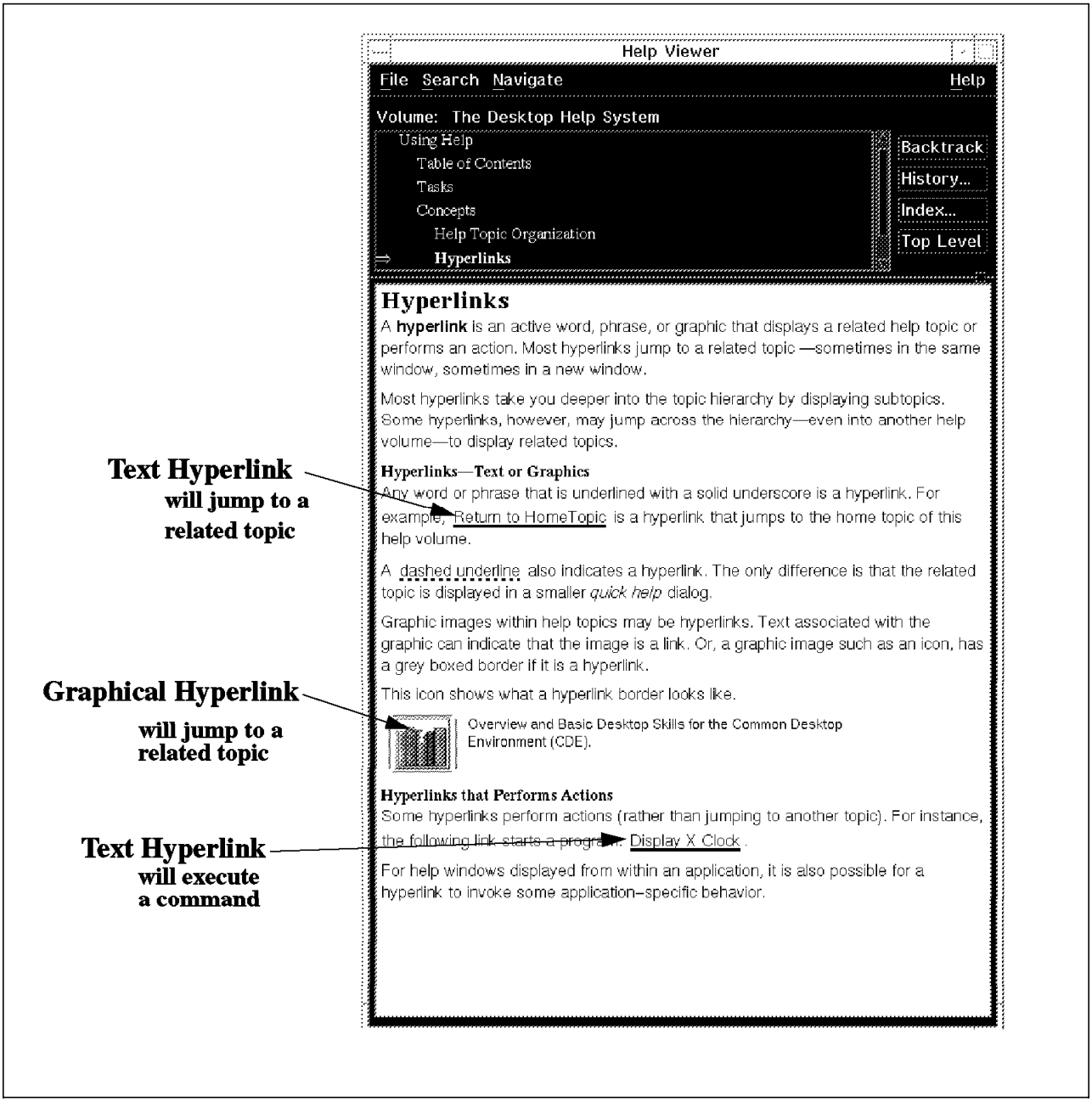


Figure 57. Different Types of Hyperlinks

10.1.5 Searching for an Entry or a Topic

The Help Manager includes an index search tool. You can access this tool by clicking the **Index...** button in the navigating and searching area. The index search interface is shown in Figure 58 on page 158.

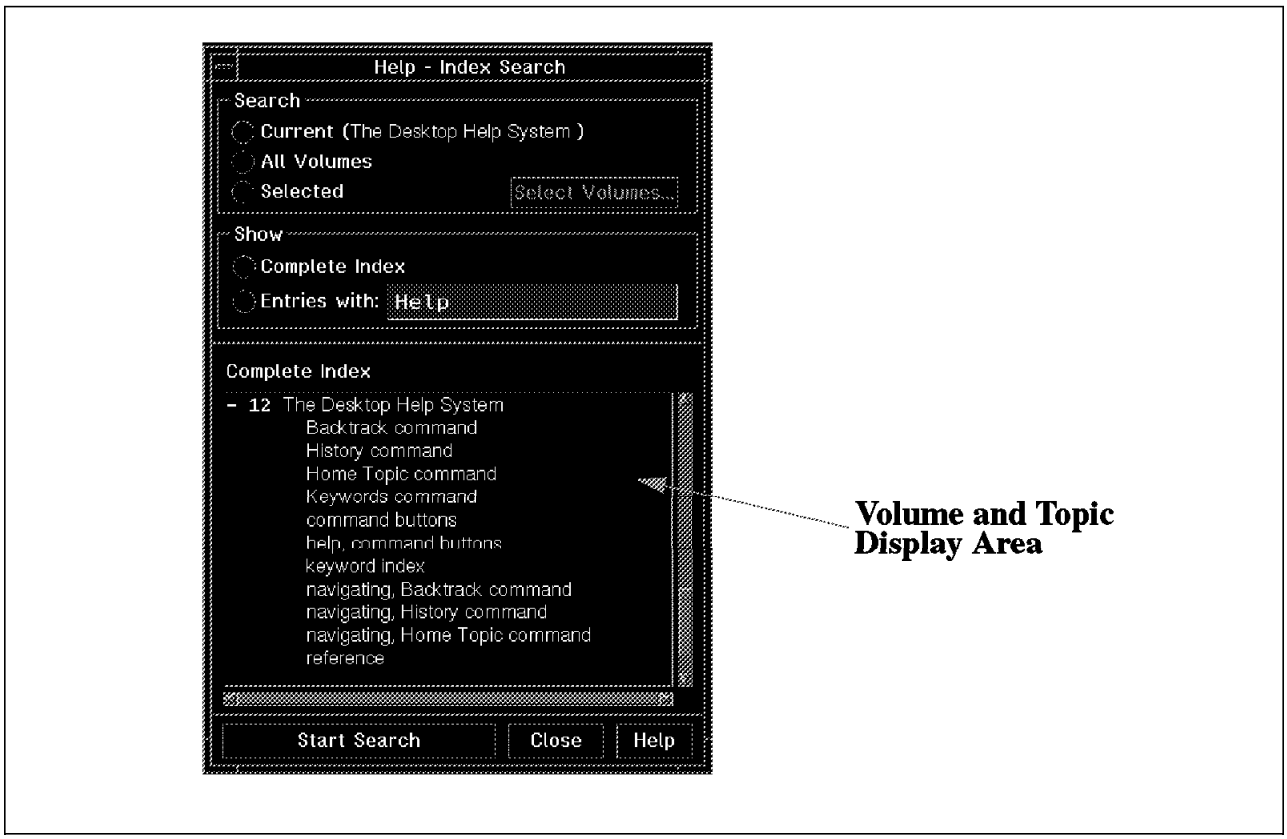


Figure 58. Index Search Tool

This interface has three different sections:

- **Search:**
  In this section you will specify the volumes which will be searched. The different selections are:
  - **Current:** This selection will search only in the current volume. The current help volume name is displayed in brackets.
  - **All Volumes:** This section will search in all volumes available to the desktop Help Manager.
  - **Selected:** This section will search in selected help volumes. To select the volumes for a search:
    1. Click on the **Select Volumes...** button. (This option is only available if the **Selected** button was pressed to initiate the search.) An interface will pop up which allows you to select the volumes to search, from the available volumes list.
    2. Select the desired volumes. Each selected volume will be highlighted.
- **Show:**
  - **Complete Index:** This section will display all the help topics in the volumes specified in the Search section.
  - **Entries with:** This section can be used to search for a topic or entry that contains the specified string in its title.
- **Volume and Topic Display Area:**
  The title of this area will change during a search.

All the volumes which had been found by the search will be represented in this section. The number in front of the volume name tells how many matching topics had been found in the listed volume. If you click on the volume, the topics which match the search will be displayed.

To view one of the topics you found by the search, click on the topic in the list of found topics. The index search interface will remain the desktop so that you can change to another topic found during the search without searching again.

For example, suppose, you want to search for all entries with the word help in the title. Do the following:

1. Bring up the index search interface by clicking the **Index...** button in the Help Manager interface.
2. Click on the **All Volumes** button in the Search area. This specifies, that you want to search in all available help volumes.
3. Click on the **Entries with:** option in the Show area.
4. Enter the search word *help* in the text field.
5. Start the search by clicking on the **Start Search** button at the bottom of the interface.
6. When the search finishes, the scrolled list in the interface will list all volumes which contain the word *help*.
7. To see all of the topics in a volume that was matched during the search, click on the desired volume name in the list. All Topics matching the search criteria *help* will be listed If there are any subtopics available, a number will be displayed at the beginning of the topics row.
8. Select the desired topic you want to read.

10.1.6 Printing a Help Volume or Topic

At this time the Desktop does not support printing a help volume or topic.

10.2 Adding Your Own Help Volume or Changing an Existing Volume

In this release of the desktop it is not possible to implement your own help volumes or change existing volumes.

10.3 Relationship Between the Help Manager and InfoExplorer

The Help Manager and InfoExplorer are totally independent. InfoExplorer is a book browser. It was not designed to be used as a help system. The desktop Help Manager is a context sensitive help system for AIX. Both tools are part of AIX. They both have their advantages and disadvantages. If you are using the desktop Help Manager you may occasionally see a hyperlink call an entry in the InfoExplorer database.

Chapter 11. Icons

Icons are used extensively throughout the desktop to represent different kinds of objects. For example:

- Actions
- File types
- Front panel controls
- Subpanel controls
- Minimized application interfaces

The desktop uses only four different icon sizes. For example, an icon in the front panel is the largest sized icon. The same icon displayed in a front panel subpanel, is smaller sized version of the icon in the front panel. The desktop also uses colored icons, called pixmaps, on color capable hardware and black and white icons, called bitmaps, on monochrome systems.

If a particular icon is to be used anywhere in the desktop and on various hardware platforms (color and monochrome) the icon has to have eight different versions of itself. There must be four different sizes of the icon in both color and black and white. For more information about the sizes and the naming conventions of desktop icons see, 11.2, “Desktop Icon Name and Size Conventions” on page 163, 11.3, “Re-Sizing Icons” on page 164 and Table 11 on page 164.

The desktop is supplied with many built-in icons which can be used for your applications and tools. If you are unable to use one of these, an icon editor is provided to allow you to create your own icons. The following sections in this chapter will describe how to create your own icons, how to name and size your icons for use by the desktop and how to use the built-in icons.

11.1 Creating Your Own Icons

The desktop includes an icon editing tool that gives you the ability to create your own icons. You may want to create icons to:

- Represent your actions with their own meaningful icon. See Chapter 8, “Launching Applications From the Desktop” on page 121 for information about actions.
- Use a different icon for one of the controls in the front panel than one of the supplied default icons. See Chapter 4, “Front Panel” on page 31 for information about front panel controls.
- Represent the data types you define with unique meaningful icons. See 8.3, “Data Types” on page 124 for information about data typing.

The tool you will use to create icons is called the icon editor. You can find an action which launches the icon editor in several places:

- The personal applications subpanel.
- The DesktopTools group of the Application Manager.
- The CreateAction tool represented as the Edit Icon button.

