TriTeal Enterprise Desktop (TED™) 4.0

Style Guide and Certification Checklist

TriTeal™

Printed in USA

Part No. TED010004000

Printing History
August 1995 ... First Edition
Printed in the U.S.A

Sales Information and Technical Support

TriTeal Corporation
2011 Palomar Airport Road, Suite 200
Carlsbad, CA 92009
USA

Tel: (619) 930-2077
Fax: (619) 930-2081
Support E-Mail: support@triteal.com
Sales E-Mail: info@triteal.com
World Wide Web: http://www.triteal.com/

Please
Recycle

Contents

Preface

Note – TriTeal Enterprise Desktop is an implementation of CDE. If you wish to build an application for the TriTeal Enterprise Desktop, it should be consistent with the CDE guidelines defined by the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

The *TriTeal Enterprise Desktop: Style Guide and Certification Checklist* provides application design style guidelines and the list of requirements for Common Desktop Environment application-level certification. Common Desktop Environment requirements consist of the OSF/Motif Version 1.2 requirements with Common Desktop Environment-specific additions.

The checklist describes keys using a model keyboard mechanism. Wherever keyboard input is specified, the keys are indicated by the engravings on the OSF/Motif model keyboard. Mouse buttons are described using a virtual button mechanism to better describe behavior independent from the number of buttons on the mouse.

Who Should Use This Book

This book provides information to assist the application designer in developing consistent applications and behaviors within the applications that are consistent with the standards for the Common Desktop Environment outlined in the *Style Guide and Certification Checklist*.

Before You Read This Book

As you compare the behavior of your application to the requirements in this checklist, you may want to consult the *OSF/Motif Style Guide (Revision 1.2)* for additional style considerations.

By default, this checklist assumes that your application is being designed for a left-to-right language environment in an English-language locale. Some sections of the checklist may require appropriate changes for other locales.

The Style Guide Part of the book refers to checklist items in Chapter 10. Each checklist item is labeled with numbers or letters. The numbered items correspond to the checklist items from the *OSF/Motif Style Guide, Revision 1.2*. The Common Desktop Environment-specific additions are labeled with alphabetic identifiers. The checklist items references are followed by the page number where the checklist item appears.

How This Book Is Organized

This book consists of two parts and appendices. Explanations of the sections of the book follow:

Part 1, "Style Guide," describes style considerations you should follow when designing applications for the Common Desktop Environment.

Chapter 1, "Introduction to Common Desktop Environment," is an introduction to the checklist and how to use it.

Chapter 2, "Input, Navigation, Selection and Activation," provides information on the keyboard focus model and the input device model, mouse-based and keyboard-based navigation, menu traversal, scrollable component navigation, selection and transfer models, selection actions, and basic activation.

Chapter 3, "Drag and Drop," provides information on incorporating drag and drop into your application.

Chapter 4, "Visual Design," provides information on designing icons and other visuals consistent with the Common Desktop Environment style.

Chapter 5, "Window and Session Control," provides information on window support, window decorations, window navigation, icons, application window management, and session management support.

Chapter 6, "Application Design Principles," provides information on layout, interaction, support for alternative visuals, messages, and work-in-progress feedback.

Chapter 7, "Common Dialogs", provides information on creating dialog boxes.

Chapter 8, "Application Messages," provides information on ways to provide feedback to the user.

Chapter 9, "Designing for Accessibility," provides information on making software applications accessible to people with disabilities.

Part 2, "Certification Checklist," is the certification checklist, which consists of a checklist divided into several topics:

Chapter 10, "Certification Checklist," provides the list of requirements for Common Desktop Environment application-level certification.

Appendix A, "Keyboard Functions," provides information on keyboard functions and keyboard engravings.

Appendix B, "Mouse Functions," provides information on mouse functions.

What Typographic Changes and Symbols Mean

The following table describes the type changes and symbols used in this book

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. system% You have mail.
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Code samples are included in boxes and may display the following:

Table P-1 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Example
`%`	UNIX C shell prompt	`system%`
`$`	UNIX Bourne and Korn shell prompt	`system$`
`#`	Superuser prompt, all shells	`system#`

Part 1 — Style Guide

Introduction to the Common Desktop Environment 1

Note – The Triteal Enterprise Desktop (TED) 4.0 is an implementation of the Common Desktop Environment (CDE) 1.0. If you are developing an application for TED, it should conform to the standards set forth in the *Style Guide and Certification Checklist* for the Common Desktop Environment.

Note – Some of the images in this book may differ slightly from the the actual TriTeal Enterpise Desktop 4.0 interface. These changes reflect TriTeal enhancments to the Common Desktop Environment.

The Common Desktop Environment is a graphical user interface for UNIX™ in its various flavors (AIX™, HP/UX™, Solaris™, UnixWare™, etc.). UNIX is a powerful and portable operating system. The desktop now brings unparalleled ease of use to UNIX. It is being adopted as a standard operating environment by many companies in the UNIX workstation market.

Advantages of a Common User Interface

The desktop interface brings greater ease-of-use and a consistent interface to UNIX. This provides many advantages to both end users and application developers. Among these advantages are:

- An easier to use interface enables users to learn the system quickly and use it efficiently.

- Consistency between UNIX platforms enables users to move from one computer to another with minimal difficulty. It also enables programmers to write a single application that can be compiled for each platform, significantly reducing development effort.
- The desktop provides as much consistency as possible with the Microsoft Windows™ and IBM OS/2™ environments. This enables users to easily move between these environments and the desktop.
- Unlike many competing operating systems, several built-in productivity applications enable the desktop user to be productive before buying application software.
- The desktop specifications have been submitted to the X/Open standards organization, ensuring that desktop is "open" and will not tie the user to proprietary solutions.

Relationship to X/Open Motif Style Guide

The desktop user interface follows the X/Open Motif guidelines. Motif, however, does not define a desktop, only the basic behaviors for applications and widgets. The *Style Guide and Certification Checklist* defines the guidelines that allow an application to integrate well with the desktop. Thus, to write a desktop-conforming application, you should follow both the *OSF/Motif Style Guide*, *Revision 1.2* and the *TriTeal Enterprise Desktop: Style Guide and Certification Checklist*.

Compliance with the desktop interface guidelines is voluntary and self-regulated. There is no formal certification process. Applications that meet all the required guidelines in this style guide and the *OSF/Motif Style Guide* can be considered desktop compliant.

Priorities have been assigned to each guideline: Required, recommended, and optional.

Input, Navigation, Selection, and Activation

2

Input devices have different actions depending on which part of the interface the user is interacting with. Usually, mouse users can access windows and controls more easily than keyboard users, due to the inherent flexibility in mouse manipulation. Keyboard users must use specific keys to move the cursor in the application.

Users have to move pointers and cursors within the interface to indicate where actions should occur. To do so, users employ *navigation* methods that vary, depending on the cursor's location in the interface. Therefore, navigation refers to how users move pointers and cursors within the interface.

Users often need to indicate which element of the interface they want to interact with. *Selection* enables users to identify individual or multiple elements for subsequent interaction.

Activation refers to using controls to perform actions. When a user chooses a button or chooses an item from a menu, for example, the user is activating those controls.

The following sections outline Common Desktop Environment requirements for input, navigation, selection, and activation.

Input Guidelines

Virtual keys are names for generic operations that the user can perform through the interface. They may be mapped to one or more physical key combinations. Application developers are strongly urged to provide support for Help, Properties, Undo, Cut, Copy, and Paste if their application has functions corresponding to these virtual keys. Other virtual keys may be supported as appropriate.

Required **a:** Components and applications that have functions corresponding to the Motif/Common Desktop Environment virtual keys must support those keys.—***page 148***

See the checklist item for a list of the virtual keys and their mappings.

Navigation

The desktop allows either BSelect or BMenu to be used to display menus. This capability has been added for users familiar with environments that provide a dedicated mouse menu button.

For mouse-based navigation of text fields, the desktop has added a requirement that the text cursor be placed at the mouse cursor position, rather than at the beginning or the end of the text field.

Mouse-Based Navigation

Optional **b:** BMenu Press or BMenu Click on a menu bar item displays the menu.—***page 153***

Required **c:** BMenu Press or BMenu Click on an option button displays the option menu.—***page 153***

Required **d:** BSelect Press on a text field causes the text cursor to be inserted at the mouse cursor position.—***page 153***

Keyboard-Based Navigation

The requirement to support Tab as a navigation key within groups of push buttons has been added to make keyboard navigation easier for users who have difficulty with the standard keyboard. This change is intended to reduce the number of side-to-side traversals of the keyboard (from Tab to arrow keys and back) that a user must perform to navigate within a single dialog box.

Optional **e:** Each time a new window is opened, keyboard focus is placed in the first field or location within the window or in a default location, if this is appropriate for the particular window.—***page 157***

Required **f:** The Tab key moves input focus between push buttons within a group.—***page 157***

The arrow keys also move the selected focus per the OSF/Motif Style Guide, Revision 1.2.

Required **g:** Use the Control, Shift, and Alt keys only to modify the function of other keys or key combinations.—***page 158***

Optional **h:** Use the Alt key only to provide access to mnemonics.—***page 158***

Selection

The desktop has incorporated two significant changes to selection in Motif. The first is that users may elect to have either Adjust or Transfer capability on the middle mouse button. In addition, the desktop integrates drag and select on the first mouse button.

On a three-button mouse, button 2 is typically used for the BTransfer (or BSelect) function. However, in a Common Desktop Environment environment, the user may change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. BAdjust can be used to toggle the selection state of elements under the multiple selection model.

The following guidelines describe the BAdjust behaviors.

Mouse-Based Multiple Selection

Required **i:** If your application contains collections that follow the multiple selection model, BAdjust is supported and behaves just like BSelect, when the BTransfer button is currently configured to behave as BAdjust.—***page 163***

Required **j:** In a collection that uses multiple selection, clicking BSelect or BAdjust on an unselected element adds that element to the current selection. Clicking BSelect or BAdjust on a selected element removes that element from the current selection. Clicking BSelect or BAdjust moves the location cursor to that element.—***page 163***

Mouse-Based Range Selection

Required **m:** If your application contains collections that follow the range selection model, BAdjust is supported and behaves just like Shift+BSelect, when the BTransfer button is currently configured to behave as BAdjust.—***page 165***

Required **n:** In a collection that uses range selection, when the user presses Shift+BSelect, or BAdjust, the anchor remains unchanged, and an extended range for the selection is determined, based on one of the extension models.—***page 165***

Reselect The extended range is determined by the anchor and the current pointer position, in exactly the same manner as when the selection was initially made.—***page 165***

Enlarge Only The selection can only be enlarged. The extended range is determined by the anchor and the current pointer position, but then is enlarged to include the current selection.***—page 165***

Balance Beam A balance point is defined at the midpoint of the current selection. When the user presses Shift+BSelect or BAdjust on the opposite side of the balance point from the anchor, this model works exactly like the reselect model. When the user presses Shift+BSelect, BAdjust, or starts a navigation action modified by Shift on the same side of the balance point as the anchor, this model moves the anchor to the opposite end of the selection and then works exactly like the reselect model.***—page 165***

When the user releases BSelect or BAdjust, the anchor does not move, all the elements within the extended range are selected, and all the elements outside of it are deselected.—*page 165*

Mouse-Based Discontiguous Selection

Required **o:** In a collection that uses discontiguous selection, BAdjust can be used to extend the range of a discontiguous selection. The extended range is determined in exactly the same way as when BAdjust is used to extend a range selection.***—page 167***

Component Activation

The following guidelines have been provided to clarify double-click timing and mnemonics, to explain changes in activation of specific components, and explain behaviors of components that are new in TED Motif.