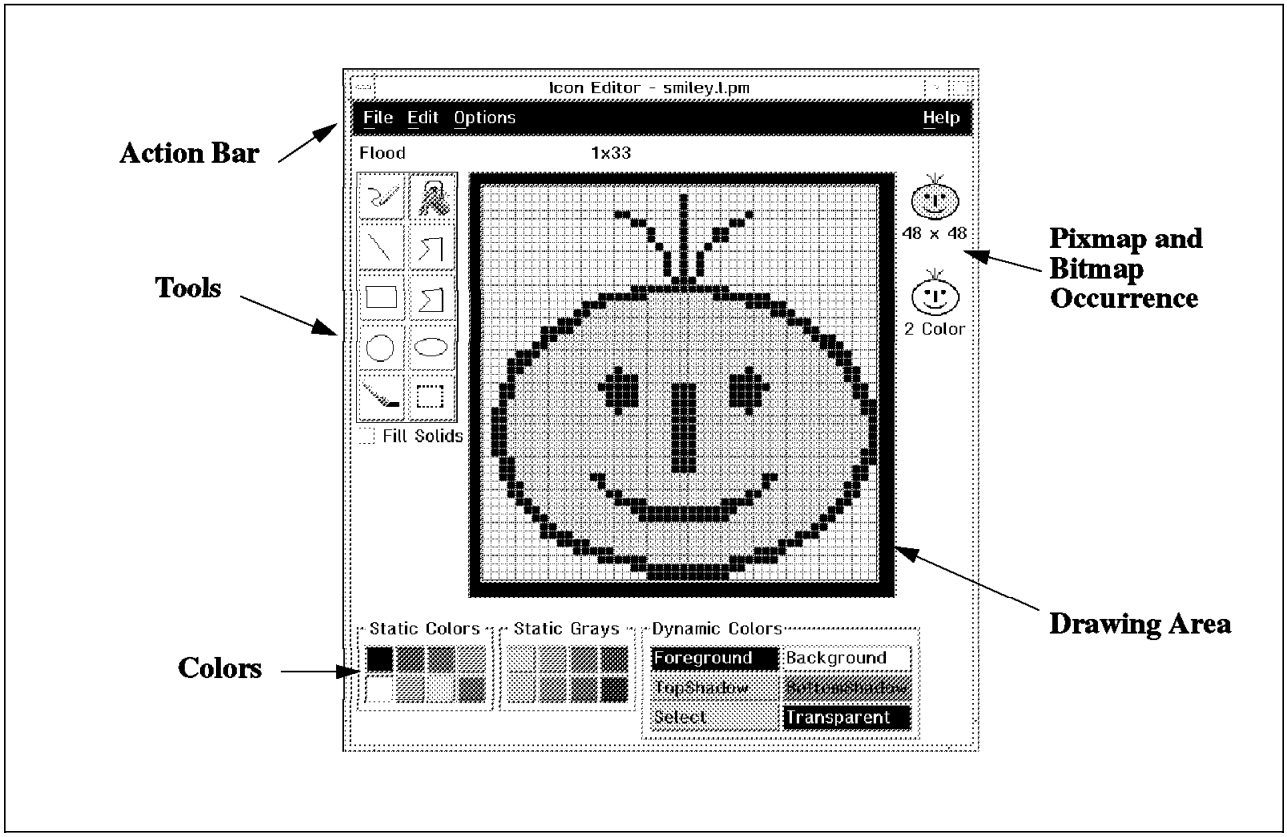


Figure 59. Icon Editor

The icon editor interface has several different areas. These different areas are shown in Figure 59 on page 162.

The icon editor tool is a very to understand and use. Its interface and functionality is very similar to the various medium to low functionality paint and graphic editing tools that are widely available. Information on how to use the icon editor can be found in: *AIXWindows Desktop Advanced User's and System Administrator's Guide*, chapter 11: Creating Icons for the Desktop.

Let's look and an example on how to create an icon using the icon editor:

In this example we will create the icon called SMILEY shown in Figure 59. The size of this icon will be 64x64 pixels. This is the default size used by the icon editor. Unfortunately the default size of the icon editor is not one of the icon sizes used by the desktop. However, we can create the icon at the 64x64 size and then re-size it for use by the desktop. The other option is that we can tell the icon editor to reduce the icon size to the needed size before we begin drawing. In this example, we will go ahead and build it at 64x64 and then re-size it.

1. Open the icon editor by clicking on the subpanel control: **Icon Editor**. You will find this on the personal applications subpanel above the text editor icon in the front panel.
2. The Icon Editor when initialized, will only display part of the icon drawing area. It is best at this time to use the re-size handles on the window to re-size the icon editor interface to get the whole drawing area visible.

   You are now ready to start creating SMILEY.

3. To draw the outline of SMILEY's face pick up the **Ellipse** tool and use it to draw an ellipse in the lower middle of the drawing area.
4. Fill it with the color yellow by:
   - Clicking on the **Fill** (pouring paint can) tool
   - Pick up the Color yellow from the box labeled **Static Color**
   - Click in the middle of the ellipse that you drew
5. To create the eyes:
   - Pick up the **Circle** tool by clicking on it
   - Draw the eyes
   - Fill them with the black also from the **Static Color** box
6. To create the nose:
   - Pick up the **Rectangle** tool
   - Draw the nose
   - Also fill it with black
7. Draw the mouth and the hair by:
   - Picking up the **Pencil** tool
   - Still using black draw them as you like
8. If needed, you can change single pixels in the icon by using the **Pencil** tool. Be sure that you first select the desired color before you begin changing pixels.
9. To erase areas you can use the **Erase** tool. With this tool each click in the drawing area will erase a 3x3 pixel area.
10. Your last step will be to save the icon you created:
    - Select the **File** option from the menu bar.
    - Select **Save As...** from the shown pull down menu. This will cause the save dialog window to pop up:
      – In the **Directory** field, change the directory path to $HOME/.dt/icons
      – Enter the filename: SMILEY in the **Save File As** field.
      – Click on the **OK** button to save the icon.

You have now created the SMILEY icon. To use this icon in the desktop it has to be re-sized to one of the supported sizes and named using the desktop naming conventions. Instructions and information on how to do this can be found in 11.3, "Re-Sizing Icons" on page 164 and 11.2, "Desktop Icon Name and Size Conventions."

11.2 Desktop Icon Name and Size Conventions

The desktop supports four different icon sizes for different representations. For example the icons in the front panel are large-sized icons and the ones in the subpanels are small-sized icons. There are also colored icons, called pixmaps (XPM format), as well as black and white icons, called bitmaps (XBM format). If an icon needs to be used anywhere in the desktop on multiple hardware platforms (color and monochrome), the icon must have eight different icon files

for itself. One for each of the four sizes and in both color (pixmap) and monochrome (bitmap).

Note: There is a difference between bitmap and monochrome color icons. A pixmap can be only two colors (black and white). This is still a pixmap. A bitmap is only two colors (black and white) and is saved in a different format (XBM). These are the icons that should be saved with the .bm extension listed below.

To keep all these icons straight, the desktop uses a naming convention. The base name of the icon can be any valid AIX file name. The naming convention is actually a specific extension for each size and color type.

Table 11. The Icon Sizes and Their Naming Conventions

	Pixmap Icons	**Bitmap Icons**
16x16 (tiny)	*iconname*.t.pm	*iconname*.t.bm
24x24 (small)	*iconname*.s.pm	*iconname*.s.bm
32x32 (medium)	*iconname*.m.pm	*iconname*.m.bm
48x48 (large)	*iconname*.l.pm	*iconname*.l.bm

For example, SIMLEY.l.pm is the 48x48 pixmap version of the SMILY icon. Similarly, SMILEY.m.bm is the 32x32 bitmap version of SMILEY.

These different icon files are used by different desktop components. Table 12 shows when and where they are used. The resolution columns indicate the display adapter resolution.

Table 12. Usage of the Different Icon Files

Desktop Component	**High Resolution**	**Medium Resolution**	**Low Resolution**
File Manager and Application Manager (view by name and icon)	medium	medium	medium
File Manager and Application Manager (view by name and small icon)	tiny	tiny	tiny
Front panel controls	large	large	medium
Front panel subpanels	medium	medium	tiny
Front panel switch controls	small	small	tiny
Minimized windows	large	large	medium
Files on desktop backdrop	medium	medium	medium

11.3 Re-Sizing Icons

To re-size your icons to meet the desktop size requirements, it is recommended that you first start with the the largest size that you are going to need. This is the easiest procedure, because after scaling down you will only have to do minor modifications to the icon itself.

To re-size the icon:

1. You must first scale the area of your current icon to the new size that you want.

2. You must then re-size the icon to the desired new size. Keep in mind that you are going to create a new icon with this procedure, so you have to save it with a different name. See Table 11 for the different naming conventions required by the desktop.

For example, let's start with our 64x64 colored icon called SMILEY and re-size it to get all four sizes needed by the desktop. This icon was created in 11.1, "Creating Your Own Icons" on page 161.

1. Open the icon editor with the icon SMILEY by dragging its icon from the File Manager and dropping it on the **Icon Editor** control on the personal applications subpanel located above the text editor icon in the front panel.
2. Select the area you want to re-size. Normally this will be the complete icon but you can choose only a part if you wish. You will do this selection by:
   a. Clicking on the select tool, which looks like a dotted rectangle in the lower right corner of the tools area in the icon editor
   b. Selecting the area to be re-sized by clicking and holding the left mouse button while the mouse pointer is over the upper left hand corner of the desired area. Then drag the mouse to the lower right hand corner of the area and release the mouse button.
3. Pull down the Edit menu bar option and select **Scale Area**.
4. You can now select the size you want the icon to be scaled to. The easiest way to do this is to move the mouse pointer into the upper left corner of the drawing area. Click and hold the left mouse button down. Then drag the mouse to the lower left corner of the new size. The coordinates of the mouse pointer will be displayed above the drawing area. You can use these to see the size of the area as you drag the mouse. For example, for the large icon it would be 48x48. When you have reached the right size release the mouse button.
5. A movable rectangle will appear to follow your mouse in the drawing area. This is the re-sized icon. Position this rectangle in the upper left corner of the drawing area and click the left mouse button.
6. Select the **Resize Icon...** option from the **Edit** menu bar option.
7. A window will pop up. Enter the desired size of your new icon in the text field of the window. The icon editor will re-size the icon to the specified size. This should eliminate the remains of the original icon that were not overlaid by the above scaling function.
8. As the last step, save the icon with a new name matching the desktop naming conventions. See Table 11 on page 164 for the different naming conventions required by the desktop.

11.4 Using Created Icons

You can use the icons you create for your applications, tools or actions by either:

- Locating them in the directory $HOME/.dt/icons. This path is by default known by DTICONSEARCHPATH environment variable.
- Modifying DTICONSEARCHPATH environment variable to point to the location of your icons.

For information on the DTICONSEARCHPATH see 11.8, “The DTICONSEARCHPATH Environment Variable” on page 168.

11.5 Using Built-In Icons?

You may want to use one of the built-in or system administrator supplied icons as your icon. Or you can start with one of them and modify it as needed. The DTICONSEARCHPATH path can be followed to find all of the available icons. See 11.8, “The DTICONSEARCHPATH Environment Variable” on page 168 for more information about the location of the built-in icons.

To change a built-in icon, follow these steps:

1. First copy it to your /$HOME/.dt/icons directory
2. Change it by using the icon editor
3. Save this icon with a new name

11.6 Using Icons from Other Products

The icons used by the desktop are in XPM or XBM format. If you have icons from other systems or products, that are either XPM or XBM format, they can be used, but must meet the sizing and naming conventions of desktop. See 11.2, “Desktop Icon Name and Size Conventions” on page 163 and 11.3, “Re-Sizing Icons” on page 164 for more information about the desktop sizing and naming conventions.

Icons from the previous version of the desktop (XDT3) can be converted by using the XDT3 conversion tool provided with the desktop. For more information about this tool refer to the document: *AIXwindows to AIXwindows Desktop Migration Guide*.

All other icons will have to be converted by using an appropriate tool or they can be captured directly from the screen. This is done by using the Grab Screen Image facility of the icon editor and saving it in either XPM or XBM format.

To capture an icon from a screen:

1. Display the icon you want to capture on the screen, by using an appropriate tool.
2. Open the icon editor.
3. Select **Grab Screen Image** from the Edit menu bar option.
4. The mouse pointer will change to a cross shape. Use this to draw a rectangle around icon you want to grab.
5. The icon will now show up in the drawing area of the icon editor.
6. You can now edit it with the icon editor, if desired.
7. Size the icon as needed. See 11.2, “Desktop Icon Name and Size Conventions” on page 163 and 11.3, “Re-Sizing Icons” on page 164 for information about icon sizes.
8. Save it with a name that follows the desktop naming conventions shown in Table 11 on page 164.

11.7 Using File Manager to Browse Icons

The desktop provides a tool which enables the File Manager to:

- Display the contents of an icon file instead of the generic pixmap icon
- Invoke the icon editor by double-clicking on the icon's object

Figure 60 is an example of this facility.

To enable this facility:

1. Copy the /usr/dt/examples/types/IconBrowse.dt file into your $HOME/.dt/types directory
2. Select **ReloadApps** from the DesktopTools folder in Application Manager

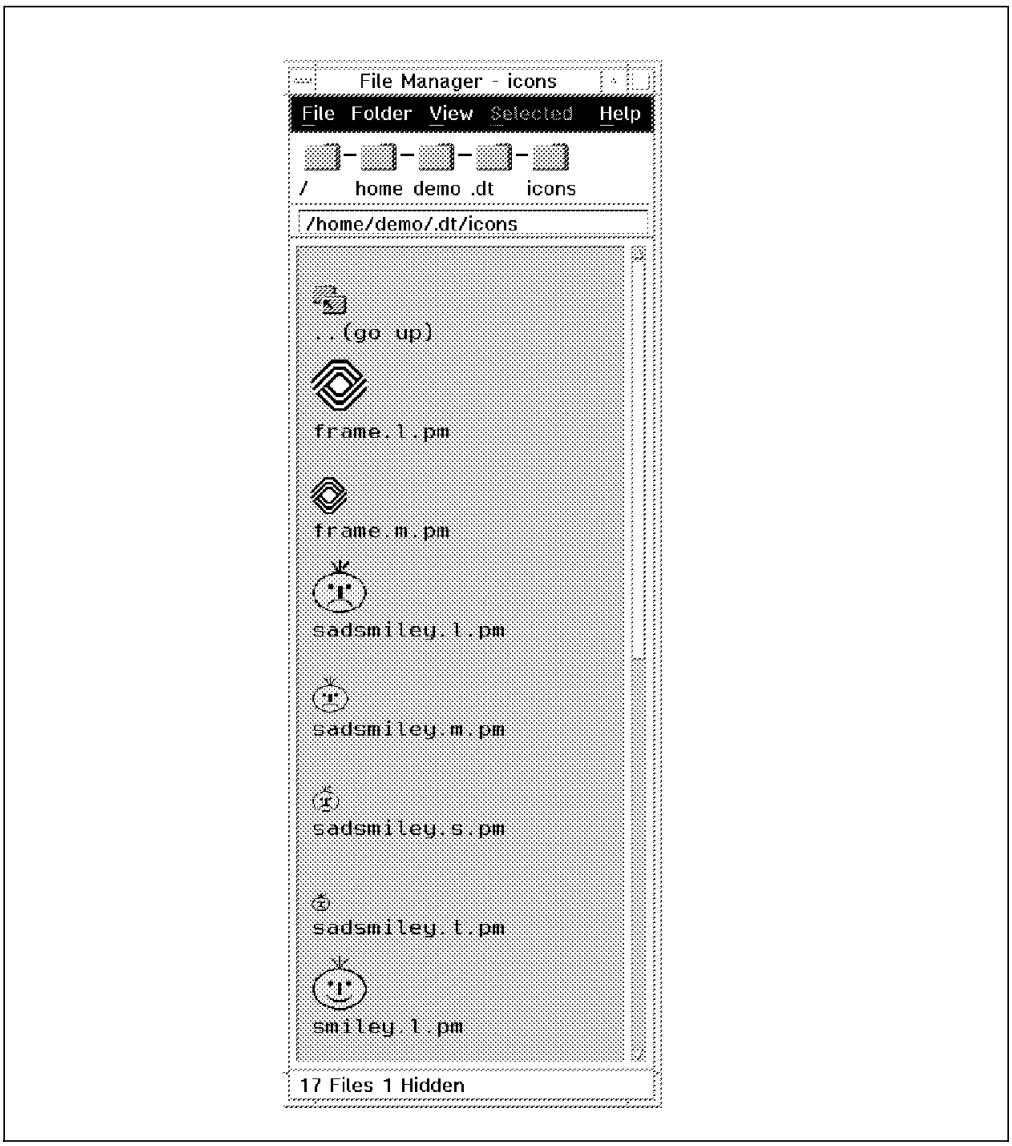


Figure 60. The File Manager With the Icon Browser Facility Enabled

Every icon file which matches the desktop naming conventions for icons will now be represented in the File Manager by its icon. Use this facility only when needed, because the desktop will significantly slow down while this function is enabled.

Note: It is possible that the desktop may run out of available colors if the directory being displayed contains a lot of colored icons with many different colors. If this happens the desktop will use the black and white version of the icon instead. This can be avoid by limiting the number of colors you use when you create your icons.

To disable this facility:

1. Delete the $HOME/.dt/types/IconBrowse.dt file.
2. Select **ReloadApps** from the DesktopTools folder in Application Manager.

11.8 The DTICONSEARCHPATH Environment Variable

The desktop uses the DTICONSEARCHPATH environment variable to locate icons. There are actually two environment variables:

DTICONSEARCHPATH	This variable contains the search path for icons on color displays
DTICONBMSEARCHPATH	This variable contains the search path for icons on monochrome displays

By default the directories for both variables are:

Personal icons
$HOME/.dt/icons

System wide icons
/etc/appconfig/icons/$LANG

Built-in Icons
/usr/appconfig/icons/$LANG

The desktop uses the following search order between the two variables:

- For color displays, the desktop first searches for a pixmap icon using the DTICONSEARCHPATH variable. If no pixmap icon is found, it will follow the DTICONBMSEARCHPATH variable to try and locate a bitmap icon instead.
- For monochrome displays, the desktop will follow the DTICONBMSEARCHPATH variable to try and locate a bitmap icon. If none is found, the desktop will search for a pixmap icon using the DTICONSEARCHPATH variable and if found, use the monochrome version of the pixmap icon.

For more information on search paths in general see 12.6, “Search Path Environment Variables” on page 175.

Chapter 12. Administering the Desktop

This chapter will cover:

- Installing or de-installing the desktop
- The file structure of the desktop
- Migrating desktop configurations to another host
- Configuring an Xstation for use with the desktop
- Printing from the desktop
- Search paths used by the desktop
- Using the desktop in a networked environment

12.1 Installing or De-installing the Desktop

If you are installing the desktop on an existing AIX 4.1.1 system use the Software Installation and Maintenance option of SMIT.

If you are installing the desktop as part of a new AIX 4.1.1 system installation, select the X11 bundle as part of the software installation task. The X11 bundle on your installation media contains all the software packages for the desktop. The packages with names that begin with X11.DT are the desktop software packages.

To de-install the desktop use the Software Installation and Maintenance option of SMIT.

12.2 Directories Containing the Desktop Software

After the installation of the desktop you will find several directories called *dt* in your directory tree. The main directories are:

- /usr/dt
- /etc/dt

These directories were created during desktop installation.

12.2.1 Contents of /usr/dt

The /usr/dt directory contains all the desktop executables along with the default definitions and configuration files. These files must not be changed. If the system administrator wants to customize the configuration files on a system wide basis, the necessary configuration files must be copied to the /etc/dt directory. The actual location of the customized version of each file is mentioned in the beginning comments of the desired configuration file.

Note: Making a copy of the configuration files in the appropriate /etc/dt directory ensures that any changes that you make to these files will be retained between desktop version upgrades.

For more information on the sub-directories of /usr/dt see Appendix A.3, “/usr/dt” on page 192.

12.2.2 Contents of /etc/dt

The /etc/dt directory contains locally customized versions of the desktop configuration and definition files. The configurations and definitions in the sub-directories of /etc/dt apply system wide and can only be changed by the system administrator.

If you want to customize any of the default desktop configuration or definition files on a system wide basis, you must first copy the desired configuration file from the /usr/dt sub-directory to the appropriate directory in /etc/dt. Each default configuration file contains a comment which tells you where to copy the file to.

For information about the sub-directories of /etc/dt see Appendix A.2, “/etc/dt” on page 192.

12.2.3 Contents of the User′s Home Directory

After a user first logs on to the desktop they will find a file called .dtprofile and a directory called .dt in their home directory.

The purpose of the .dtprofile file is similar to the .profile file in the user′s home directory. During a desktop login, .dtprofile is read and executed just like the .profile file was in the past.

There are reasons that you may want to have both the .dtprofile and the .profile files. One of the possible reasons might be that you do not always use a desktop capable workstation to log on to the system. When you do not log on to the desktop, the .dtprofile file will not be executed, instead only your .profile will be executed. Maintaining both the .dtprofile and .profile files can be a real hassle. To avoid this hassle, you can force the desktop to read your .profile in addition to your .dtprofile. This is done by simply uncommenting a line in your .dtprofile. If you do this, you may have to change entries in your .profile. For information on what to change in your .profile and how to enable the reading of your .profile, see the comment at the beginning of your .dtprofile file.

The directory .dt contains user specific desktop definitions and configurations. For information abut the sub-directories of $HOME/.dt see Appendix A.1, “Home Directory” on page 191

12.2.4 Precedence of the Different Definitions and Configurations

If a user has their own definitions or configurations in $HOME/.dt. These definitions or configurations have a higher priority than the system wide definitions in /etc/dt. Definition or configurations in /etc/dt have a higher priority than the default definitions or configurations in /usr/dt. There is one exception to this rule. You can specify that certain front panel resources are locked and can not be overridden by user supplied definitions. For more information on this, see 4.2.5, “Preventing Changes to a Control and or a Subpanel” on page 56

12.3 Migrating Desktop Customizations to Another Desktop Host

You may want to have the same the desktop configurations and definitions you have on one host, as on another. You can either do this on a system wide basis, if you have root access on the desired new host, or as a user for your own environment.

If you, as a system administrator, want to copy the whole the desktop configuration you have on one host onto another, you need to move the files in /etc/dt from your original host to the desired new one.

Similarly, if a user wants to get their environment onto another host, the files in their $HOME/.dt directory and the $HOME/.dtprofile must be copied to the new host. If there are any system wide changes made by the system administrator that are not on the new host, the user can use them on the new host by copying them to their $HOME directory on the new host.

For more information about the special files and their function see Appendix A, “The Desktop Directories and Files” on page 191

12.4 Configuring an Xstation to Use the Desktop

If your Xstation supports the XDMCP protocol, you can use the standard Xstation configuration procedure to configure it to use the desktop. For example, if you are using IBM Xstations:

- Make sure that the desktop is installed and running on the system that you want to use as your Xstation server. In desktop terminology this system is called the session server.
- On the session server configure your Xstation using SMIT. On the Add an Xstation panel make sure you set:
  1. XDMCP mode to direct.
  2. xdmcp HOST to the hostname of your session server.

With this set, the Xstation will contact the session server through the XDMCP protocol and request a log in screen. The session server will send the desktop Login Manager screen to the Xstation.

If your Xstation was already configured to use the XDM login from AIXwindows before you installed the desktop on the server that boots your Xstation, your Xstation will receive the desktop log in screen without any configuration changes. In fact the desktop Login Manager is based on XDM (version for X11R4) and uses the same XDMCP protocol.

In this release of the desktop, the Login Manager doesn′t support the XDMCP indirect mode, which is supported by XDM version for X11R5. So if your Xstation was configured to use this mode, you have to change the configuration to use the direct mode, as shown above.

If your Xstation doesn′t support the XDMCP protocol, you can access the desktop with the following procedure:

:i1/etc/dt/config/Xservers :i1/usr/dt/config/Xservers :i1/etc/dt/config/Xconfig :i1/usr/dt/config/Xconfig

- If the file /etc/dt/config/Xservers doesn't exist, copy it from /usr/dt/config/Xservers using the following command:

  ```
  cp /usr/dt/config/Xservers /etc/dt/config
  ```

- If the file /etc/dt/config/Xconfig doesn't exist, copy it from /usr/dt/config/Xconfig using the following command:

  ```
  cp /usr/dt/config/Xconfig /etc/dt/config
  ```

- Edit /etc/dt/config/Xconfig and change the Dtlogin*servers line to:

  ```
  Dtlogin*servers: /etc/dt/config/Xservers
  ```

- Edit /etc/dt/config/Xservers and add a line for your Xstation, like:

  ```
  hostname:0 [class] foreign
  ```

  Where hostname is your Xstation's hostname and class is an optional class name for your Xstation, such as NPD200X.

These changes will take effect after the next reboot, or after the Login Manager receives a SIGHUP signal.

For more information on the Login Manager see Chapter 2, "Login Manager" on page 5.

For more information on how to configure Xstations see *AIXWindows Desktop Advanced User's and System Administrator's Guide* or the *The IBM Xstation Handbook*.

12.5 Printing From the Desktop

In the front panel you will see the printer control on the right side of the workspace switch. This control is used to print files and access the printer status application.

To send a file to a printer you have several choices:

- If you have a file you want to print, drag the object that represents the file to be printed from the File Manager and drop it on the printer control in the front panel. This sends the file to the desktop default printer.
- If you want to send the output to a printer other than the desktop default printer, pop up the subpanel located above the printer control in the front panel and drop the object which represents the file to be printed on one of the other printer objects in the subpanel.
- Open the Application Manager and select the **Printers** group. All the printers which are known to the desktop will be presented as objects in this folder. You can drag the object that represents the file to be printed and drop it on one of these printer objects.

Note: If the desktop does not have any printers defined or the desired printer is not listed, go on to 12.5.1, "Integrating a New Printer into the Desktop" on page 173

12.5.1 Integrating a New Printer into the Desktop

You can access the desktop default printer through the printer control in the front panel. If you want to integrate other print queues into the desktop, you can use the dtprintegrate command to specify a mapping between the desktop and the desired AIX printer queue.

The following procedure shows how to integrate either local or remote printers into the desktop. You must have root access to perform this procedure.

1. Make sure that the print queue that you want to integrate is known to the system. To list the defined print queues, you can use the command:

   ```
   qchk -A
   ```

   If the print queue you are looking for is not defined, you will have to add it using the standard procedure for adding an AIX print queue with SMIT.

2. Execute the dtprintegrate command using the following syntax:

   ```
   dtprintegrate -p <printer_name> [ -i <icon_file>]
                                   [ -d <destination_name>]
   ```

   where

   -p printer_name

   This parameter is not optional. The printer_name is the name used to label the printer's object in the Printers folder in the Application Manager, or on the personal printers subpanel. If you specify a name that is not the printer queue name known to the AIX printing subsystem, you must also specify the -d parameter to specify the destination_name.

   -i icon_file

   This is the name of the file containing the icon for the printer. If the icon file is located in the icon search path, you can use the file name only. Otherwise, you must use the absolute path. The default icon search path contains the following directories:

   ```
   /$HOME/.dt/icons
   /etc/dt/appconfig/icons/$LANG
   /usr/dt/appconfig/icons/$LANG
   ```

   For more information on the DTICONSEARCHPATH see 11.8, "The DTICONSEARCHPATH Environment Variable" on page 168.