Basic Activation

Required **x:** The time allowed to detect a double click (*doubleClickTime: 500) should be no less than 500 milliseconds.—***page 182***

Mnemonics

Required **y:** Mnemonic characters must be chosen for ease-of-location within the text of a label. Wherever possible, use the first character of the label. If that is not possible, try to use the last character of the label, or if there is more than one word, the first character of the second word. After that, go through the label from the second character on until a unique mnemonic is found.—***page 184***

CheckButton

Required **7-1:** The application uses check buttons to select settings that are not mutually exclusive. A check button graphically indicates its state with the presence or absence of a check mark.—***page 234***

A check button is used to select settings that are not mutually exclusive. The user needs to know whether the button is set or not.

OptionButton

Required **7-23:** When the user presses BSelect or BMenu in an option button, the associated option menu is posted.—***page 243***

BSelect Press is a consistent way of activating an option button.

Required **7-24:** When the user releases BSelect or BMenu within the same option button that the press occurred in, the associated option menu is posted if it was not posted at the time of the press. When the user releases BSelect or BMenu outside of the option button, the associated option menu is unposted.—***page 243***

BSelect Release or BMenu Release posts or unposts an option menu, depending on whether the release occurs inside the option button and whether the option menu was posted at the time of the press.

Gauge

Required **ib:** A gauge is similar to a scale except that a gauge is a display-only device with no user interactions. The appearance of a gauge is similar to a scale, but the gauge lacks a scale slider.—***page 254***

Optional **ic:** Despite being a display-only device, a gauge should get keyboard focus so that the user can access Help or Settings for that control.—***page 254***

Drag and Drop 3

Drag and drop enables the user to directly manipulate objects in the computing environment. This chapter discusses and provides guidelines for incorporating drag and drop into your application. If you plan to use drag and drop or attachments in your application, then you should read this chapter.

You should be familiar with the description of Motif 1.2 drag and drop as covered in the *OSF/Motif Style Guide, Version 1.2*. If you are not, read Section 4.3.4, "Drag Transfer," in that book. Where differences occur, this chapter supersedes the *OSF/Motif Style Guide, Version 1.2*.

Drag-and-Drop User Model

Direct manipulation of objects in the computing environment helps the user feel in control of the computer, which in turn helps the user feel more comfortable about trying new operations and about computers in general. It is also a more efficient method for performing many operations.

To understand the drag-and-drop user model, you must understand the following terms:

drop zone An area of the workspace that accepts a dropped icon. Drop zones are usually represented by the control or icon graphic, such as the Trash Can or the Print Manager control.

drag icon The composite cursor used during the drag. See "Parts of a Drag Icon" on page 19 for details.

In the Common Desktop Environment, the user can select and drag icons in File Manager, mail messages and attachments in Mailer, appointments in Calendar, and text from text editors or text fields. These items can be dropped onto any *drop zone* that accepts them. For example, a document icon from File Manager can be dropped onto a folder icon in File Manager, onto the Print Manager icon on the Front Panel, or on the attachment list in Mailer.

When an item is dropped, some action happens with the dropped item. For example, the document might get printed, moved, or attached. The action taken depends on the object being dragged and where it is dropped.

Recommended **dq:** You should provide a drag-and-drop (DND) method for all objects represented as icons. Provide a DND method for all elements that the user can directly manipulate.*—page 219*

Recommended **dr:** Any basic function that your application supports through drag and drop is also supported through menus, buttons, or dialog boxes.*—page 219*

Drag and drop is considered an accelerator to functionality that is accessible through other user interface controls supported within your application. There should be no basic operation that is supported solely through drag and drop.

Drag-and-Drop Feedback

Visual feedback to the user is one of the critical elements for making drag and drop work. Without clear visual feedback, it is difficult for the user to predict the results of a drag-and-drop operation. Visual feedback must also be timely.

The user should be able to identify visually the following items during a drag-and-drop operation:

- Type of item being dragged—The drag icon should show the user what type of item is being dragged.
- Drop zone—The drop zone should visually indicate when the dragged item is over the drop zone, and whether the drop zone is valid or invalid for that item.
- Resulting activity—The drag icon should visually indicate what action will take place when the item is dropped.

- Success or failure—When the item is dropped, transition effects should indicate the success or failure of the drop.

Type of Item Being Dragged

The visual feedback for drag and drop is based on the concept that what you see is what you drag. For example, if the user selects a folder icon in File Manager and starts dragging it, the drag icon should show the folder icon as part of the drag icon, as illustrated in Figure 3-1.

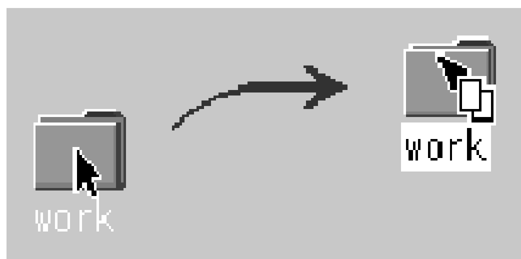


Figure 3-1 Drag icon showing the object being dragged

Required **dt:** During a drag operation, your application changes the current pointer to a drag icon.—***page 220***

Recommended **du:** During a drag operation, your application changes the current drag cursor to include a source indicator.—***page 220***

While this may seem like an obvious behavior, it is critical to give users feedback showing exactly what they are dragging. Providing this kind of consistency makes drag and drop more predictable to the user. Text drags are the exception in that the actual text is not dragged. Figure 3-2 shows a text drag icon.

Figure 3-2 Example of a text drag icon

Applications are responsible for supplying a graphic to be used in the drag icon. This graphic is usually one of the icons already supplied with the application, such as the 32x32 icons used in File Manager to represent documents. The graphic used depends on what is being dragged.

In cases where the user does not select an icon to start a drag, it is still appropriate to show a relevant graphic in the drag icon especially if that graphic will be used by the destination application upon drop. For example, in Calendar Appointment Editor, the user can select an appointment from the scrolling list, which does not show icons. An appointment icon is used as part of the drag icon. If the appointment were dropped on File Manager, File Manager would display the appointment using the same graphic.

Drop Zone

The user needs to know when the dragged item is over a valid drop zone. There are two pieces to this feedback; the drag icon and the drop zone. The drag icon changes from an arrow to a cannot pointer when the user moves the drag icon over anything but a valid drop zone. This behavior is sometimes referred to as *drag over feedback*. On the desktop, there is no differentiation between invalid drop zones and something that is not ever a drop zone. Figure 3-3 shows a sample cannot pointer drag icon.

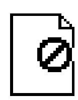


Figure 3-3 Example of a cannot pointer drag icon

The drop zone feedback options (referred to as *drag under feedback*) indicate valid drop zones. The options include a solid line drawn around the site, a raised or lowered surface with a beveled edge around the drop zone, or drawing a pixmap over the drop zone. The beveled edge effect is used to make the drop zone look recessed. Figure 3-4 shows the recessed appearance. On the desktop, most drop zones use the recessed appearance when a drag icon is over the drop zone to indicate it is a valid drop zone.

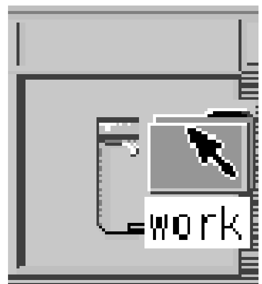


Figure 3-4 Example of drop zone feedback from the Front Panel.

Recommended **dw:** During a drag operation, your application changes the drop zone feedback to indicate a valid drop zone.***—page 220***

Recommended **dv:** During a drag operation, your application changes the current drag cursor to indicate invalid drop zones. It uses the standard Common Desktop Environment cannot pointer.***—page 220***

Optional **dz:** Any cursor change or drag-over effect your application uses occurs within .2 seconds of the mouse pointer reaching the target area and does not interfere, in any noticeable way, with the interactive performance of the drag operation.***—page 221***

Resulting Activity

In the Common Desktop Environment, there are three operations that can be associated with the drag icon. These operations are explained in "Parts of a Drag Icon" on page 19. The drag icon has alternate graphics, supplied as part of the desktop, used to indicate to the user when the operation is a copy or link. The default operation, move, requires no alternate icon graphics.

Success or Failure

The user needs an indication that the drag-and-drop operation either succeeded or failed. You should use transition effects to indicate the success or failure of the drop.

There are two kinds of transition effects: *melt* and *snap back*. The *melt* effect is used when the user drops a drag icon on a valid drop zone. The effect looks like the drag icon melts into the drop zone. The drag icon goes away and is replaced by whatever is appropriate for the destination application. Dropping a drag icon on the Print Manager control on the Front Panel may show nothing other than the melt in effect. Dropping a drag icon on the File Manager control would show the melt in effect followed by the icon appearing in File Manager.

The *snap back* effect is used when the drop fails. Drops can fail in two ways: because the drop zone is invalid, or because the data transfer fails. If the user drops a drag icon over an invalid drop zone, one that shows the cannot pointer drag icon, then the drag icon snaps back to the source application.

Once a drop occurs, the source and destination applications have to transfer the data. If the data transfer fails, there are two things that the destination application should do. The first is to indicate to the API that the drop failed so that the dropped item will get snapped back to the source application. The second is to put up an error notice to the user that clearly indicates why the drop failed and what, if anything, the user can do to correct the situation.

Sometimes the transition effect does not take place immediately. The icon appears where it is placed until the transfer is done. During this time, applications should set the cursor to the busy state. The icon cannot be moved or selected by the user until the transfer is complete; the busy cursor tells the user the transfer is in process.

Recommended **ea:** In a collection that supports copy, move, or link operations that can be performed by dragging, the feedback presented to the user during the drag operation indicates whether a single object or multiple objects are being manipulated.—***page 221***

Required **eb:** After a successful transfer, the data is placed in the drop zone, and any transfer icon used by your application is removed.—***page 221***

Required **ec:** If your application removes data upon the completion of a drag and drop, it does so only if the drag-and-drop transfer has completed successfully.—***page 221***

Required **ed:** After a failed transfer, the data remains at the drag source and is not placed in the drop zone. Any transfer icon used by your application is removed.—***page 222***

Recommended **ee:** If the user drops an object at an inappropriate drop zone within your application's window, your application participates in the display of a snap back effect and also posts an error dialog box indicating the reason the drop was disallowed.—***page 222***

Required **6-17:** If your application provides any drag-and-drop help dialog boxes, they contain a Cancel button for canceling the drag-and-drop operation in progress.*—page 222*

Parts of a Drag Icon

The drag icon is composed of three parts, as shown in Figure 3-5. Starting from the left, the drag icon is composed of the state indicator plus the operation indicator plus the source indicator (here shown as a folder icon). On the right is the composited drag icon.