   -d destination_name

   Specifies the queue name known to the AIX printing subsystem. This parameter is required if the name specified on the -p parameter is not the same name as the AIX print queue.

   For example:

   To integrate a printer titled PostScript, whose AIX queue name is psc and uses the icon MyPrinter into the desktop. Issue the following command:

   ```
   dtprintegrate -p PostScript -i MyPrinter -d psc
   ```

   The dtprintegrate command will create the following two files:

   - /etc/dt/appconfig/types/$LANG/<printer_name>.dt
   - /etc/dt/appconfig/appmanager/$LANG/Printers/<printer_name>

For more information about dtprintegrate, including additional command-line options, see *AIXWindows Desktop Advanced User's and System Administrator's Guide*, chapter 6.

The icon for the new printer should now appear in the Printers folder in the Application Manager. If it does not appear, log out and then log back in.

To add a printer to the printers subpanel, drag the printer icon from the Printers folder and drop it onto the Install Icon area on the printers subpanel.

12.5.2 Changing the Desktop Default Printer

The desktop has a concept of a default printer that is used to print output when the user:

- Drops an object that represents a file to be printed onto the printer control in the front panel. This assumes that you have not copied a different printer from the printers subpanel into the front panel
- Selects the print action for an object in the File Manager and uses the default options on the print dialog.
- Drops an object that represents a file to be printed onto the printer object labeled Default

The default printer in the desktop is not necessarily the same as the default printer defined to the AIX printing subsystem. Therefore, you can change the default printer in the desktop without changing the default printer known by the AIX printing subsystem. The desktop default printer may be changed by each user for their own environment or by the system administrator for the whole system.

To change the desktop default printer for an individual user, set and export the LPDEST=<destination_name> environment variable in the $HOME/.dtprofile file.

For example:

If you want the printer, we integrated into the desktop in 12.5.1, “Integrating a New Printer into the Desktop” on page 173, to be the new default printer known in your desktop enter the following line in your $HOME/.dtprofile:

```
export LPDEST=psc
```

To change the default printer for all users, you have to have root access. Export the LPDEST=<destination_name> environment variable in the 0010.dtpaths file located in the /etc/dt/config/Xsession.d directory. If the file does not exist copy the default file from /usr/dt/config/Xsession.d./0010.dtpaths into /etc/dt/config/Xsession.d/0010.dtpaths.

Note: Making a copy of the configuration files to the appropriate /etc/dt directory ensures that any changes you make to these files will be retained between desktop version upgrades.

For example:

If you want the printer, we integrated into the desktop in 12.5.1, “Integrating a New Printer into the Desktop” on page 173, to be the new default printer for all users enter the following line in the /etc/dt/config/Xsession.d./0010.dtpaths file:

```
export LPDEST=psc
```

These changes, at the user level, as well as at the system wide level , will be effective at the next login.

12.6 Search Path Environment Variables

The search path environment variables define the location where the desktop searches for actions, data type definitions, icon files, help files and Application Manager folders. Each of these items has a specific search path:

- **DTDATABASESEARCHPATH** for actions and data type definitions
- **DTAPPSEARCHPATH** for Application Manager folders
- **DTHELPSEARCHPATH** for help volume files
- **DTICONSEARCHPATH** for icon files
- **DTICONBMSEARCHPATH** for bitmap icon files
- **DTMANPATH** for manual pages

If an action or an icon or any other item is not located in one of the directories listed in the respective search path, it will not be found by the desktop.

During login the /usr/dt/bin/Xsession startup shell script invokes the command /usr/dt/bin/dtsearchpath to set and export the search path environment variables to their default value.

The desktop traces the DTDATABASESEARCHPATH only at login and when the **ReloadApps** action is launched. The action, data type and front panel definitions defined in the directories mentioned in the DTDATABASESEARCHPATH are loaded in a memory table called Dtdatabase. To find an action, data type or front panel definition the desktop does not search in the DTDATABASESEARCHPATH. It only searches in the memory table. For more information about the memory table Dtdatabase refer to 8.4, “Action and Data Type Database” on page 126

The syntax for dtsearchpath command is:

```
dtsearchpath [-u username] [-v] [-o]
```

Where:

-u Causes dtsearchpath to return the search paths for the specified user. This option is useful for system administrators who need to understand the search paths for a particular user.

-v The verbose option causes dtsearchpath to print to standard output the values of the search environment. By default, the command runs silently.

-o The don’t-optimize option causes dtsearchpath to add a path to the search path even if the path does not exist.

The following is a sample output from the dtsearchpath -v command:

```
DTMOUNTPOINT: /nfs/
DTAPPSEARCHPATH:
 $HOME/.dt/appmanager
 /etc/dt/appconfig/appmanager/%L
 /etc/dt/appconfig/appmanager/C
 /usr/dt/appconfig/appmanager/%L
```

```
 /usr/dt/appconfig/appmanager/C

DTDATABASESEARCHPATH:
 $HOME/.dt/types
 /etc/dt/appconfig/types/%L
 /etc/dt/appconfig/types/C
 /usr/dt/appconfig/types/%L
 /usr/dt/appconfig/types/C

DTICONSEARCHPATH:
 $HOME/.dt/icons/%B%M.pm
 $HOME/.dt/icons/%B%M.bm
 $HOME/.dt/icons/%B
 /etc/dt/appconfig/icons/%L/%B%M.pm
 /etc/dt/appconfig/icons/%L/%B%M.bm
 /etc/dt/appconfig/icons/%L/%B
 /etc/dt/appconfig/icons/C/%B%M.pm
 /etc/dt/appconfig/icons/C/%B%M.bm
 /etc/dt/appconfig/icons/C/%B
 /usr/dt/appconfig/icons/%L/%B%M.pm
 /usr/dt/appconfig/icons/%L/%B%M.bm
 /usr/dt/appconfig/icons/%L/%B
 /usr/dt/appconfig/icons/C/%B%M.pm
 /usr/dt/appconfig/icons/C/%B%M.bm
 /usr/dt/appconfig/icons/C/%B

DTICONBMSEARCHPATH:
 $HOME/.dt/icons/%B%M.bm
 $HOME/.dt/icons/%B%M.pm
 $HOME/.dt/icons/%B
 /etc/dt/appconfig/icons/%L/%B%M.bm
 /etc/dt/appconfig/icons/%L/%B%M.pm
 /etc/dt/appconfig/icons/%L/%B
 /etc/dt/appconfig/icons/C/%B%M.bm
 /etc/dt/appconfig/icons/C/%B%M.pm
 /etc/dt/appconfig/icons/C/%B
 /usr/dt/appconfig/icons/%L/%B%M.bm
 /usr/dt/appconfig/icons/%L/%B%M.pm
 /usr/dt/appconfig/icons/%L/%B
 /usr/dt/appconfig/icons/C/%B%M.bm
 /usr/dt/appconfig/icons/C/%B%M.pm
 /usr/dt/appconfig/icons/C/%B

DTHELPSEARCHPATH:
 $HOME/.dt/help/%H
 $HOME/.dt/help/%H.sdl
 $HOME/.dt/help/%H.hv
 /etc/dt/appconfig/help/%L/%H
 /etc/dt/appconfig/help/%L/%H.sdl
 /etc/dt/appconfig/help/%L/%H.hv
 /etc/dt/appconfig/help/C/%H
 /etc/dt/appconfig/help/C/%H.sdl
 /etc/dt/appconfig/help/C/%H.hv
 /usr/dt/appconfig/help/%L/%H
 /usr/dt/appconfig/help/%L/%H.sdl
 /usr/dt/appconfig/help/%L/%H.hv
 /usr/dt/appconfig/help/C/%H
 /usr/dt/appconfig/help/C/%H.sdl
 /usr/dt/appconfig/help/C/%H.hv
```

```
MANPATH:
 /usr/dt/man
 /usr/dt/man
 /usr/share/man
 /usr/lpp/info
```

Notice the following:

- The list of directories shown for each variable also represent the precedence that the desktop uses during the search, from top down. In fact the desktop terminates the search for a component when the first match occurs.
- The %L symbol will be translated to the value of the LANG environment variable. Notice that the locale directory takes precedence over the corresponding C directory.
- In the icon search paths the %B symbol is translated to the basename of the requested icon and the %M symbol to the size of the icon (for information about the sizing and naming conventions see 11.2, “Desktop Icon Name and Size Conventions” on page 163). In the Help search path the %H symbol is translated to the basename of the requested help volume.

For more information read the man pages for the command dtsearchpath.

12.7 Customizing the Default Search Paths

The search path environment variable values are set at login. The /usr/dt/bin/Xsession start up script sources the user′s $HOME/.dtprofile script and any scripts located under the directories /etc/dt/config/Xsession.d and /usr/dt/config/Xsession.d. After sourcing the Xsession.d files, the Xsession script will invoke /usr/dt/bin/dtsearchpath to set and export the environment variables DTDATABASESEARCHPATH, DTHELPSEARCHPATH, DTICONSEARCHPATH, and DTICONBMSEARCHPATH.

All the search path variables can be customized on a system wide or on a single user basis:

- System wide customization must be included in any file under the /etc/dt/config/Xsession.d directory. The filename is optional, you can create a new file with a new filename, or copy the file /usr/dt/config/Xsession.d/0010.dtpaths under /etc/dt/config/Xsession.d/0010.dtpaths and modify it.
- Single user customization must be included in the file $HOME/.dtprofile.

Each search path variable accepts input from two other environment variables:

DTAPPSEARCHPATH accepts input from:

- DTSPSYSAPPHOSTS for system wide customization
- DTSPUSERAPPHOSTS for single user customization

DTDATABASESEARCHPATH accepts input from:

- DTSPSYSDATABASEHOSTS for system wide customization
- DTSPUSERDATABASEHOSTS for single user customization

DTICONSEARCHPATH and DTICONBMSEARCHPATH accept input from:

- DTSPSYSICON for system wide customization
- DTSPUSERICON for single user customization

DTHELPSEARCHPATH accepts input from:

- DTSPSYSHELP for system wide customization
- DTSPUSERHELP for single user customization

Input environment variables fall into two categories, those that support remote host specification and those who don't. If the name of the variable contains the string HOST it will support remote host specification. This means that you cannot specify remote help or icon directories by defining them in one of the input variables, but if you specify a remote Application Manager folder, the associated remote help and icon directories will be known to your local system.

All the input environment variables accept a comma-separated list of local paths:

```
export VARIABLENAME=/path1,/path2,...
```

The input variables that support remote host specification also accept this syntax:

```
export VARIABLENAME=remotehost:/remotepath,/path1,/path2,...
```

The variables that contain the word USER are exported in the $HOME/.dtprofile, they will be prepended to $HOME/.dt location in the default search path value, and therefore they will get the highest priority. Those variables who contain the word SYS are exported in a file under the /etc/dt/config/Xsession.d directory. As mentioned above any script file in here will be sourced. Definitions in these scripts will take precedence over definitions in the factory default location (/usr/dt paths).

For example, to set the icon search path to include the /usr/local/games/icons subdirectory, on a system wide basis the following line would appear in a script in the /etc/dt/config/Xsession.d subdirectory:

```
export DTSPSYSICON=/usr/local/games/icons
```

To set the DTDATABASESEARCHPATH to include the /myapplpath path, on a single user basis, the following line would appear in a script in the $HOME/.dtprofile file:

```
export DTSPUSERDATABASEHOSTS=/myapplpath
```

To set the Application Manager path to include the applications on host goby, under the /vipapps path, plus the applications under the /opt local path on a system wide basis, the following line would appear in a script in the /etc/dt/config/Xsession.d subdirectory:

```
export DTSPSYSAPPHOSTS=goby:vipapps,/opt
```

If you now execute dtsearchpath -v, you will see the following value for the variable DTAPPSEARCHPATH:

```
DTAPPSEARCHPATH:
                    $HOME/.dt/appmanager
                    /etc/dt/appconfig/appmanager/%L
                    /etc/dt/appconfig/appmanager/C
                    /nfs/goby/vipapps/appconfig/appmanager/%L
                    /nfs/goby/vipapps/appconfig/appmanager/C
```

```
/opt/appconfig/appmanager/%L
/opt/appconfig/appmanager/C
/usr/dt/appconfig/appmanager/%L
/usr/dt/appconfig/appmanager/C
```

As you can see the $HOME and the local /etc paths have higher priority, the newly added paths take precedence in the order they are listed in the exported environment variable. The supplied default paths still have the lowest priority.

To change the precedence of local and remote directories you can use the special name `localhost` to specify the precedence of the directories /etc/dt/appconfig/appmanager in relation to the other directories. For example:

```
export DTSPSYSAPPHOSTS=goby:/vipapps,localhost:,/opt
```

In this case the result would be:

```
DTAPPSEARCHPATH:
                    $HOME/.dt/appmanager
                    /nfs/goby/vipapps/appconfig/appmanager/%L
                    /nfs/goby/vipapps/appconfig/appmanager/C
                    /etc/dt/appconfig/appmanager/%L
                    /etc/dt/appconfig/appmanager/C
                    /opt/appconfig/appmanager/%L
                    /opt/appconfig/appmanager/C
                    /usr/dt/appconfig/appmanager/%L
                    /usr/dt/appconfig/appmanager/C
```

For more information see the man pages for the `dtsearchpath` command.

12.8 Accessing Remote Data With the Desktop

The easiest way to access remote data is mounting a remote file system on your local machine using NFS. Once a file system is mounted the desktop will be able to work on it just as on any local file system.

Figure 61 shows this situation.

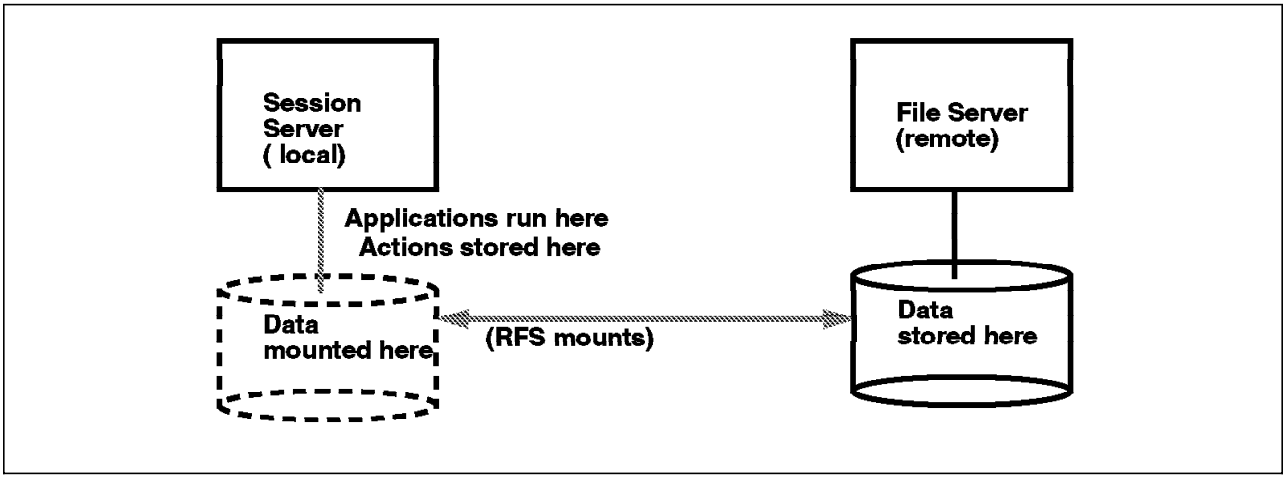


Figure 61. Accessing Remote Data

For example, suppose you want to work with your local FrameMaker application on a set of files that are physically placed on a remote system, you have to do this:

- Make sure that the filesystem where the data files reside has been exported on the remote system, or ask the system administrator to export it.
- Mount the remote file system on your local machine, for instance you could issue the following command:

  ```
  mount remotehost:/remotepath /localpath
  ```

  If your system provides the automount facility, you can use that as well.
- You can now navigate with the File Manager into the mounted directory. If you created actions and data types for the FrameMaker application, as we did in the example 8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132, you can now see the remote FrameMaker data files represented with the same icon that you normally use for your local FrameMaker files, and you can operate on the remote data files just as you do on your local files.

12.9 Running a Remote Application From the Desktop

A specific daemon is provided by the desktop to perform remote execution. It is called the sub-process control daemon (dtspcd) and it is responsible for sending and receiving remote execution requests and to evaluate whether the requester has authority for remote execution. Another daemon involved in remote execution is rpc.ttdtserverd. This handles file name mapping requests. These daemons must be configured and running both on the local system (requesting the remote execution) and on the remote system (receiving the request). When the desktop is installed these daemons are automatically configured and started at boot time.

There are two ways to start a remote execution using the desktop daemons:

- Configuring application server and clients. In this configuration the application server contains the application binaries, configuration files and all the the desktop components (actions, data types, icons, help volumes and application group definitions). The clients import the application group from the application server so that an icon representing the remote application appears in the local Application Manager. The user can start the remote application by double clicking on this object just as if the application is local.
- Using the EXEC_HOST field in the action definition. In this case the action is installed and executed on your local system (session manager) and all of the other the desktop components are on the session manager. The application is installed and executed on the remote system (execute host).

  You can use this method instead of the application server when the application is installed on a remote system but it’s not launch integrated in the desktop. You remember that this level of integration means there are actions, data types and icons provided for a specific application. Refer to 9.1, “Integrating an Application into the Desktop” on page 144 for more information.

12.9.1 Configuring Application Server and Clients

On the application server you have to:

- Make sure the system is configured in the network and the desktop is installed and running.
- Make sure the application is installed and integrated into the desktop. For more information see 9.1, “Integrating an Application into the Desktop” on page 144.
- Make sure your networked home directory is mounted on the application server, see 12.10, “Configuring a Networked Home Directory” on page 183.
- If the data resides elsewhere, mount the filesystem containing the data.

On the local system you have to:

- Make sure the system is configured in the network and the desktop is installed and running.
- Make sure your networked home directory is mounted, see 12.10, “Configuring a Networked Home Directory” on page 183. Consider that your home directory is not always physically located on your local system.
- Add the application server to the application search path:
  - If the /etc/dt/config/Xsession.d/0010.dtpaths file doesn’t exist copy the default from /usr/dt/config/Xsession.d/0010.dtpaths into /etc/dt/config/Xsession.d/0010.dtpaths using the following command.

    ```
    cp /usr/dt/config/Xsession.d/0010.dtpaths  /etc/dt/config/Xsession.d
    ```

  - Edit /etc/dt/config/Xsession.d/0010.dtpaths and add the following line:

    ```
    export DTSPSYSAPPHOST=hostname:/applpath
    ```

    Where `hostname` is the hostname of the application server and /applpath is the path where the application is installed.
- Mount the remote file system where the application is installed: For example:

  ```
  mount hostname:/applpath /applpath
  ```

- Log out and log back in. If you now issue the command `dtsearchpath -v`, you’ll notice that the /applpath directory has been added to all the default search paths. If you open the Application Manager you should see the application icon and double-clicking on it you should be able to start the remote application.

This situation is shown in Figure 62 on page 182.

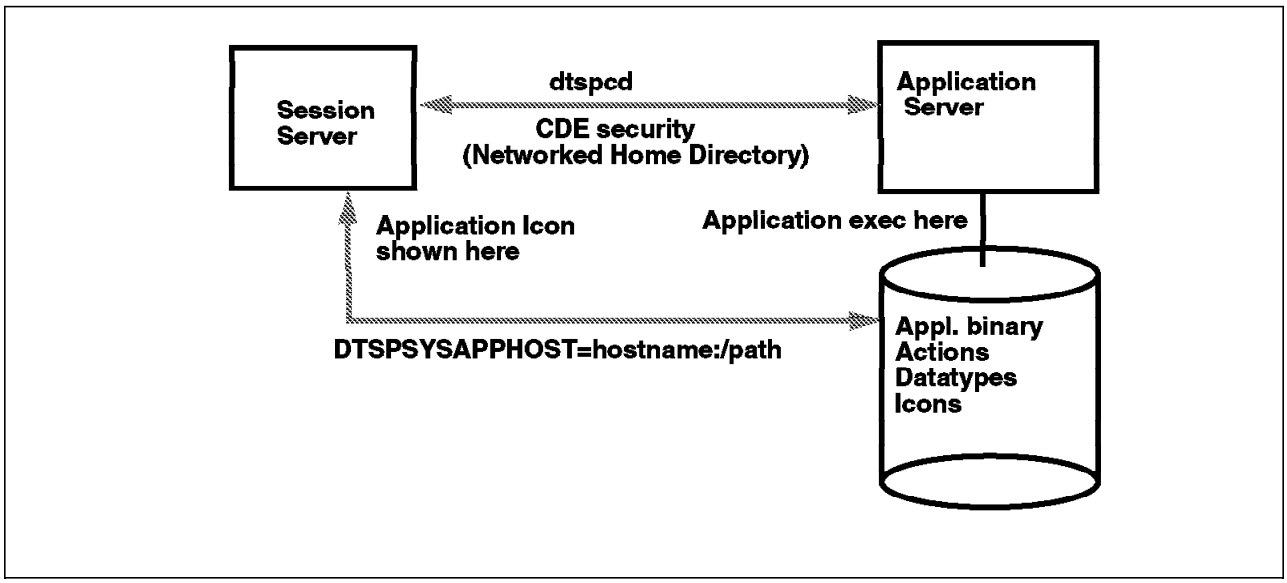


Figure 62. Application Server

12.9.2 Using the EXEC_HOST Field in the Action Definition

On the remote system, the execute host, you have to:

- Make sure that the system is configured in the network and the desktop is installed and running
- Make sure your networked home directory is exported, see 12.10, "Configuring a Networked Home Directory" on page 183
- If the data resides elsewhere mount the filesystem containing the data

On the local system, the session manager, you have to:

- Make sure the system is configured in the network and the desktop is installed and running.
- Make sure your networked home directory is mounted, see 12.10, "Configuring a Networked Home Directory" on page 183. Consider that your home directory is not always physically located on your local system.
- Create actions, data types and icons for the application. If you want you can use the CreateAction tool. For more information see 8.7, "Creating a Simple Data Type Using the CreateAction Tool" on page 132
- Edit the Dtdatabase definition file that you just created and add the EXEC_HOST statement to your action:

```
ACTION ActionName
{
   ...
   EXEC_HOST     hostname
   ...
}
```

  Where hostname is the hostname of the Exec_Host.

- Configure and export your networked home directory, see 12.10, "Configuring a Networked Home Directory" on page 183.

Figure 63 on page 183 illustrates the Exec_Host configuration.

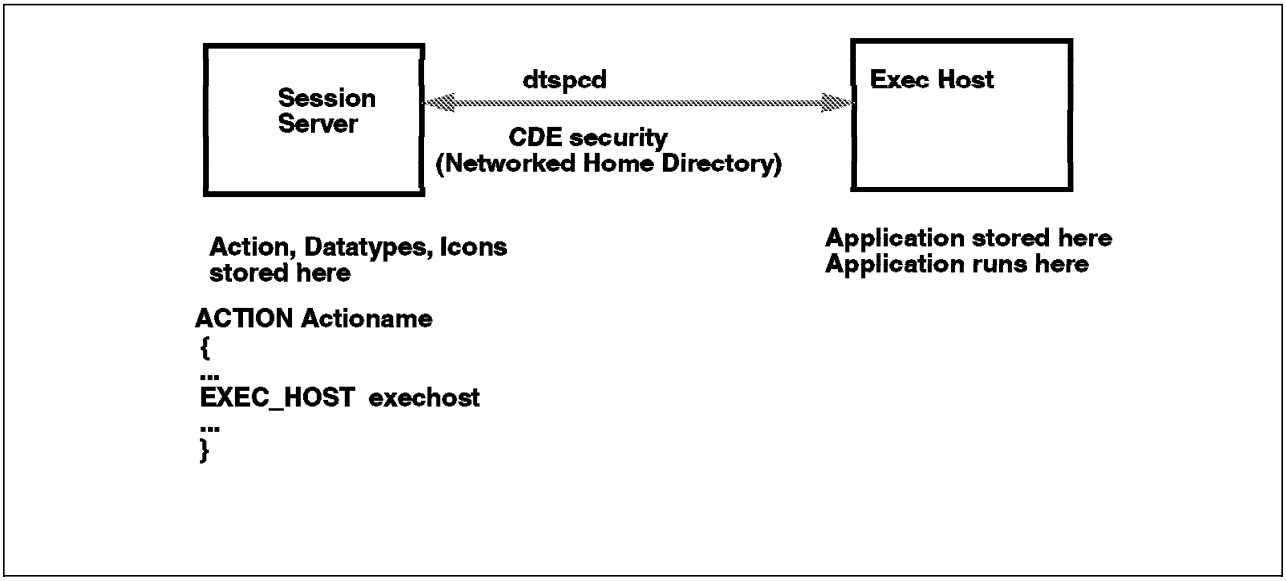


Figure 63. Using EXEC_HOST in the Action

12.9.3 Remote Execution Without Using the Desktop Daemons

There is also one other possibility to be considered. It is possible to use remote execution without using the the desktop daemons. This can be useful when the systems where you want the remote application executed doesn't have the desktop installed. There are two possibilities:

- Use the standard UNIX remote facilities, like rsh or rexec. In this case your Action must contain a statement like this:

  ```
  EXEC_STRING   rsh hostname command -display $DISPLAY
  ```

  The local system (session manager) must have the authority to execute on the remote host, and the remote host must have the authority to send output on the local display.

- Mount the remote filesystem where the application is installed on the local system and integrate it into the desktop. See 9.1, "Integrating an Application into the Desktop" on page 144 or 8.7, "Creating a Simple Data Type Using the CreateAction Tool" on page 132. In this case from the the desktop point of view it's just like if the application was installed locally. In fact the application also runs locally, unless you use the EXEC_HOST field in your action, as explained above.

12.10 Configuring a Networked Home Directory

The desktop works better in a networked environment if each user has a single home directory that is mounted over all the clients and servers on the network. In this way, in fact, the user can log into any system in the network and use their own customizations and files. Moreover using the networked home directory is a prerequisite for launching remote execution using the the desktop daemons. See 12.9, "Running a Remote Application From the Desktop" on page 180.