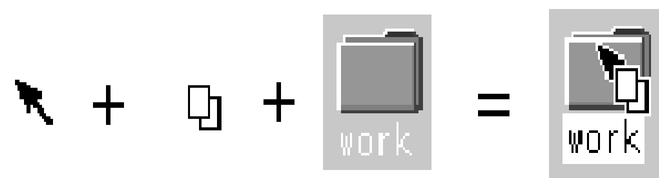


Figure 3-5 Example drag icon showing the three parts

State Indicator

The *state indicator* is really a pointer for positioning combined with a valid or invalid drop zone indicator. The valid state indicator should resemble an arrow pointer with a hot spot so users can position the cursor in a predictable manner. The invalid state indicator, a cannot pointer (illustrated in Figure 3-3), appears when users are over an invalid drop zone.

Operation Indicator

The second part of the drag icon is the operation indicator. A drag operation can be a *move*, *copy* or *link*.

move The item being dragged is moved to the destination.

copy The item is copied to the destination.

link A connection is retained between the destination and the source. This operation is defined to some extent by the application and is not commonly used.

See “Actions” on page 23 to see how move, copy, and link map to user actions.

The *operation indicator* gives the user feedback on what operation is occurring during the drag. Figure 3-6 shows the copy and link feedback. Because most drags are move operations, an operation indicator is added to the drag icon only for copy or link operations.

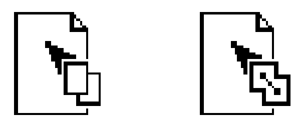


Figure 3-6 Examples of copy (left) and link operation indicators

---

Note – Operation feedback is drawn on top of the state and source feedback. This is basic Motif behavior.

The user can force a drag to be a move, copy or link by pressing certain keys during a drag (Shift = move, Control = copy, and Shift+Control = link).

The source application can also force a copy. When the user forces an operation, the drop zone should match that operation for the drop to succeed; otherwise the drop zone should indicate the operation is invalid.

Required **4-36:** If the move, copy, or link operation the user requests is not available, the transfer operation fails.—***page 173***

Required **4-55:** In a collection that supports selection, Shift+BTransfer Release or Shift+BSelect Release forces a drag move operation. If a move is not possible, the operation fails.—***page 178***

Required **4-56:** In a collection that supports selection, Control+BTransfer Release or Shift+BSelect Release forces a drag copy operation. If a copy is not possible, the operation fails.—***page 178***

Required **4-57:** In a collection that supports selection, Control+Shift+BTransfer Release pr Shift+BSelect Release forces a drag link operation. If a link is not possible, the operation fails.—***page 178***

Recommended **s:** In a collection that supports selection, if BTransfer Motion (or BSelect Motion) results in the start of a drag operation, feedback is presented to the user that indicates that a copy, move, or link operation is in progress. Whether the operation is a copy, move, or link depends on the type of object created at the drop zone and whether the source object is removed.***—page 180***

Recommended **t:** In a collection that supports selection, if Control+BTransfer Motion or Control+BSelect Motion results in the start of a drag operation, feedback is presented to the user that indicates that a copy operation is in progress.***—page 181***

Recommended **u:** In a collection that supports selection, if Control+Shift+BTransfer Motion or Control+Shift+BSelect Motion results in the start of a drag operation, feedback is presented to the user that indicates that a link operation is in progress.***—page 181***

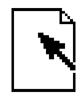
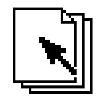
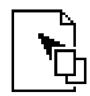
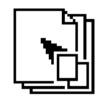
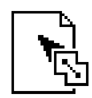
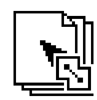
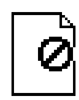
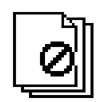
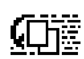
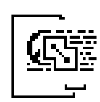
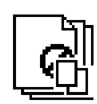
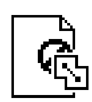
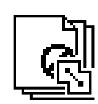
Source Indicator

The *source indicator* is a representation of the selection (or the thing being dragged). It comes in several versions depending upon whether the selection represents single or multiple items and what kind of thing the selection is representing. Table 3-1 shows examples of drag icons.

Note – Since the drag icons are dynamically composited as cursors, screen shots cannot be taken; therefore, Table 3-1 is an approximation of what is seen on the screen and may not be completely accurate.

The hot spot is located on the top left corner (1,1) for the text drag icons. The hot spot for the single and multiple drag icons is located at the top left pixel of the invalid icon (3,3). Each drag icon has been tuned to increase user accuracy at targeting and positioning.

Table 3-1 The possible drag icons shown with generic source indicators.

Operation	Text Selection	Single Selection	Multiple Selection
Valid Move			
Valid Copy			
Valid Link			
Invalid Move None			
Invalid Copy			
Invalid Link			

Drag-and-Drop Mechanics

There are several areas of the underlying application architecture that are useful to understand when designing drag and drop.

- What type of object is being dragged.
- What action takes place when the object is dropped.
- How to match operations between source and destination applications.

Types

In the Common Desktop Environment, there are three types of draggable objects: files, buffers, and text selections.

Each application has its own objects that can be dragged and dropped. For example, Calendar uses appointments, Mailer uses mail messages, and File Manager uses folders and files. The folder and file icons in File Manager exist as separate entities in the underlying file system and are, therefore, treated as files when dragged and dropped. However, Calendar appointments and Mailer messages do not exist as separate entities in the file system. When these objects get dragged they are treated as buffers.

This difference can lead to some conflicts for the user. The user sees both of these types of objects as the same—both can appear as icons and both can be manipulated separately from other similar objects. Yet, due to the implementation, the user may see different results from a drag-and-drop operation based on which type of object is being manipulated.

Text selections fall into a different category because selecting a piece of text is very different to the user than selecting an icon. The user selects a range of text in a document window; the text does not represent the whole document, it only represents a piece of a document. Rarely does a user see the piece of text as a distinct object and the user does not expect a piece of text to behave like an icon when dropped. For this reason, the drag-and-drop model for text mirrors the cut, copy, and paste operations available from the Edit menu.

Actions

There are two actions that can take place when an object is dropped: insert or load.

The insert action inserts the dropped object into the destination, adding it to the current data in the application or document. The object is inserted when a user schedules an appointment, prints a document, attaches a document, pastes text, or appends a mail message. Such an action is a move or copy operation depending on the destination and the user. The user might decide to copy a piece of selected text, as opposed to moving it. The drag icon should indicate whether the operation is a copy or move.

The load action operates the same as if the user had chosen Open from the File menu, selected a file, and clicked the Open button. The dropped object gets loaded into the application. The user can edit it and save changes back to the original file. Load only works with files at this time, not with buffers or text. The user should see the copy drag icon when dragging an object over a drop zone that supports the load action.

Load does work with buffers; however, buffers are loaded as read-only. See the section on "Attachment User Model" for more details.

Matching Operations

When designing how drag and drop works in an application, you must understand how Motif figures out what operation gets done when the source and destination of a drag and drop don't match.

For each drag source, an application advertises which drag-and-drop operations are possible and on what destinations it can be dropped. For each drag destination, the application advertises the possible sources and the types of operations. If a source and its destination have two or more operations in common, Motif follows a specific order to determine which operation to use. That order is move, copy, link. The application cannot change the operation that is accepted based on the type of thing being dragged.

For example, application A might advertise that an element can be moved or copied. Application B advertises that the destination accepts copy or link. The intersection in this example is copy. If the destination in application B accepts move or copy, then the source is moved because the move operation comes first in the operation order.

In this example, the user could override the move operation by holding down a modifier key, for example the Control key to make the operation a copy. This will work if the copy operation is in the common set of operations. If the copy operation is not in the common set, then the drag becomes an invalid drag.

The only time matching operations may be a consideration is when you have a destination that could accept moves but prefers copies. In that case, the destination is better off only accepting copies.

It is wise to always accept copy. Accepting copy broadens the scope of acceptable drops. In most cases where a move is accepted, a copy would work just as well. Remember, move is implemented as a copy followed by a delete.

Determining Drag Sources

When you decide to use drag and drop in an application, you must decide what control elements can be dragged and how that element is to be represented. Typically, the user selects something like text or a file to drag, but the application may have other elements for which drag and drop may make sense, such as mail messages or appointments.

This section provides general guidelines for determining drag sources and then talks about specific elements that can be dragged.

Note – Drags are only sourced from human interface elements that have selectable components or items. Drags cannot be sourced from static labels like those on buttons or menus.

Before you decide on a drag source from your application, you must consider how the drag is to be integrated into the selection mechanism of your application. The user must be able to select or drag without any confusion as to the result of that action.

Required **4-58:** When a drag move operation moves a selection within the same component, the selection moves along with the elements selected.—***page 178***

In other words, when selected elements are moved with a drag operation, they should stay selected after the move.

Required **4-59:** In text-like collections, initiating a drag within a selected region drags the entire text selection.—***page 179***

Required **4-60:** In list-like and graphics-like collections, initiating a drag with either BSelect or BTransfer on a selected element drags the entire selection.—***page 179***

Required **4-61:** In list-like and graphics-like collections, initiating a drag with BTransfer or BSelect on an unselected element drags just that element and leaves the selection unaffected.—***page 179***

Required **4-62:** When a drag is initiated in an unselected region and the pointer is over two possible draggable elements, the drag uses the draggable element highest in the stacking order.—***page 179***

Required **4-67:** When BTransfer (or BSelect) is released, the drop operation ordinarily occurs at the location of the hot spot of the drag icon pointer and into the highest drop zone in the stacking order. However, if a drop occurs within a selection and pending delete is enabled, the transferred data replaces the contents of the entire selection.—***page 180***

Scrolling Lists

In the Common Desktop Environment, items in a scrolling list are text objects by default. They can be buffer objects, but they cannot be both text and buffers. For example, the Calendar Appointment Editor has a scrolling list of appointments that the user can select and drag. When the user drags an appointment, they are manipulating a buffer and the drag icon shows an appointment icon as the source indicator, as shown in Figure 3-7. A Mailer container window has a list of mail messages in the upper portion of the window. Users can select and drag one or more messages from this list. These messages are actually buffers and the drag icon shows a mail message as the source indicator. If multiple mail messages are dragged, then the drag icon shows the multiple source indicator.

Figure 3-7 Example of a scrolling list with an item selected for dragging

If your application uses a scrolling list to show mail message headers or list other kinds of objects, then you need to integrate dragging with extended selection. this behavior can be seen in the Mailer application.

Required **k:** In a collection that uses range selection, pressing BSelect on an unselected element sets an anchor on the element, or at the position where BSelect was pressed, and deselects all elements in the collection. If BSelect is released before the drag threshold has been exceeded, then the element under the pointer should be selected. If BSelect Motion exceeds the drag threshold, then a new selection should begin. The anchor and the current position of the pointer determine the current range. As BSelect is dragged through the collection, the current range is highlighted. When BSelect is released, the anchor does not move, and all the elements within the current range are selected.*—page 163*

Required **l:** In a collection that uses range selection, pressing BSelect on an currently selected element should not cause all other elements in the selection set to be deselected. If BSelect is released before the drag threshold is exceeded, then, at that point, all other elements should be deselected and the element under the pointer should remain selected. If BSelect Motion exceeds the drag threshold, then no element should be deselected and a drag operation should begin.*—page 164*

Dialog Boxes

Sometimes an application needs to be able to drag from inside a dialog box. For example, in the Calendar Appointment Editor, there are a series of text fields on the left side where the user enters information about an appointment. Allowing drags from this area lets the user drag text from the appointment description.