A networked home directory works more efficiently if each user account is created on all the systems on the network with the same user ID (uid) and group ID (gid).

If you want to perform remote execution using the sub-process control daemon (dtspcd), you have to create a networked home directory and have it mounted on both systems participating in the remote execution. In fact the desktop uses the networked home directory as the authorization method for remote execution.

To create a networked home directory you have to:

- Create the user account and a home directory on the file server that will provide the networked home directory. Export the home directory.
- On all other systems (clients) create a user account with the same name uid and gid, and same home directory. Configure each client to mount the networked home directory automatically after each reboot, from the file server. Notice that the static mount is not enough, the desktop looks for an entry in /etc/filesystems.

Figure 64 shows the networked home directory configuration.

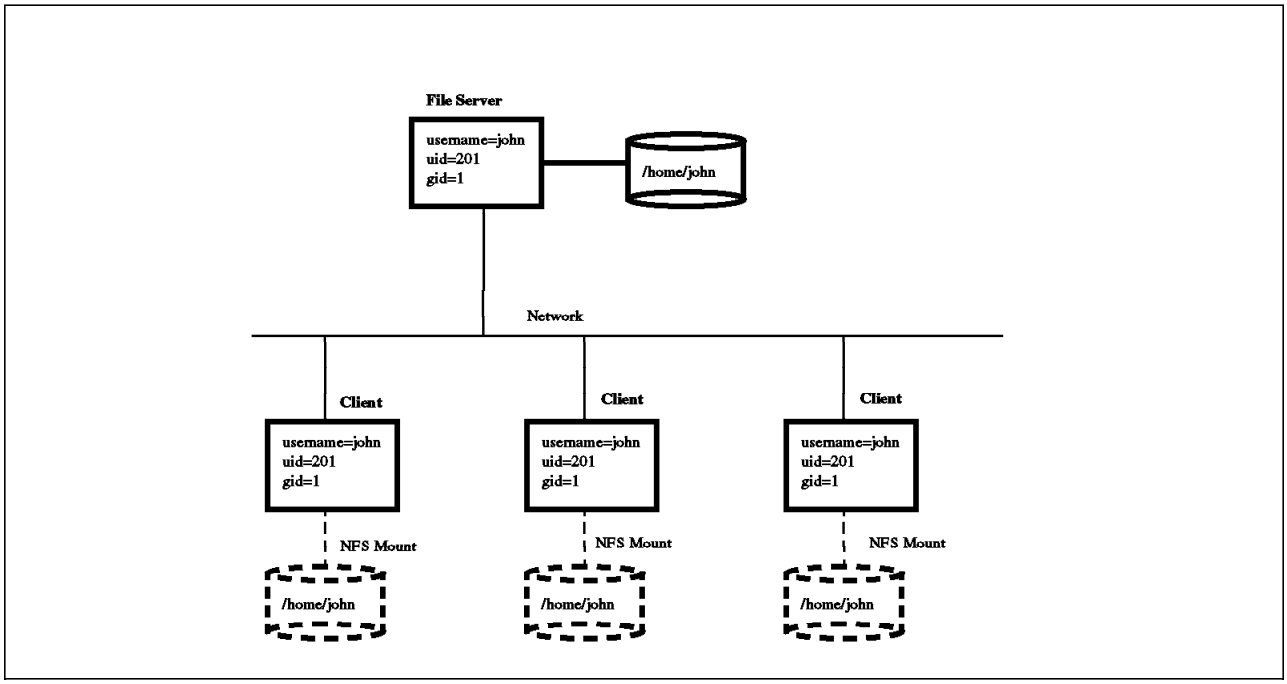


Figure 64. Network Home Directory Configuration

Chapter 13. Localization

Localization provides the ability to select the user language environment, on a system wide or single user basis.

Languages supported by the desktop are:

- Western languages
- Japanese
- Traditional Chinese
- Simplified Chinese
- Korean

The localization system in the desktop is based on the value of the LANG environment variable. 13.2, “Specifying Different Language Settings in the Desktop” on page 188 explains how to set this variable for the desktop.

The search path environment variables in the desktop are designed to take the LANG variable into account. For more information on search paths see 12.6, “Search Path Environment Variables” on page 175 and 12.7, “Customizing the Default Search Paths” on page 177. The `dtsearchpath -v` command shows the value of all the search path environment variables. For instance, the default value for DTDATABASESEARCHPATH is:

```
DTDATABASESEARCHPATH:
        $HOME/.dt/types
        /etc/dt/appconfig/types/%L
        /etc/dt/appconfig/types/C
        /usr/dt/appconfig/types/%L
        /usr/dt/appconfig/types/C
```

The %L symbol will be translated to the value of the LANG variable. The list shown above also represents the precedence that the desktop uses during the search, from top down. In fact the desktop terminates the search when the first match occurs.

The default locale is C. This will be always searched even when a different locale is defined. However, items found in the current locale directories will take precedence over those in the C directories. For instance, if actions with the same name exist in the locale directory and in the C directory, the action in the locale directory is executed.

When the desktop is installed, the installation procedure automatically populates some of the the desktop directories, with as many locale subdirectories as the number of locales installed in the system. For instance, if on your system the American_English, German and Italian locales are installed under /usr/dt/appconfig/types then under /etc/dt/appconfig/types you will find the following subdirectories:

- C
- en_US
- de_DE
- it_IT

For other components locale subdirectories must be created manually by the system administrator, when necessary.

The following is a brief description of the localized directory used by the desktop. Again, some of these don't exist by default or are empty and must be created and populated manually when needed. For more information on the the desktop directory tree see Appendix A, "The Desktop Directories and Files" on page 191

13.1.1 Actions, Data Types and Front Panel Definitions

The search path for these components includes the following language dependent directories:

- /etc/dt/appconfig/types/$LANG

  System wide configuration directory in the current locale

- /etc/dt/appconfig/types/C

  System wide configuration directory in the default locale

- /usr/dt/appconfig/types/$LANG

  Factory default (built-in) directory in the current locale

- /usr/dt/appconfig/types/C

  Factory default (built-in) directory in the default locale

13.1.2 Application Manager Configuration Files

The search path for these components includes the following language dependent directories:

- /etc/dt/appconfig/appmanager/$LANG

  System wide configuration directory in the current locale

- /etc/dt/appconfig/appmanager/C

  System wide configuration directory in the default locale

- /usr/dt/appconfig/appmanager/$LANG

  Factory default (built-in) directory in the current locale

- /usr/dt/appconfig/appmanager/C

  Factory default (built-in) directory in the default locale

13.1.3 Icon Files

The search path for these components includes the following language dependent directories:

- /etc/dt/appconfig/icons/$LANG

  System wide configuration directory in the current locale

- /etc/dt/appconfig/icons/C

  System wide configuration directory in the default locale

- /usr/dt/appconfig/icons/$LANG

  Factory default (built-in) directory in the current locale

- /usr/dt/appconfig/icons/C

Factory default (built-in) directory in the default locale

13.1.4 Help Files

The search path for these components includes the following language dependent directories:

- /etc/dt/appconfig/help/$LANG

  System wide configuration directory in the current locale

- /etc/dt/appconfig/help/C

  System wide configuration directory in the default locale

- /usr/dt/appconfig/help/$LANG

  Factory default (built-in) directory in the current locale

- /usr/dt/appconfig/help/C

  Factory default (built-in) directory in the default locale

13.1.5 Session, Login and Workspace Manager Configuration Files

The following configuration files are localized:

- /etc/dt/config/%L/Xresources

  Customized Login Manager resource file

- /usr/dt/config/%L/Xresources

  Default (built-in) Login Manager resource file

- /etc/dt/config/%L/sys.font

  Customized Session Manager font resource file

- /usr/dt/config/%L/sys.font

  Default (built-in) Session Manager font resource file

- /etc/dt/config/%L/sys.resources

  Customized Session Manager resource file

- /usr/dt/config/%L/sys.resources

  Default (built-in) Session Manager resource file

- /etc/dt/config/%L/sys.session

  Customized Session Manager executable file

- /usr/dt/config/%L/sys.session

  Default (built-in) Session Manager executable file

- /etc/dt/config/%L/sys.dtwmrc

  Customized Workspace Manager executable file

- /usr/dt/config/%L/sys.dtwmrc

  Default (built-in) Workspace Manager executable file

13.1.6 Colors and Backdrops Configuration Files

The following configuration files are localized:

For Colors:

- /etc/dt/palettes/$LANG
- /usr/dt/palettes/$LANG

For Backdrops:

- /etc/dt/backdrops/$LANG
- /usr/dt/backdrops/$LANG

13.1.7 Message Catalogs

The NLSPATH environment variable determines the directory path where the applications search for message catalogs. Both LANG and NLSPATH must be set to use message catalogs.

The search path for localized catalog files is:

- /usr/dt/lib/nls/msg/%L

13.2 Specifying Different Language Settings in the Desktop

The localization system in the desktop is based on the value of the LANG environment variable.

There are three ways to set the LANG environment variable for the desktop.

- Customizing the /etc/dt/config/Xconfig file. The LANG variable is set for all users and also the login screen is localized for all displays.
- From the option menu in the login screen. The login screen is localized and the LANG variable is set but only for that display and only for the upcoming session. After the user logs out LANG will be set back to its default value.
- Customizing the .dtprofile. The login screen is not localized but LANG is set for that user.

13.2.1 Configuring Xconfig File

The system administrator can use this file to customize the locale on a system wide basis, for a specific display or for all displays, by the following procedure.

1. If the /etc/dt/config/Xconfig file doesn't exist yet, copy it from /usr/dt/config/Xconfig

   Note: Making a copy of the configuration files in the according /etc/dt directories ensures that any changes that you make to these files will be retained between desktop version upgrades.

2. Edit /etc/dt/config/Xconfig and place the following line in it:

   ```
   dtlogin.displayname_0.language: language
   ```

   For example:

   - To set the Italian locale for the display myhost:

     ```
     dtlogin.myhost_0.language:    it_IT
     ```

   - To set the German locale for all the displays:

```
dtlogin*language:    de_DE
```

3. Log out and log in again.

13.2.2 Setting LANG from the Option Menu

The option menu in the Login Manager screen shows all the locales installed in the system. Choose one of these to change the default LANG variable value for the next session. The value of LANG will return to its original value at the conclusion of the session. As soon as you change the locale setting the Login Manager screen will reset and will show the greetings in the new locale language.

13.2.3 Setting LANG in .dtprofile

You can override the system wide LANG setting within the .dtprofile file. The following example overrides the system LANG variable setting to specify Italian as the language environment.

```
export LANG=it_IT
```

13.3 Configuring an Xstation to Use a Non-US Keyboard

To configure an Xstation to use a non-US keyboard you can follow this procedure:

1. If the /etc/dt/config/Xconfig file doesn't exist yet copy it from /usr/dt/config/Xconfig. Change its permissions to make it readable, writable and executable.
2. In the file /etc/dt/config/Xconfig add one or more entries like this:

   ```
   Dtlogin*xstationname_0.setup:    /etc/dt/Xsetup_Lang
   ```

   Where the string xstationname is the host name of the Xstation, and Xsetup_Lang is the name of the configuration file. You can change this name to any one you prefer. For instance if your Xstation is called mickey and has an Italian keyboard attached, you could enter the following line:

   ```
   Dtlogin*mickey_0.setup:     /etc/dt/Xsetup_It
   ```

3. The /etc/dt/config/Xsetup_Lang file must be executable and should contain the following lines:

   ```
   #!/bin/ksh

   #Source the environment
    . /etc/environment

   #Set the language for the keyboard
    KBD_LANG=language  #for example: it_IT

   #Invoke the xmodmap program
    XDIR=/usr/lpp/X11/defaults/xmodmap
    /usr/bin/X11/xmodmap $XDIR/$KBD_LANG/keyboard

   #if needed add a language specific font path
    /usr/lpp/x_st_mgr/bin/xset +fp /usr/lib/X11/fonts/$KBD_LANG
   ```

If you are working in an environment where Xstations with keyboard of different languages exist, you can create a file like the one shown above for each

language, and in the Xconfig file set each Xstation to use the corresponding language file. For example:

```
Dtlogin*xst1_0.setup:    /etc/dt/Xsetup_It
Dtlogin*xst2_0.setup:    /etc/dt/Xsetup_It
Dtlogin*xst3_0.setup:    /etc/dt/Xsetup_De
Dtlogin*xst4_0.setup:    /etc/dt/Xsetup_De
```

13.4 Configuring an Application to Use Language Specific Resources

The default location for application resource file in the desktop is:

```
/usr/dt/app-defaults/$LANG
```

The XUSERFILESEARCHPATH environment variable can be used to change the app-default directory. For example it could be set in your $HOME/.dtprofile to:

```
export XUSERFILESEARCHPATH=/users/app-defaults/%L
```

Appendix A. The Desktop Directories and Files

The following entries will list the directories used by the desktop to hold configurations or definitions. Not all directories are listed. Only those directories that may be of interest to the average user or system administrator are listed.

A.1 Home Directory

This section will list all the important user configuration files and directories. Depending on your implementation and personal configurations not all of the mentioned directories or files will exist.