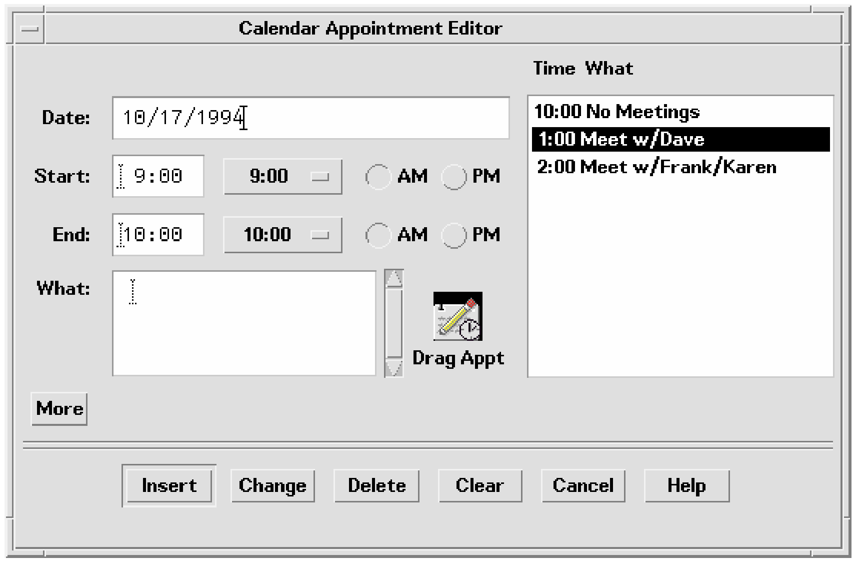


Figure 3-8 Example Calendar Appointment Editor dialog box

The recommended method for indicating that something can be dragged is to include an icon graphic in the dialog box. The icon graphic should be draggable. This icon graphic must be the same icon used to represent the data in File Manager. In Calendar, the appointment icon is shown just as it would appear in File Manager (see Figure 3-8), with a label under it. This is the same icon used in the drag icon source indicator.

Place the icon graphic in the dialog box adjacent to the information to be dragged. The upper right corner of the dialog box or window is the default position, but it can be changed depending on the application. In Calendar Appointment Editor, the icon is placed near the main text field to indicate that you can drag the text fields.

Recommended **ds:** Use an icon graphic in a dialog box or window to indicate that objects within the dialog box or window can be dragged. Use the same icon graphic used to represent the draggable object in File Manager. Place the icon adjacent to any display of the contents of the object, if such display exists. If there is no such display, place the icon in the upper right corner of the dialog box or window, unless a more suitable placement is determined. The icon should be 32x32 in size and have a label under it.

The label should indicate what kind of object the icon graphic represents. The icon graphic should also be used as the source indicator in the drag icon.—***page 220***

Windows

In some applications, you may want to allow the user to drag the entire document or file. For example, in Icon Editor, it might make sense to allow the user to drag the file for the icon currently in the editor. The application should indicate to the user that the document or file can be dragged by incorporating an icon graphic in the window of the application. The icon graphic should be draggable. In the case of Icon Editor, one of the icons used for displaying the contents of the icon file could be used for the drag. The icon should follow the guidelines for an icon used in dialog boxes; that is, it should be the same icon used to represent the document in File Manager, be 32x32 pixels, have a label, be placed adjacent to the data being dragged, and be used as the source indicator in the drag icon.

Dragging and Dropping a Multiple Selection

When the user selects more than one item to be dragged, the drag icon should change to indicate there is more than one item selected.

Some drop zones may be able to accept only a single item. It is not possible for a drop zone to differentiate between a single and a multiple set of items being dropped. The drop icon does not display the cannot pointer; instead the items should melt in and then be snapped back by the destination application. The snap back should be followed by an error notice that tells the user why the drop failed.

Recommended **ef:** Applications that accept only single items should reject all multiple-item drops.—***page 222***

There is no consistent method to determine which of the selected items the user really wants to drop.

Standard Supported Drop Zones

The standard supported drop zones in the Common Desktop Environment are Front Panel controls, open windows, and folder, action, and certain other icons in File Manager. Dropping on minimized icons and on File Manager icons that do not support drops are not supported in the Common Desktop Environment.

The Front Panel

The Front Panel is a collection of controls and other functions put together for easier and faster access for users. As a consequence, its drag and drop behaviors are heavily dependent upon the context of the destination. For example, if the destination is a printer, then print it. If the destination is a subpanel, then install it. Most applications will not vary in behavior quite as broadly as the Front Panel.

File Manager

File Manager for the Common Desktop Environment allows users to drop icons on the desktop. The icon on the desktop becomes a reference. The creation of the reference and resulting behaviors are not consistent with the future user model for the Common Desktop Environment. Until the user model and architecture are further specified, developers are *not* encouraged to do any drops onto the desktop or to copy the File Manager behavior.

Within File Manager windows, File Manager allows dropping onto icons other than folders and action icons for the Common Desktop Environment 1.0 or TriTeal Enterprise Desktop 4.0. For example, dropping a mail message icon onto a mail container icon appends the mail message.

When mail messages or calendar appointments, or other buffers are dragged from the source application and dropped onto File Manager, they must be named. The underlying API supports a name field for the item being dragged. This name should be used as the name of the buffer. The name should be determined in a manner consistent with the application from where it came. If there is no appropriate name, as in dropping a text selection in File Manager, File Manager should name the resulting file “unnamed”. If there is a name conflict, File Manager should put up a dialog box and ask the user to rename the dropped file.

No Drop-Only Targets

The Common Desktop Environment does not support the concept of a specific control or graphical target used only for drops. Any control in the human interface that has selectable items can be dropped upon and should provide drop zone feedback. This includes data panes, scrolling lists, and text entry fields. The operation that takes place upon the drop should be consistent with the users expectations for that application type.

Mouse Button Usage

In the Common Desktop Environment, the user can modify mouse mappings to accommodate different working styles. BSelect has been modified to support drag and drop in addition to BTransfer, which has been used historically for Transfer operations in Motif.

Users can set up their systems to use Bselect or BTransfer to perform drag and drops, or use BSelect only. Consider this when you design your application. Check how the user has set the mouse button mappings and use those mappings.

Required **p:** Your application supports the use of mouse button 1 to perform drag-and-drop operations.*—page 174*

In Motif 1.2, drag and drop is typically performed using button 2 on a three-button mouse (BTransfer). However, in the Common Desktop Environment environment, mouse button 1 (BSelect) should be supported for drag and drop to support mouse usage compatible with other graphical user interface (GUI) environments. A drag can be initiated with either mouse button 1 or mouse button 2.

Required **q:** When button 2 of a three-button mouse is configured to operate as BAdjust, your application does not perform any BTransfer operations when clicking mouse button 2.*—page 174*

On a three-button mouse, button 2 is typically used for the BTransfer function. However, in a Common Desktop Environment environment, the user may change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. BAdjust can be used to extend the selection set in the same manner as Shift+BSelect.

Required **r:** BSelect should always initiate a drag if the drag is started on a selected item. The drag starts once the drag threshold has been reached. This is true for text regions, scrolling lists, and other similar elements.—***page 175***

Placement upon Drop

From the user's point of view, the placement of the dropped item is dependent on the task the user is doing and the application or context the task is in.

In File Manager, if the default is set to As Placed, then icons are placed where they are dropped. If the default is set to Sorted Grid, then a dropped icon is automatically sorted and then placed, which means it may not be placed where the user drops it.

In some cases, where the dropped item gets placed is not a criteria. For example, Front Panel controls require only that the dragged item be over the control to activate the drop zone.

In the Compose window in Mailer, the placement depends on what is being dropped. If the user is dragging a piece of text, then the text is inserted at the *drop point*. From user testing, this is what users expect. If the user drops an icon, a file or a buffer, then the contents are included at the *insertion point*. This mirrors the behavior the user gets when the user selects a file from the Include File Selection Box.

You should determine appropriate behavior for your application based on what type of tasks the user is doing.

Required **4-37:** If a collection does not have a fixed insertion point or keep elements ordered in a specific way, the insertion position for transferred data is determined as follows:—***page 174***

- For BTransfer-based (or BSelect) primary and drag transfer operations, excepted as noted below for text collections, the insertion position is the position at which the user releases BTransfer (or BSelect).
- In a text-like collection, when the user drops selected text, the insertion position is the position at which the user releases BTransfer (or Bselect). When the user drops an icon, the insertion position is the text cursor and the data is pasted before it.

- In a list-like collection, the insertion position for other transfer operations is the element with the location cursor, and the data is pasted before it.

The insertion position is the position in the destination where transferred data is placed. Some mouse-based transfer operations place data at the pointer position if possible. Other operations, including keyboard-based transfer, generally place the data at the location cursor.

Ending a Drag

The following two items allow users to stop a drag operation without any data loss or other negative result.

Required **dx:** Pressing Cancel ends a drag-and-drop operation by canceling the drag in progress.—***page 221***

Required **dy:** Releasing BTransfer (or BSelect) when not over a drop target ends a drag-and-drop operation.—***page 221***

Performance Guidelines

There are several points during a drag-and-drop operation that the timing and response to the user is critical. The responsibility for ensuring optimal performance belongs to the source and destination applications, as well as the Motif Drag-and-Drop API and Drag-and-Drop Convenience API.

The following time line explains the individual user steps and system responses in a drag-and-drop operation. The suggested guideline for interaction timing is noted after the relevant step.

1. The user makes a selection. The pointer is over the selected object. The user presses and holds down the mouse button.
2. The user starts to move the pointer. The user should be able to move the pointer 10 pixels before a drag is initiated. If the user is pressing BTransfer, there is no drag threshold.

   ⇨ The pointer changes to a drag icon when the drag is initiated.

   Movement Latency: The latency from hand movement to drag icon display should be less than 50 msec. with a maximum limit of 70 msec.

3. The user drags over a drop zone, crossing the boundary line with the hot spot on the drag icon.

   ➪ The drag icon changes to the cannot pointer if it is not over a valid drop zone. The drop zone becomes highlighted if it is a valid drop zone.

   Movement Latency: The latency from hand movement to drag icon display should be less than 50 msec. with a maximum limit of 70 msec.

4. The user drops the drag icon on the drop zone.

   ➪ If the drop zone is not valid, the drag icon is snapped back to the source using the snap back transition effect.

   ➪ If the drop zone is valid, the drag icon is melted into the destination using the melt in transition effect.

   Echo Latency: The display latency from mouse button release to feedback echo should be less than 50 msec. with a maximum limit of 120 msec.

   Snappy Transitions: Transitional animations should run from 200 to 350 msec. with a maximum limit of 500 msec. The animation should run at the same speed regardless of hardware conditions.

5. The destination application starts the data transfer.

   ➪ A message is displayed to the user indicating data transfer has started.

   ➪ Progress is indicated by further messages.

   ➪ The completion of the data transfer is indicated to the user.

   ➪ If the data transfer fails, it is up to the destination application to provide the user with appropriate feedback as to why it failed.

   Command Latency: The latency from command invocation (drop occurred) to completion should be in the range of 0.3 to 1 second with a maximum of 2 seconds.

   Busy Feedback: When a command may run longer than 2 seconds, display a busy cursor whenever the cursor is over the busy object. When possible, display partial results. The progress indicator or busy cursor should be displayed in less than 0.5 second.

Progress Indicators: A status message or an In Progress *messagebox* should be displayed upon completion of the transitional animation indicating the data transfer is taking place. For example: `Data transfer is 10% complete`. This message can then be updated every 2 to 3 seconds until the transfer is 100% complete.

Notice: If the data transfer fails, a message should appear, either in the status area or in an In Progress *messagebox*, indicating why the drop failed and what the user can do about it, if anything.

Using Attachments in Your Application

This section discusses the user model and guidelines for attaching documents to documents in the Common Desktop Environment 1.0 (TriTeal Enterprise Desktop 4.0). This functionality can be seen in the Mailer software application. If you plan to include an attachment list in the interface of your application, then you should read this section.

Note – This set of guidelines is not a description of an embedded document architecture.

For the Common Desktop Environment, attachment and attachment list are defined as follows:

attachment Suppose you had two documents called A and B. If a document, A, is attached to another document, B, then A continues to exist as a separate document that is "carried" by B. A is shown as an icon within B. A can be opened and viewed independently and can be detached from B at a later time, as if never attached at all.

attachment list The area in which attachments are displayed. Should be scrollable and include room for showing icon labels.

Note – Users do not think of a document that has several attached documents as a container. Containers are an implementation concept that should not appear in the attachments human interface. For that purpose, the term *container* should not be used to describe attachments to the user. (It may be an appropriate term elsewhere.)

Attachment User Model

Attachments are shown as icons where they are attached. These icons are the same icons as those used in File Manager and other places in the Common Desktop Environment. The basic rule of behavior is that if the same icon is used in File Manager as in an attachment, then every effort should be made to make the two behave the same in every situation.

There are three levels of functionality for attached documents:

1. Ability to attach and detach documents.
2. Ability to open, view, and quit an attached document in a separate window.
3. Ability to edit the attachment in a separate window and save changes back to the attached document.

The goal is to provide Level 3 functionality whenever possible. If an attachment cannot provide this level, then it should degrade its level of functionality in the steps shown. This section is written assuming Level 3 functionality.

If a document provides significantly different functionality as an attachment from that provided as a File Manager icon, then provide a different icon for the attachment to clearly indicate to the user the difference in functionality.

Attachment Functionality

To incorporate attachments into an application, several issues should be considered.

What Can Be Attached?

You should determine for each application what items it can attach. For example, Mailer can attach documents, scripts, and applications, but not folders.

What is the Method for Attaching?

There are two methods of attachment, through the file selection dialog box that comes up when you choose Add File from the Attachment menu, and through drag and drop from File Manager or another application.

Recommended **eh:** Drag and drop should not be the only method for attaching objects.*—page 223*

Recommended **el:** When the user chooses something to attach from the file selection dialog box that is not an attachable item, then the user receives an error message explaining why the chosen item cannot be attached. For example:*—page 223*

```
The folder "My.Stuff" cannot be attached because it is a folder.
Only documents, applications, and scripts can be attached.
```

Recommended **ej:** When the user attempts to drop something into the attachment list that is not attachable, then the drop fails and the item is snapped back to its source.*—page 223*

What Happens When Something Is Attached?

The act of attaching document A to document B copies the bits of document A into document B. There is no further connection with the original file. If the user opens the attached document and makes changes, the changes are saved back to the attached document only, not back to a file in the file system.

Attachments in Attachments

Users can attach messages or text files that have attachments inside them. This is sometimes referred to as "nesting". The user reading the text file would perhaps see a mail message icon that the user could then open, which may have a text message and more attachments.

Editing and Saving Attachments

The user should be able to open an attachment, edit it, and save the changes back to the attachment. If the attachment does not have the ability to do this, then the Open action should not appear in the Actions portion of the menu when that attachment is selected and double-clicking should not open the attachment.

Recommended **ei:** Double-clicking is a shortcut for selecting the attachment and choosing the Open menu item for attachments and should never be the only way to access attachments.*—page 223*

Recommended **ek:** When the user has one or more attachments open for editing and attempts to do any operation that would result in potentially losing the user's edits, the user should be clearly warned and given the opportunity to save changes.*—page 223*

Executing Attachments

If the user tries to open or double-click on an executable attachment, then there may be times when the user should be asked to confirm this operation. Both the name of the attachment and the name of the action being taken on the attachment should be variables. An example error message follows:

```
"Invitation" is an executable attachment. Do you want to Run it?
Buttons: Run, Cancel, Help
```

Read-Only Attachments

Read-only attachments can be opened for reading only. This state should be indicated to the user by inactivating the menus in the attachment application, inactivating the selection cursor, or some other obvious method. At a minimum, the Save menu item in the attachment application should be dimmed.

Drag Load and Attachments

Drag load can be accomplished in two ways. In applications that directly support drag load, users can drag an icon, from File Manager, over the open window for that application and drop it which loads the file represented by that icon. The same result can be accomplished by dropping an icon onto an action icon. The action starts an editor, which then loads the file represented by the icon. When an icon from File Manager is drag loaded, it is equivalent to choosing Open from the File menu. The open file can be edited and saved.

In the case of attachments, users can drag and drop an attachment onto editors or actions that support drag load but any edits made are not saved back to the attachment. Attachments, which are implemented as buffers, are loaded as read only data.

When the user tries to save changes to a loaded attachment, the editor displays a file selection dialog box and asks the user to confirm the name and to choose a place in the file system to save the file. The name used in the file selection dialog box is the same as the attachment name. If the editor (command line application) cannot bring up a file selection dialog box, then it should clearly and visibly indicate to the user that the loaded file is read-only.

If the user wants to edit the attachment directly, the user must select the attachment in the attachment list, choose Open from the Attachment menu or double-click on the attachment. This opens the attachment in a manner that allows for editing and saving changes.

Another option is to drag load an attachment, edit it, save it to a new file name, and replace the old attachment with the new one manually.

Attachment Menu Contents

Recommended **bw:** If your application uses an attachment menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application.—***page 204***

Recommended **Add File...** Selects files and other items to be attached. A file selection box is displayed allowing the user to select the desired files to attach. The default button in the file selection box is Attach.—***page 204***

Recommended **Save As...** Saves the currently selected attachments. The user is prompted with a file selection dialog box for indicating where in the file system the attachments are to be saved. When multiple attachments are selected, the name field is inactive and the current names of the attachments are used as the name of the new file. This menu item is active only when one or more attachments are selected.—***page 205***

Recommended **Rename...** Renames the attachment icon. The application should provide in-line renaming of attachment icons, such as File Manager uses. If the application cannot provide in-line renaming, then Rename allows the user to rename an attachment by displaying a dialog box, requesting the name from the user. This menu item is active only when a single attachment is selected. It is not active when multiple attachments are selected.—***page 205***

Recommended	**Delete**	Deletes attachments from the attachment list. This menu item is active only when an attachment is selected.—***page 205***
Recommended	**Select All**	Selects all the attachments in the attachment list.—***page 205***

Visual Design 4

The Common Desktop Environment is a visually rich environment. This chapter provides information on designing icons and other visuals consistent with the desktop style. It discusses some of the design philosophy behind the desktop, and some useful hints to help you successfully create icons in the desktop.

Icons are graphics that represent the objects (that is, applications, files, and devices) present in a graphical user interface (GUI). Fitting your application into the desktop means designing icons to represent your application and the data files it creates. These icons should be created in several sizes and color depths.

This chapter discusses the desktop components where icons are used, the requirements of the environment, and discusses the design process. A series of examples have been provided that may parallel your own implementation situation.

Color Philosophy

In most other GUIs, the color is applied in a localized and specific sense, either in individual icons or specific control areas, such as window borders or title bars. In the desktop, color is pervasive, as virtually everything is drawn with colors, with a notable absence of black lines.

Most of the icons in this environment use color sparingly, preferring grays instead. This limited use of colors keeps the number of colors used from the palette in the desktop to a minimum and works well visually. Because the icons, being largely colorless, always appear in the context of colored backgrounds, they stand out more.

What Is an Icon?

An icon can be defined as a specific graphical element, one that can be moved, copied, deleted, opened, and so on, usually through direct manipulation.

Numerous graphical elements in the desktop are not manipulable and, therefore, not technically icons, but may still be needed in your application. This book discusses the entire range of graphical elements you may need to provide.

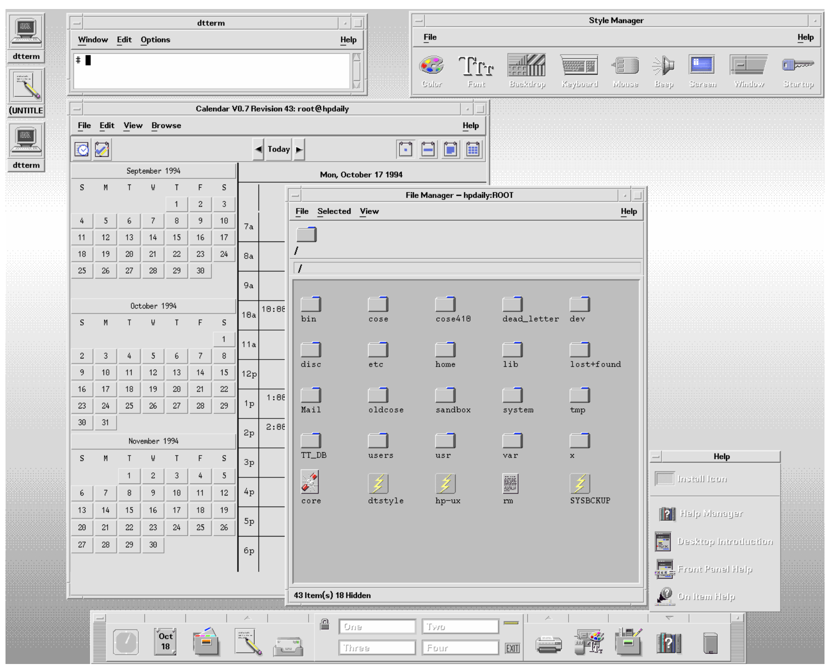


Figure 4-1 Screen shot of the desktop environment

Icons can serve to:

- Identify data and application objects.
- Facilitate direct manipulation of objects.
- Indicate an object's state (selected, and others).
- Convey a recognizable product identity.
- Show the relationships between the objects of a product.