A.1.1 $HOME/.dtprofile

$HOME/.dtprofile is created automatically for each user the first time the user logs on to the desktop. The File $HOME/.dtprofile is similar to $HOME/.profile.

A.1.2 $HOME/.dt

This directory contains all user specific desktop configurations, definitions and files.

A.1.2.1 $HOME/.dt/appmanager

This directory contains Application Manager folders which have been set up by the user to group their tools and applications.

A.1.2.2 $HOME/.dt/help

This directory contains help access history information. This information is used by the Help Manager to display your help history.

A.1.2.3 $HOME/.dt/icons

This directory contains the user created icons.

A.1.2.4 $HOME/.dt/palettes

This directory contains the user created color palettes.

A.1.2.5 $HOME/.dt/sessions

This directory contains four sub-directories. These four sub-directories contain the definitions for your the current session, your last session, your home session definition and your previous home session definitions.

A.1.2.6 $HOME/.dt/Trash

This directory contains all the files you deleted to trash.

A.1.2.7 $HOME/.dt/types

This directory contains all your user created actions, data types and front panel definitions.

A.1.2.8 $HOME/.dt/types/fp.dynamic

This directory contains front panel definition file created by interactively changing the front panel.

A.2 /etc/dt

This directory contains the configurations and definitions specific to your system and have been created and or changed by your system administrator.

A.2.1 /etc/dt/app-defaults

This directory contains customized X Window resource files for the desktop applications.

A.2.2 /etc/dt/appconfig/appmanager/$LANG

The $LANG sub-directories of this directory contain folders displayed by the Application Manager. These folders are merged with the user-defined folders and the system supplied default folders.

A.2.3 /etc/dt/appconfig/help/$LANG

This $LANG sub-directories contain the system wide Help Families, Help Volumes and Help Topics, for the particular language.

A.2.4 /etc/dt/appconfig/icons/$LANG

The $LANG sub-directories contain the system wide icons.

A.2.5 /etc/dt/appconfig/types/$LANG

The $LANG sub-directories contain the system wide action, data types and front panel definition files.

A.2.6 /etc/dt/config

This directory contains the desktop configuration and settings files.

A.3 /usr/dt

This directory contains the default configurations and definitions supplied with the desktop. The files in these directories should not be modified. Instead, any desired modifications should be made in copies of the desired file appropriately placed in the /etc/dt directory.

A.3.1 /usr/dt/app-defaults/$LANG

This directory contains the default X Window resource files for the desktop applications.

A.3.2 /usr/dt/appconfig/appmanager/$LANG

The $LANG sub-directories of this directory contain the default folders displayed by the Application Manager. These folders are merged with the user defined folders and the system administrator customized folders.

A.3.3 /usr/dt/appconfig/help/$LANG

This $LANG sub-directories contain the default Help Families, Help Volumes and Help Topics, for the particular language.

A.3.4 /usr/dt/appconfig/icons/$LANG

The $LANG sub-directories contain the default icons.

A.3.5 /usr/dt/appconfig/types/$LANG

The $LANG sub-directories contain the default action, data types and front panel definition files.

A.3.6 /usr/dt/backdrops

This directory contains the default backdrops.

A.3.7 /usr/dt/bin

This directory contains the desktop executables.

A.3.8 /usr/dt/config

This directory contains the desktop configuration and settings files.

A.3.9 /usr/dt/contrib

This directory contains contributed programs, definitions and examples.

A.3.10 /usr/dt/examples

This directory contains the IconBrowse.dt definition which enables the File Manager to display the contents of an icon file as an icon.

A.3.11 /usr/dt/man

This directory contains the desktop man pages.

A.3.12 /usr/dt/palettes

This directory contains the default color palettes.

A.3.13 /usr/dt/xdt2cde

This directory contains a conversion tool that can be used to convert customizations made to the previous AIXwindows desktop (XDT) into customizations for this desktop.

Appendix B. Question Cross Reference

This appendix cross references the information in this document into a list of questions about the desktop. The questions are grouped according to the various chapters in the document. The referenced sections referred to by each question is the primary section in which the question is answered.

B.1 Introduction

What´s COSE?

1.1, “The COSE Consortium” on page 1

What´s CDE?

1.2, “The Common Desktop Environment” on page 2

What are the CDE main components?

1.2.1, “CDE Main Components” on page 2

B.2 Login Manager

What´s the Login Manager?

Chapter 2, “Login Manager” on page 5

Can I change the appearance of the login window?

2.1, “Customizing the Appearance of the Login Window” on page 6

Can I have my own logo displayed?

2.2, “Changing the Logo on the Login Screen” on page 11

Can I change the message of the day screen?

2.3, “Customizing the Message of the Day” on page 12

Can I access the desktop without using the Login Manager?

2.4, “Accessing the Desktop Without Using the Login Manager” on page 18

Can I use desktop installed on another system on my local system?

2.5, “Using the Desktop Installed on Another System” on page 18

B.3 Session Manager

What´s the session Manager?

Chapter 3, “Session Manager” on page 21

What applications can be restarted by the Session Manager?

3.2, “Applications Which Can Be Restarted By the Session Manager” on page 24

What´s a home session?

Chapter 3, “Session Manager” on page 21

Can I have a display dependent session?

3.3, “Display Dependant Sessions” on page 27

How can I get back in if I mess up the Session Manager configuration files?

3.4, “Single Window Sessions” on page 28

Is my .Xdefaults file used by the desktop?

3.5, “The Session Manager′s Role With Respect to X Resources” on page 29

B.4 Front Panel

What′s the front panel?

Chapter 4, “Front Panel” on page 31

How can I change the name of a workspace?

4.2.4.3, “Interactively Renaming a Workspace” on page 54

How can I add a workspace?

4.2.4.1, “Interactively Adding Workspaces for Existing Users” on page 53

How can I remove a workspace?

4.2.4.2, “Interactively Removing Workspaces for Existing Users” on page 54

How can I add a subpanel?

4.2.2.1, “Adding Subpanels Using Interactive Customization” on page 50

How can I remove a subpanel?

4.2.2.2, “Removing Subpanels Using Interactive Customization” on page 51

How can I remove a control in a subpanel?

4.2.3.2, “Removing Controls From Subpanels Using Interactive Customization” on page 52

How can I add my own control to the default front panel.

4.2.1.1, “Adding Controls Using Interactive Customization” on page 46

How can I remove a control from the front panel?

4.2.1.2, “Removing Controls Using Interactive Customization” on page 47

How can I create my own front panel?

4.3, “Creating a New Front Panel” on page 58

Can I call my own tools (Editor, Terminal ...) from the front panel?

4.2.1.2, “Removing Controls Using Interactive Customization” on page 47

How can I lock the default front panel?

4.2.5, “Preventing Changes to a Control and or a Subpanel” on page 56

How can I add a client control?

4.3.2, “Creating Controls” on page 60

How can I create a two row front panel?

4.3.1, “Creating Boxes” on page 60

B.5 Style Manager

What′s the Style Manager?
Chapter 5, “Style Manager” on page 73

How can I change colors in my workspace
5.2, “Color Customization” on page 74

Can I create a new color palette?
5.2, “Color Customization” on page 74

Can I change the font size used by the desktop?
5.3, “Font Size Customization” on page 78

Can I change the font family used by the desktop?
5.3, “Font Size Customization” on page 78

Can I change the backdrop of my Workspace?
5.4, “Backdrop Customization” on page 79

Can I create my own backdrop?
5.4, “Backdrop Customization” on page 79

How can I see the root window in my workspace?
5.4, “Backdrop Customization” on page 79

Can I change the screen lock saver image?
5.8, “Screen Customization” on page 83

Can I set a timed screen lock?
5.8, “Screen Customization” on page 83

How can I get the icon box?
5.9, “Window Customization” on page 84

How can I set a home session?
5.10, “Login and Logout Customization” on page 85

Can I hide part of the Style Manager from my users?
5.1, “Customizing the Style Manager” on page 73

Can I have a vertical Style Manager interface?
5.1, “Customizing the Style Manager” on page 73

B.6 Workspace Manager

What′s the Workspace Manager?
Chapter 6, “Workspace Manager” on page 87

Why are my old X11/Motif configuration files not read anymore?
6.1, “Customizing Through Configuration Files” on page 87

How can I convert X11/Motif configuration files for use by the desktop?
6.1.1, “Converting mwm Configuration Files For Use By dtwm” on page 88

How can I move a window from one workspace to another?
6.2, “Moving, Copying and Removing Windows From and Between Workspaces” on page 103

How can I direct an X client to a specific workspace?
6.3, “Directing an X Window Client to a Specific Workspace” on page 107

How can I switch workspace using the keyboard?
6.4, “Switching Workspaces Using the Keyboard” on page 107

Can I use multiple head display with the desktop?
6.5, “Using a Multiple Screen Display With the Desktop” on page 108

How does the Window Manager get started?
6.6, “Starting the Workspace Manager” on page 108

How can I affect the appearance and behavior of my desktop environment?
6.1, “Customizing Through Configuration Files” on page 87

B.7 File Manager

What′s the File Manager?
Chapter 7, “File Manager” on page 109

How can I set a directory tree representation?
7.2, “Interactively Customizing the File Manager” on page 112

Can I see the files by properties?
7.3.1, “Viewing and Changing the File Properties” on page 114

How can I see hidden files?
7.3.2, “Viewing Hidden Files” on page 115

How can I see only the files I need?
7.3.3, “Viewing Only the Needed Files” on page 115

How can I move/copy/link files?
7.3.4, “Moving/Copying/Linking Files” on page 115

How can I rename a file?
7.3.5, “Renaming a File” on page 117

How can I change properties of a file?
7.3.1, “Viewing and Changing the File Properties” on page 114

How can I delete a file?
7.3.6, “Deleting a File” on page 117

How can I find the file I need?
7.3.8, “Searching for a File by Filename or File Contents” on page 117

How can I find which files contain a particular string?
7.3.8, “Searching for a File by Filename or File Contents” on page 117

How can I open the File Manager on a specific directory?
7.3.9, “Opening the File Manager on a Specific Directory” on page 118

How can I move File Manager icons to the root window?
7.3.10, “Moving File Manager Objects to the Workspace” on page 119

How can I copy a File Manager icon to the front panel?
7.3.11, “Copying a File Manager Object to the Front Panel” on page 119

How can I open the File Manager as another user?
8.9, “Invoking an Action from the Command Line” on page 140

B.8 Application Manager

What′s the Application Manager?
Chapter 9, “Application Manager” on page 143

How can I integrate my application into the desktop?
9.1, “Integrating an Application into the Desktop” on page 144

How can I add system wide actions to the Application Manager?
9.2, “Adding System Wide Applications to the Application Manager” on page 145

How can I add user specific actions to the Application Manager?
9.3, “Adding User Specific Applications to the Application Manager” on page 148

Does the EditResources action work?
29

B.9 Help Manager

What′s the Help Manager?
Chapter 10, “Help Manager” on page 151

How can I add my own help volume?
10.2, “Adding Your Own Help Volume or Changing an Existing Volume” on page 159

Can I change an existing help volume?
10.2, “Adding Your Own Help Volume or Changing an Existing Volume” on page 159

How can I search the Help Manager?
10.1.5, “Searching for an Entry or a Topic” on page 157

Can I print information in the Help Manager?
10.1.6, “Printing a Help Volume or Topic” on page 159

How does the Help Manager relate to Info Explorer?
10.3, “Relationship Between the Help Manager and InfoExplorer” on page 159

B.10 Actions and Data Typing

What′s an Action?
8.2, “Actions” on page 122

What′s data typing?
8.3, “Data Types” on page 124

What′s the action and data types database?
8.4, “Action and Data Type Database” on page 126

How can I create a simple action using CreateAction?
8.5, “Creating a Simple Action Using the CreateAction Tool” on page 128

How can I pass arguments to my actions?
8.6, “Passing Arguments to Actions” on page 130

How can I have all my data files share the same icon?
8.3, “Data Types” on page 124

How can I double click on my data file and open my application on it?
8.3, “Data Types” on page 124

How can I drag my data file on the application icon to open it?
8.6, “Passing Arguments to Actions” on page 130

How can I create a simple data type using CreateAction?
8.7, “Creating a Simple Data Type Using the CreateAction Tool” on page 132

Can I invoke an action from the command line?
8.9, “Invoking an Action from the Command Line” on page 140

B.11 Icons

How do I create my own icon?
11.1, “Creating Your Own Icons” on page 161

Can I use default icons?
11.5, “Using Built-In Icons?” on page 166

Can I use icons from other products (for example: XDT3, PM ...)?
11.6, “Using Icons from Other Products” on page 166

What is an icon?
Chapter 11, “Icons” on page 161

Can I browse icon files?
11.7, “Using File Manager to Browse Icons” on page 167

How can I reseize my icon, to get the four required icon sizes?
11.3, “Re-Sizing Icons” on page 164

How can I use the icon I created for my application?
11.4, “Using Created Icons” on page 165

What is the Icon Database?
11.8, “The DTICONSEARCHPATH Environment Variable” on page 168

B.12 Administering the Desktop

How can I install/de-install the desktop?
12.1, “Installing or De-installing the Desktop” on page 169

In which directories is the desktop installed?
12.2, “Directories Containing the Desktop Software” on page 169

How can I configure my Xstation to use the desktop?
12.4, “Configuring an Xstation to Use the Desktop” on page 171

How can I print from from the desktop?
12.5, “Printing From the Desktop” on page 172

How do I integrate a new printer into the desktop?
12.5.1, “Integrating a New Printer into the Desktop” on page 173

How can I change the default printer?
12.5.2, “Changing the Desktop Default Printer” on page 174

How can I use remote data from my application?