All of these purposes for an icon serve as guidelines for designing icons. The visual designer has more responsibility for some of these requirements than for others. For example, the direct manipulation of objects and the indication of an object's state and location are done by the desktop system, while identifying and conveying product identity and object relationships fall mainly under the responsibility of the visual designer.

Recommended **ey:** Icons should be used to represent only objects and applications.—***page 226***

Icons provide a visual representation for objects and facilitate direct manipulation. If icons are used for other purposes (for example, as illustrations) where the user can't drag them, select them, and so on, it creates a confusing inconsistency.

Applying a set of design guidelines, like the ones here, should be considered during icon design. A new product on the user's desktop means adding a new set of icons to the ones already present. Conforming to these guidelines ensures the new icons do not clash with the user's expectations.

Icon-Centric Components

File Manager

File Manager is the tool that provides for the presentation and organization of the user's file structure. The basic types of iconic objects displayed in File Manager are files, directories (folders), executables and actions. In this chapter, these objects are referred to as documents, folders, and applications. File Manager displays the icons in two sizes, called Icon and Small Icon views in the Set View Properties dialog box. Icon is size 32x32 and Small Icon is size 16x16.

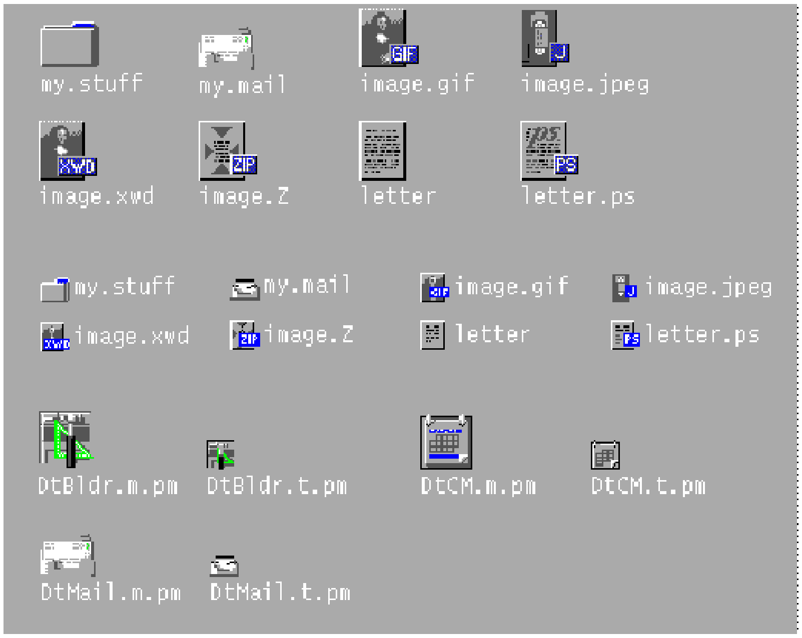


Figure 4-2 Collection of icons at sizes 32x32 and 16x16 as used in File Manager

Documents, folders, and applications are represented by three different shapes. Documents are vertical rectangles meant to look like pieces of paper. Folders are horizontal rectangles with a tab to look like a file folder. Applications can be any shape and use the entire icon square. All objects in File Manager should indicate to the user that they can be manipulated, that is, dragged and dropped.

Application Manager

This window is similar to File Manager, but its focus is on holding applications rather than documents. All network-accessible applications in the desktop are placed here in containers, called *application groups,* rather than folders.

Application Manager is like a "network store." This is the place users go to find the latest applications available on their system.

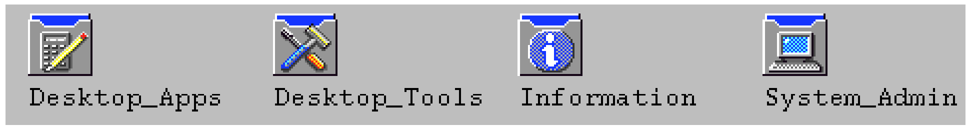


Figure 4-3 Examples of application group icons from Application Manager

Application group icons, as illustrated in Figure 4-3, are like folders in that they represent a collection of objects, in this case related objects. If your application requires support files or comes with sample files, for example, you can design your own application group icon that represents where a user can get the related files for your application.

Front Panel and Subpanels

The Front Panel is the “control” panel for the desktop and usually appears at the bottom of the screen. Front Panel icons provide quick access to the user’s most commonly used applications.

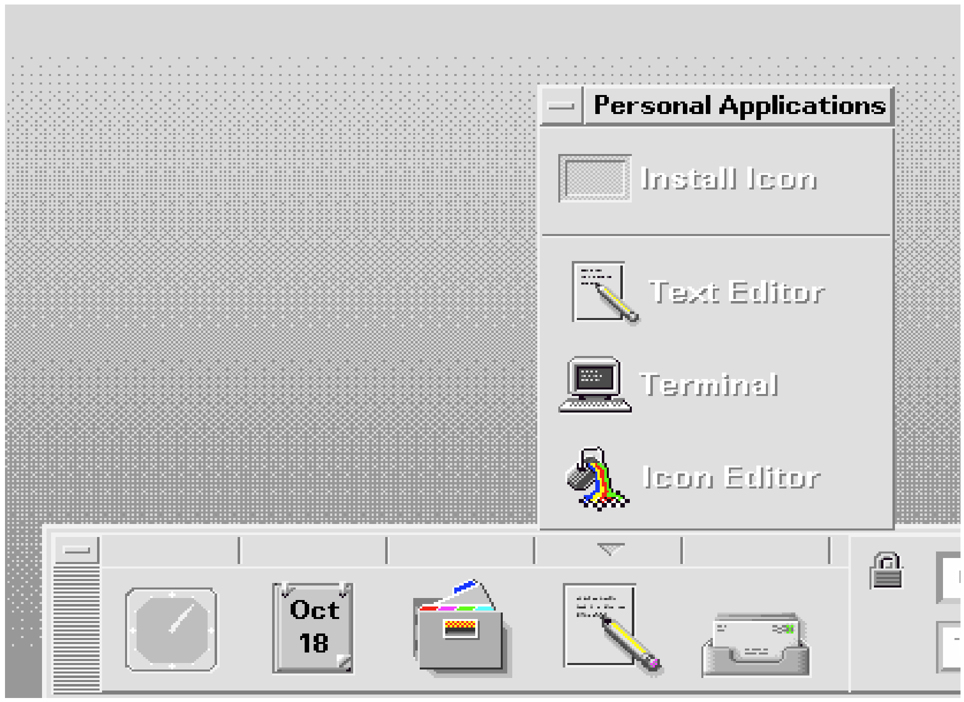


Figure 4-4 Partial screen shot of Front Panel with the Personal Applications subpanel open

The Front Panel also has subpanels of icons that can be accessed through the arrow buttons on the Front Panel. The concept of the subpanel is that it is an extension of that Front Panel icon. For example, Figure 3-4 shows the Personal Applications subpanel open. Users can add applications to this subpanel by dropping them on the Install drop site. Users can choose to promote icons in the subpanel to the Front Panel via the pop-up menu.

Minimized Window Icons

Minimized window icons appear on the desktop when a window is minimized. The icon should represent the application that controls the minimized window (see Figure 4-5). These icons are different from the icons used in the Front Panel in that they represent running applications, although they are the same size.

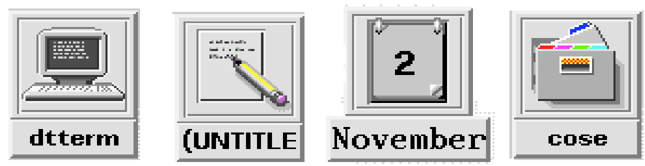


Figure 4-5 Minimized window icons for Terminal, Text Editor, Calendar, and File Manager

Other Graphics

The dominant elements in this category include button graphics, tool bar graphics (see figure below), and graphics used as labels. A tool palette in a paint program is one example. A document orientation button (landscape or portrait) in a printer dialog box is another. These are graphics that you create for use in your application and are not used elsewhere.

Figure 4-6 Example of a tool bar used in the Calendar application

Color Usage in Icons

When designing icons for a desktop application, you must be aware of the available color palette and the dynamic mapping of colors.

Icon Color Palette

The icons in the desktop use a palette of 22 colors:

- Eight static grays
- Eight static colors: white, black, red, blue, green, cyan, magenta, and yellow
- Five dynamic colors: Foreground, Background, TopShadow, BottomShadow, and Select
- A transparent “color” which allows the background to show through

These colors are the default colors in the Icon Editor, which is the recommended tool for creating desktop icons (see Figure 4-7). This set of colors provides a reasonable palette with which to create icons. This limited palette was chosen to maximize the attractiveness and readability of icons without using an unnecessary number of colors.

If you use colors other than the ones listed here, then your icons may experience color flashing effects that can make the icon unreadable. The best way to ensure predictability of appearance of your icons is to use only the 22 colors in the desktop palette.

Recommended **ez:** Icons should use only the palette of 22 colors.—***page 226***

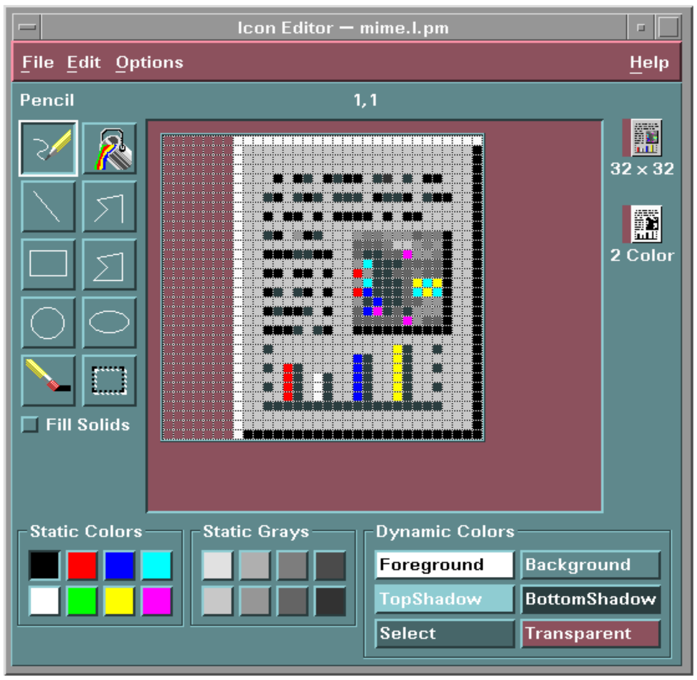


Figure 4-7 Icon Editor, showing the 22 Common Desktop Environment colors available for icons

Dynamic Colors

It is important to understand the limited role of the dynamic colors. These represent the colors used to display the user interface elements on which your icon appears. If your icon appears in File Manager, File Manager determines what the background color is. If the user changes the color palette in Style Manager, the colors in the user interface change to match, and the background color the icons are displayed on changes.

In general, these colors have little use in most icons. There are two ways they are used:

- If your icon does not fill the entire bounding box, then fill the unused area with the transparent color.
- You can draw a shadow under your icon. This is only recommended for Front-Panel-style icons. Do not use this for File Manager icons. See “Optional Front Panel Icon Style” on page 60 for more detail.