12.8, “Accessing Remote Data With the Desktop” on page 179

How I start a remote application from my desktop?

12.9, “Running a Remote Application From the Desktop” on page 180

How can I use a remote action?

12.9, “Running a Remote Application From the Desktop” on page 180

How can I use my desktop environment on another system?

12.3, “Migrating Desktop Customizations to Another Desktop Host” on page 171

What are the search paths environment variables for?

12.6, “Search Path Environment Variables” on page 175

How can I customize the default search paths?

12.6, “Search Path Environment Variables” on page 175

Why is my .profile file not read anymore?

12.2.3, “Contents of the User′s Home Directory” on page 170

B.13 Localization

How can I specify a different language setting?

13.2, “Specifying Different Language Settings in the Desktop” on page 188

How can I configure a non-US keyboard?

13.3, “Configuring an Xstation to Use a Non-US Keyboard” on page 189

How can I configure an application for using language specific resources?

13.4, “Configuring an Application to Use Language Specific Resources” on page 190

Glossary

accelerator. A key or sequence of keys on the keyboard or multiple clicks of mouse buttons that quickly performs specific menu or application functions without using a menu.

action. A desktop construct that provides a method for running applications, executing commands, and other activities such as printing, removing files, and changing directories.

action database. A database containing action and data type definitions.

action definition file. A file that defines an action by associating an icon with a command or executable file.

dction definition. A set of configuration statements that define the type, icon and command that is associated to a particular action.

API. Application program interface. A series of defined interface standards for an application. An API typically defines how input should be requested and obtained, and how output should be done.

app-defaults file. A file for each application that programmers use to define the X resources.

applications root directory. The toplevel directory of an application.

Application Manager. The software application that manages the tools and other software applications available to you.

application server. A host computer that provides access to a software application.

B

backdrop. The pattern that covers the workspace background.

background (color). The color in a color set that is used for the background of elements.

bitmap. An image stored in a raster format. Usually refers to an image limited to two colors (a foreground and a background color). Contrast with pixmap.

bottom shadow. The color in a color set that is used for the bottom and right-side beveled edges of raised elements such as buttons and recessed elements.

built-in icons. Icons that come with the desktop.

busy light. A control that blinks when a Front Panel action has been invoked prior to the appearance of a window or when reloading actions. The mouse pointer displayed when an application is busy and cannot accept input.

button. A generic term for a window control. See push button.

button binding. Association of a mouse button operation with a particular behavior.

button 1. On a mouse or similar pointing device, the leftmost button when configured for right-handed use; the rightmost button when configured for left-handed use. Button 1 is bound to the Select button functions. In the desktop, it also performs the functions of the Manipulation button in Transfer mode. See Select button.

button 2. On a three-button mouse or similar pointing device, the middle button. On a two-button mouse, the right button if configured for right-handed use or the left button if configured for left-handed use. Button 2 is bound to the Manipulation button functions In the desktop, the Manipulation button has two modes, Transfer and Adjust. See Manipulation button.

button 3. On a mouse or similar pointing device, the rightmost button when configured for right-handed use; the leftmost button when configured for left-handed use. Button 3 is bound to the Menu button functions. See Menu button.

by tree. A view of a file system or portion of a file system in which the folders and files contained within a parent folder are shown as linked with lines as in a ″family tree″ diagram.

C

CDE. Common Desktop Environment, a graphical user interface running on UNIX.

client. A system or software application that requests services from another application, usually across a network.

colormap. A numerically indexed set of color options in a graphics system. Although most display hardware uses only one color map, multiple color maps may exist in software. Applications may assign one of these maps to be used by the hardware.

command line login. A login choice that enables the user to access a traditional UNIX command-line login.

control. Describes an object in the Front Panel.

CreateAction. The software application that enables users to associate an icon with a command so that

the command can be issued by clicking on the icon. CreateAction is also used to define specific data types for an application's data files and to associate icons with those data types.

current session. The session saved by Session Manager when you log off. At the next login, unless you specify otherwise, this session automatically opens, enabling work to continue where you left off. Contrast with home session.

D

data type. A mechanism that associates particular data files with the appropriate applications and actions. Data types can determine the type of a file based on file naming conventions, such as a particular extension name, or on the contents of the file.

database host. A host computer where an action is defined.

default action. A choice that is activated when you press the Return key, double-click the Manipulation button on an object, or directly manipulate an object. The action that the designers expect you are most likely to want in the given situation.

default printer. The default printer is the AIX printer queue that is used when the print action in invoked either through an object's menu or by dropping the object onto the default printer icon.

dialog box. A window displayed by an application that requires user input.

drag. A user interaction in which a user drags source elements to a target element on which they are dropped.

drop. After starting a drag of an object, the act of releasing the mouse button. If the object is dropped in an appropriate area, an action is initiated.

dtappintegrate. A software application that is used to integrate an application into the Application Manager.

dtdatabase. An in memory table that defines the known actions and data types.

dtdatabase Definition File. Files that contain definitions for actions and data types that are used to load the Dtdatabase.

dtspcd. The sub-process control daemon.

F

File Manager. The software application used to manage the files and folders on your system.

file server. A host computer that stores data files used by applications.

focus. The place to which keyboard input is directed.

folder. An icon that represents a directory or a group in the Application Manager..

foreground (color). The color in a color set that is used for text. The Foreground color is always either black or white, depending upon the brightness of the corresponding background color.

front panel. A centrally located window containing controls for accessing applications and utilities, including the workspace switch. The Front Panel occupies all workspaces.

front panel control. An object in the Front Panel used as an interface to basic system services and frequently performed tasks and operations.

full Integration. An application that is integrated at all levels into the desktop. See Level 1, Level 2 and Level 3.

G

group. A folder that holds a specific software application or set of software applications.

H

help family. A volume or set of help volumes that provide online information about related applications. A help family must have a family file to enable the information to display in the browser volume.

help index. An alphabetic listing of help topics.

Help Manager. The software application that displays online help.

help volume. A complete body of help information about a subject.

home session. A choice at logout to designate a particular session, other than the one you are currently in, as the one you will automatically return to at the next login.

home topic. The topic at the top of the hierarchy in a help volume. This is the topic that is displayed when the user indicates a desire to browse a help volume.

hyperlink. In help text, information that when chosen displays another help topic. Hyperlinks can also be used to invoke other kinds of behavior, such as executing a system command or invoking specific application behavior.

I

icon. A graphical representation of an object; an icon can be directly manipulated. It consists of an image, an image background, and a label.

icon box. A window for organizing icons

icon browser. A special configuration state of File Manager that enables pixmap and bitmap files to be displayed with the contents of the files as their icons instead of the default pixmap and bitmap icons.

iconic path. Used to show the current directory below the menu bar using icons.

iconify. To turn a window into an icon. The push button that minimizes a window is located near the upper right corner of the window frame.

inactive workspace. Workspace that is not currently being displayed.

input focus. The state of a control that is to receive input from a particular input device.

install icon. Use to install icons into the Front Panel using drag and drop.

L

LANG. Environment variable that defines the current language environment.

launch Integration. The integration that is necessary to start an application from the desktop by either selecting its icon or one of its data files. See Level 1.

level 0. An application uses the default desktop integration.

level 1. An application is launch integrated into the desktop.

level 2. An application is launch integrated and style compliant.

level 3. An application is launch integrated, style compliant and is programmed to use facilities of the desktop.

local host. The CPU or computer on which a software application is running; your workstation.

locale. A language-specific or country-specific environment.

Login Manager. A software application that manages user logins into the desktop.

M

man pages. Help information in the AIX help format. Usually accessed by issuing the `man` command.

manipulation button. The button on a pointing device that you press to manipulate an object directly. The Manipulation button has two modes, Adjust and Transfer.

menu bar. A menu displayed below the title bar that contains only cascading choices.

menu-bar item. A cascading choice that appears on a menu bar. Menu-bar items provide access to menus, which contain additional choices.

menu button. The button on a pointing device that the user presses to view a pop-up menu.

mnemonic. A character that the user can type (possibly augmented with Alt) to move the focus elsewhere in a window or menu and/or to activate or toggle a choice whose label contains and emphasizes that character.

mount. To extend the directory hierarchy by attaching a file system from somewhere else in the hierarchy on a mount point directory.

N

networked home directory. The user′s home directory that has been mounted onto other systems so that the user has the same home directory regardless of which system the user logs in on.

O

object. An entity in the filesystem which is represented by an icon, has a predefined set of actions and an associated title.

P

palette. A range of graphically displayed choices, such as colors or collections of tools, that you can select in an application.

pixel. A picture element that is one spot in a rectilinear grid of thousands of such spots that are individually ″painted″ to form an image produced on the screen by a computer or on paper by a printer.

pixmap. An image stored in a raster format. Usually refers to an image that may have more than two colors. Contrast with bitmap.

platform. A term commonly used to represent a computer hardware and software configuration usually provided by a single vendor.

pop up menu. A menu that, when requested, is displayed next to the object it is associated with.

pull down menu. A cascaded menu that is displayed from a menu cascade button or a menu bar.

push button. A control that immediately starts an action by an application, such as executing a command, displaying a window, or displaying a menu.

R

ReloadApps. The Reload Applications action reloads the database of action, data type, and front panel definitions.

resource. A mechanism of the X Window System for specifying an attribute (appearance or behavior) of a window or application. Resources are usually named after the elements they control.

restart Workspace Manager. Use to re-initialize the Workspace Manager after making changes to resources or configuration files.

rpc.ttdbserverd. The database server process.

S

screen saver. A function that, after a specified time period, switches off the workstation display or varies the images that are displayed, thereby preventing raster burn of the screen.

select (color). The color in a color set that is used to indicate the selection of certain controls.

select button. The button on a pointing device that the user presses to make a selection.

session. A particular configuration of workspaces that includes Style Manager settings, open applications, and the size and position of objects.

Session Manager. A desktop software application that controls saving sessions, restoring sessions, screen locking and unlocking, and the use of screen savers. When a session is saved, the state of the desktop environment (location of icons, size and location of open windows, open/closed status of applications, current color palette, and so on) are preserved so that it can be restored at the next login.

session server. A system that provides networked sessions. Session files reside on the session server and are used whenever you log in to a system on the network.

style compliance. An application that conforms to the guidelines of the desktop style guide. See Level 2.

Style Manager. The software application used to customize some of the visual elements and system device behaviors of the workspace environment, including colors and fonts, and keyboard, mouse, window, and session start-up behaviors.

subpanel. An extension of the Front Panel that slides up providing access to additional components. Subpanels usually contain groups of related components.

T

topic. Information about a specific subject. Usually, this is approximately one screen full of information. Online help topics are linked to one another through hyperlinks.

topic hierarchy. A help volume's branching structure in which the home topic branches out (via hyperlinks) to progressively more detailed topics. See also home topic.

topic tree. A scrollable list of topics that can be selected to display help information.

top shadow. The color in a color set that is used for the top and left-side beveled edges of raised elements such as buttons or the bottom and right-side beveled edges of recessed elements.

transfer button. The button (or virtual button) on a pointing device that is used for data transfer operations.

trash can. A container for deleted files.

tree view. A view of a folder or files that includes all lower-level folders in the search path.

W

workspace. The current screen display, the icons and windows it contains, and the unoccupied screen area where icons can be placed.

Workspace Manager. The software application that controls the size, placement, and operation of windows within multiple workspaces.

workspace switch. A control that enables you to select one workspace from among several workspaces.

X

X bitmap. Two-color images limited to a foreground and a background color. Also known as XBM format.

X pixmap. Multicolored images that include static and dynamic colors. Also known as XPM format.

XBM. See X bitmap.

XPM. See X pixmap.

List of Abbreviations

API	Application Programming Interface
CDE	Common Desktop Environment
HP	Hewlett-Packard Company
IBM	International Business Machines Corporation
ISV	Independent Solution Vendor
ITSO	International Technical Support Organization
OSF	Open Software Fondation
MIT	Massachusetts Institute of Technology
motd	message of the day
MWM	Motif Window Manager
SCO	The Santa Cruz Operation
USL	Unix System Laboratory Incorporated
xbm	X Bitmap
XDMCP	X Display Manager Control Protocol
XDM	X Display Manager
XDT3	X Desktop Version 3
xpm	X Pixmap

Index

Special Characters

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

W

X

ITSO Technical Bulletin Evaluation RED000

AIX Version 4 Desktop Handbook

Publication No. GG24-4451-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

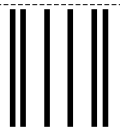
_____ _____
Name Address

_____ _____
Company or Organization

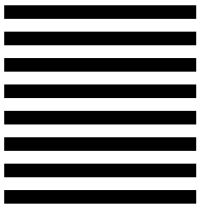
_____ _____
Phone No.

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 948, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.