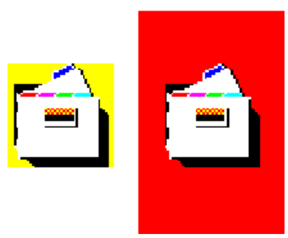


Figure 4-8 Example of dynamic color shadows

Design Philosophy and Helpful Hints

The visual designer must approach the design of icons both individually and globally. Each icon should be individually designed according to the metaphor for that object. Pay careful attention to the visual effect produced by the entire set of icons for an application; they should work together as a family of icons.

Icon design is an iterative process. It is useful to save as many of the iterations as possible whether they are done on paper or computer. When icons are tested with users, it helps to have a varied set of choices to work with.

The philosophy behind the graphic language of the desktop is that users benefit if the computer world more closely resembles the real world. This extends from the three-dimensional appearance of windows and controls, such as push buttons and menu bars, to the general appearance of icons.

Your application icon can range from a logo to an object like the paint bucket in Figure 4-12 on page 52. An icon that looks “real” looks more like something that can be dragged and dropped and manipulated in other ways.

Designing with Color

Your icon should primarily use the eight static grays with the eight static colors used mostly as accents. The eight static colors are very strong and can easily be overused. Using mostly the static grays allows icons to blend gracefully with the already colorful desktop environment. The static colors can be dithered with the static grays to tone the colors down for coverage of larger areas. The grays can also be used to smooth the edges of icons, this is sometimes referred to as “anti-aliasing”.

It is recommended not to use the dynamic colors in File Manager icons, because the appearance of the icon will change when the user changes color palettes. Such a change could be inappropriate as well as unpredictably ugly.

Icon Styles

Icons run the gamut of graphical styles. From the earliest GUI days, the favorite has been a simple black outline style. As color has been added, the style has been that of coloring books, adding color within the black lines. This is a natural drawing style, especially when done on white backgrounds. Many icons are pictographic, while others are abstract.

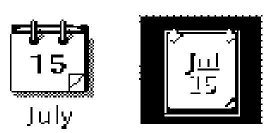


Figure 4-9 Calendar icon in black and white outline style on left and the desktop gray style on right

The desktop, with its pervasive use of colored and medium-value backgrounds, uses both lighter and darker shades to create fairly realistic images. You are encouraged to explore this rendered style.

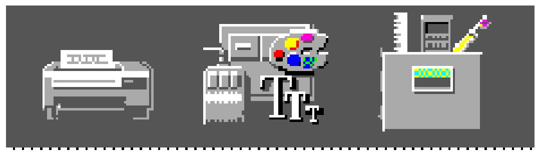


Figure 4-10 Examples of three-dimensional icons in the desktop

Another element of style is the point of view taken in portraying the object. The Common Desktop Environment uses a head on view, as can be seen in Figure 4-10, usually from slightly above if the object in question is a three-dimensional one, such as a printer. It is best to use a treatment that gives the icon a slight dimensional quality, as this reinforces the perception that the icon can be dragged and dropped.

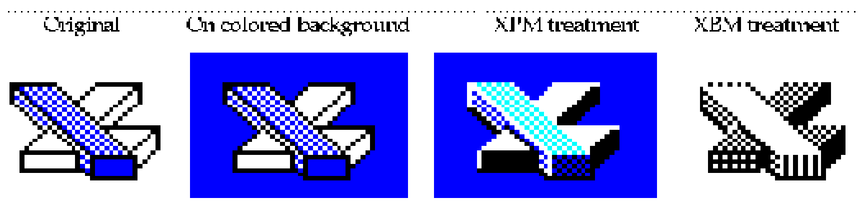


Figure 4-11 Outline style converted to Common Desktop Environment style, in XPM and XBM formats

Designing Your Application Icon

The application icon is the most important icon for you to design. This is the place for your product identity, as well as a clear indication to the user of what your application does. The application icon is what the user opens to run your application.

There are no shape conventions for application icons. They can fill the entire icon bounding box or they can be irregular in shape. It is recommended that your icon have a three-dimensional style to it. The icons shown in Figure 4-12 on page 52 are application icons used in the desktop that you can also use as templates when designing your own icons.

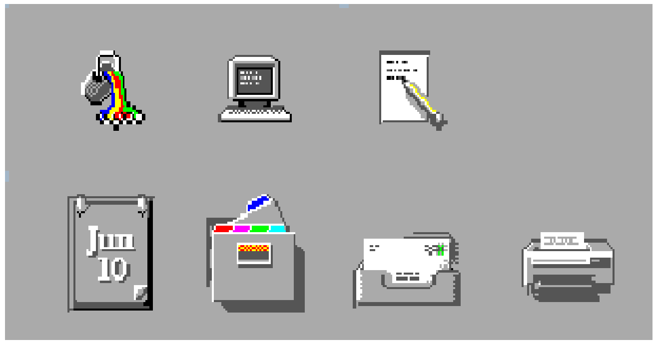


Figure 4-12 Examples of application icons used in the desktop

Designing Your Application Group Icon

The application group icon represents the container in which users find your application, as well as any other files you may choose to include, such as ReadMe or sample files. Design the icon in such a way that users know it is a container, such as a folder or box.

Figure 4-13 Examples of application group icons

The concept used in Application Manager is that of an icon based on accordion-style folders, as seen in Figure 4-13. This icon is large enough that images can be stamped on the front of it to indicate to the user what kind of things can be found inside.

Designing Your Document Icons

A document icon should help the user understand what kind of data is stored in that document icon and what application is associated with the document. Figure 4-14 shows a number of document icons used in the desktop that you can use as templates when designing your own document icons.

Figure 4-14 Examples of document icons used in the desktop

Applications that support multiple file formats need different document icons for the different output formats. Rather than creating a distinctively different graphic for each format, you might use the same graphic for the basic file and add a "tag" to delineate the format.

In the case of the document icon, the basic rectangle of the document is left-aligned in the icon square. If the tag approach is used, place the tag on the right side of the icon, half on and half off the basic icon, but not obliterating the descriptive graphic, as illustrated in Figure 4-14.

International Icons

If your program will be used in more than one country, you should either design your icons for worldwide use or create separate icons for each country.

Worldwide icons are ones that are universally understood. For example, a document icon can be understood around the world because it represents paper, which is used everywhere. Icons for things like a mailbox or trash can are not universal because these objects look different in different countries.

Humor usually does not translate well. Text and symbols are also country-specific and should not be used in icons. Avoid the use of animals or body parts (heads, hands, and so on) because these have varying meanings and may be offensive in some cultures.

Recommended **fa:** Icons should be designed for international use.—***page 226***

Differences with Other Platforms

The desktop is different from the application spaces you may be familiar with in the following ways:

- The desktop requires the larger 48x48 size to accommodate higher resolution displays.
- The desktop has a different color space for icons. You may be able to reuse icons from other environments, but if they have color in them, chances are some of the colors will need to be changed to map onto the desktop palette. The basic design should still work. See Table 4-1 for aid in translating colors.
- Perhaps the most significant difference is that, in most cases, desktop icons appear against a background color other than white, which seems to be the norm in other environments. This can make your icon appear unreadable if you simply copy it from another environment. You should test any icons from other environments before using them on the desktop.

Table 4-1 RGB Values for Common Desktop Environment Icon Colors

Color	RGB Values Decimal	RGB Values Hex	Grays	RGB Values Decimal	RGB Values Hex
Black	0, 0, 0	#00	Gray1	222, 222,222	#de
White	255,255, 255	#ff	Gray2	189, 189,189	#bd
Red	255, 0, 0	#ff0000	Gray3	173, 173,173	#ad
Green	0, 255, 0	#00ff00	Gray4	148, 148,148	#94
Blue	0, 0, 255	#0000ff	Gray5	115, 115,115	#73
Yellow	255, 255, 0	#ffff00	Gray6	99, 99, 99	#63
Cyan	0, 255, 255	#00ffff	Gray7	66, 66, 66	#42
Magenta	255,0,255	#ff00ff	Gray8	33, 33, 33	#33

Implementation of Required Icons

This section discusses the details you need to know to create icons that display correctly in the desktop environment, such as formats, resolutions, sizes, naming, and so on.

Formats

The desktop runs in both color and monochrome modes, so you must create your icons in two formats: XPM for color, and XBM for monochrome. The Icon Editor saves icon files to both formats.

Note – The monochrome icons generated by the Icon Editor usually need some further refinement. For example, when converting the colors and greys to black and white, parts of the icon may disappear altogether or appear too thick.

In the desktop, buttons and palettes can use either the XBM or XPM formats. It is strongly recommended that you use XPM format wherever possible for your button, palette, and tool bar graphics.

The XBM file format has only two colors: foreground and background. In the desktop, the foregound color is not fixed, but varies according to the background color. In one color scheme, the background color might be a dark gray causing any text or graphics to appear in white. However, a color scheme with a light gray background will cause text and graphic to appear in black.

This inverting of the foreground color will have strange effects on certain icons. For something simple, like an arrow shape, there is no adverse consequence. But for other images, the “negative” version created by the inverting of the foreground color might be illegible and, therefore, unusable.

For example, an ice cream cone graphic, with white as the foreground color to create a solid white scoop of ice cream on top of an outlined cone, will look quite different when the ice cream cone becomes a black outline with a black scoop of ice cream. If your application lets users choose the flavor of ice cream, a confusing message will be sent to your user when the color changes.

.

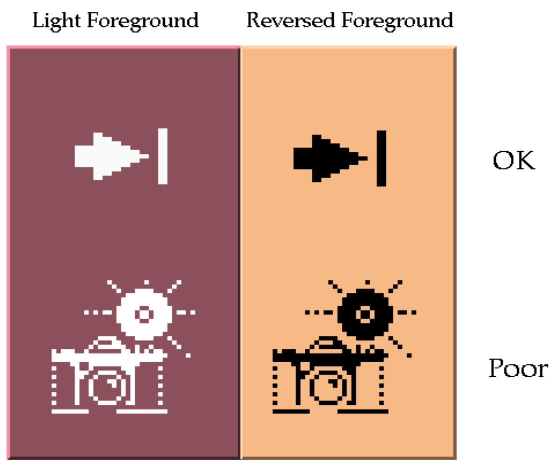


Figure 4-15 Monochrome (XBM) bitmaps, with foreground reversal consequences

Resolutions

The desktop accommodates three display resolutions: low resolution (640x480), medium resolution (800x600), and high resolution (mega-pixel). The size of the Front Panel and some of the icons change automatically depending on the display resolution. For this reason, your application must provide different sized icons.

Recommended **ew:** Any icons or graphics displayed by your application are designed to be distinguishable on low- (640x480), medium- (800x600), and high- (mega-pixel) resolution displays. Alternatively, your application provides different sized visuals for low-, medium-, and high-resolution displays.*—page 226*

Sizes

There are three sizes of the desktop icons: 16x16, 32x32, and 48x48, referred to as 16, 32, and 48. (They have suffixes of `.t`, `.m` and `.l` respectively.) If your application comes from the PC domain, then the size 16 and 32 icons used in the desktop should be familiar sizes. Table 4-2 defines where each size is used.

Table 4-2 Icon Sizes and Usage

Component	**Low Resolution**	**Med. Resolution**	**High Resolution**
File Manager	32, 16	32, 16	32, 16
Application Manager	32, 16	32, 16	32, 16
Front Panel	32	48	48
Subpanels	16	32	32
Front Panel Controls	16	16	16
Minimized Window	32	48	48

Note – 24x24 icons (suffixes of `.s`) are used for internal application graphics like toolbar graphics and are not part of the standard set of desktop icons.

Table 4-3 lists the icons you need to create for an application. A total of 16 icon files are needed, assuming one of each type and size. Figure 4-16 on page 59 shows an example set of icons.

Table 4-3 Minimum Required Icon Set

Type of Icon	**Color 16**	**Color 32**	**Color 48**	**Mono. 16**	**Mono. 32**	**Mono. 48**
Application Icon	✓	✓	✓	✓	✓	✓

Table 4-3 Minimum Required Icon Set

Type of Icon	Color 16	Color 32	Color 48	Mono. 16	Mono. 32	Mono. 48
Document or File Icon	✓	✓		✓	✓	
Application Container Icon	✓	✓		✓	✓	
Minimized Windows			✓			✓

Recommended **ex:** Any icons or graphics displayed by your application are designed to display well on black-and-white and gray-scale monitors. These visuals also display well on low-color (16) systems.—***page 226***

Icon Naming Conventions

The basic name for the icon must be no more than seven characters. The size and color suffixes are appended to the name as shown in Table 4-4.

Table 4-4 Icon Name Conventions

Size	COLOR	B&W	B&W Mask
48	*Iconame*`.l.pm`	*Iconame*`.l.bm`	*Iconame*`.l_m.bm`
32	*Iconame*`.m.pm`	*Iconame*`.m.bm`	*Iconame*`.m_m.bm`
24	*Iconame*`.s.pm`	*Iconame*`.s.bm`	*Iconame*`.s_m.bm`
16	*Iconame*`.t.pm`	*Iconame*`.t.bm`	*Iconame*`.t_m.bm`

The suffix `.pm` is for the color XPM format. The suffix `.bm` is for the XBM format. The suffix `_m` refers to the mask for the black and white icon.

Please note that you do not have to provide icons in all these configurations. Table 4-3 lists the required icons. For example, the `.s` icons are used primarily for things like tool bars, which your application may not have.

Figure 4-16 Example of a minimum required set of icons for Mailer

Alignment

Depending on the graphic you use for your icon, the bits may not take up the entire space allocated for the icon. The recommended rules for where the empty space goes in a desktop icon are shown in Figure 4-17 on page 60. Following these rules ensures your icons visually line up with other icons when used in File Manager or on the Front Panel.

Recommended **fb:** 16x16 and 32x32 icons are left-aligned; any empty bits are on the right side of the bounding box.***—page 226***

Recommended **fc:** 48x48 icons are centered in the bounding box.***—page 226***

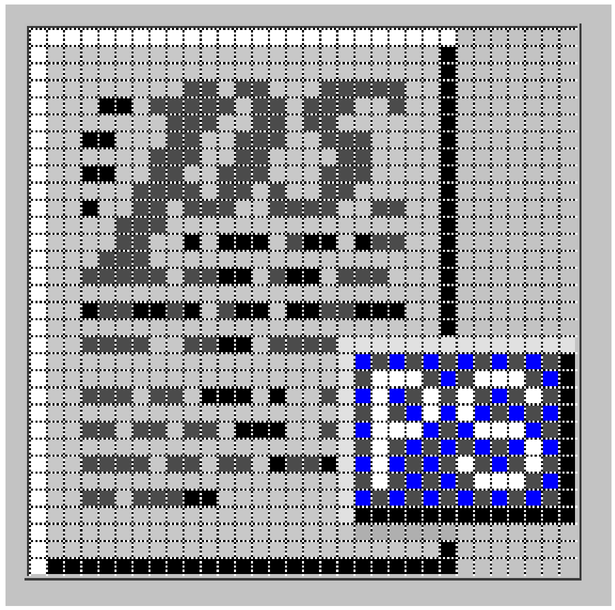


Figure 4-17 Example of a left-aligned 32x32 icon with a tag on the right side

Optional Icons Sizes

There is no 48 requirement for the document or application group icons because neither are expected to be used for a minimized window icon (the tool's icon is used instead) or on the Front Panel. But it is possible that a user might promote one of these icons to the Front Panel.

When a size 48 icon is not available, the Front Panel uses the size 32 icon instead. If you think your icons might be used in the Front Panel, you can supply size 48 icons.

Optional Front Panel Icon Style

The icons that appear by default in the Front Panel have a slightly different appearance from File Manager icons. They appear to be etched into the surface. This gives the icons a more permanent look, because they cannot be dragged and dropped.

You do not have to provide size 48 icons with an etched appearance like the default icons in the desktop. It is not easy to determine if and when your icons will be used in the Front Panel; therefore, it is recommended you not design specifically for this usage, rather design for File Manager usage which will be more common.

If you decide to design Front Panel icons, the following are instructions on developing the etched look. It is strongly recommended that you work with a visual designer to implement this style.

Achieving the Etched Look

You must be familiar with dynamic colors to apply etching. Start with a size 48 icon that does not use the entire 48x48 space. The artwork should leave a few pixels on all sides for the etching.

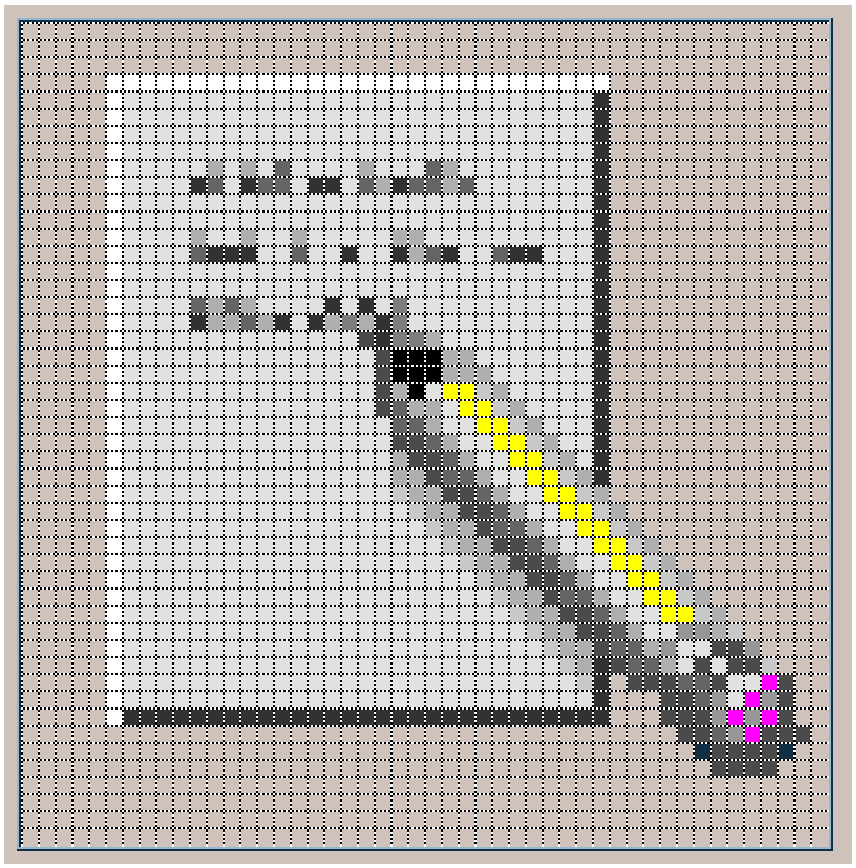


Figure 4-18 Example of an icon lighted from the top left edge

The icon has to be rendered with both light and dark colors, preferably grays. The icon must be lit from above and to the left. The upper and left edges must be light in color, while the lower and right edges must be dark in color. Figure 4-18 shows the desktop Text Editor icon before any etching has been applied.

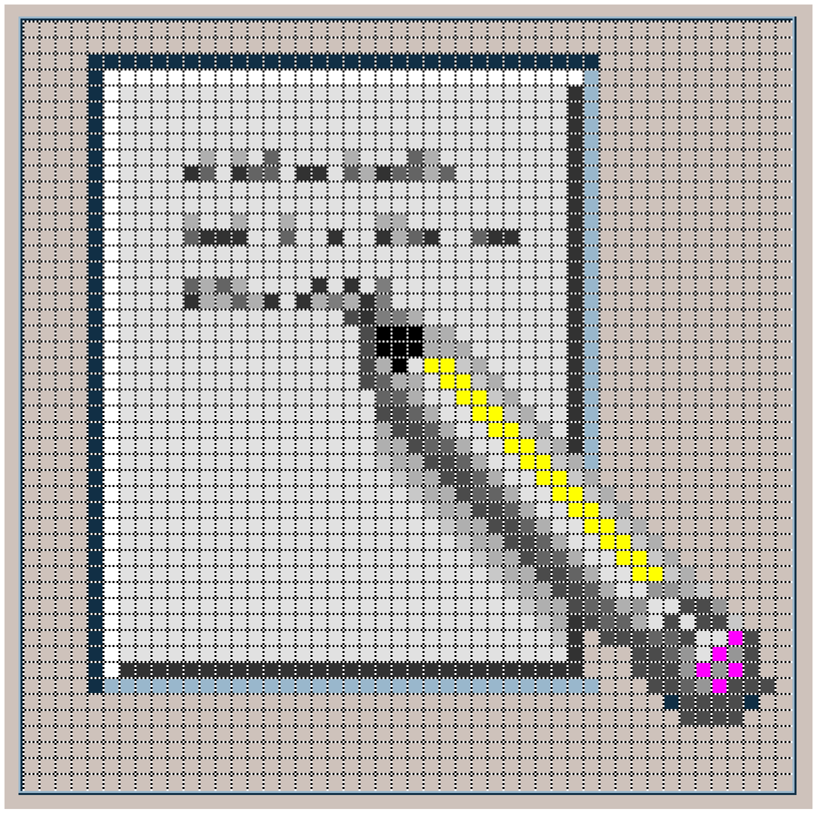


Figure 4-19 Applying the *bottomShadow* and *topShadow* colors

The etched effect is applied by drawing a single-pixel line of the *bottomShadow* color just outside the upper and left edges of the icon artwork, and by drawing a single-pixel line of *topShadow* color just below and to the right of the artwork.

The lighting model of the icon *and* of the etched shadow must be consistent or the effect does not work. If your icon is drawn with black lines, the etched look will be flawed by doubling the dark lines on the top and left edges.

The style of the icon is critical to making the etched effect look right and to making your icon blend in with other Front Panel icons. Study the Front Panel icons supplied with the desktop for guidance. Icons with perspective scenes, icons with black outlines, and icons on raised “slabs” will not work.

Evolving the Etched Look

Etching is a way of making the icon appear to be part of the surface it is etched on. Not all the parts of an icon have to be etched into the surface. You can apply selective etching, making part of the object anchor itself *in* the panel and some of it lie *on* the panel or protrude from it slightly.

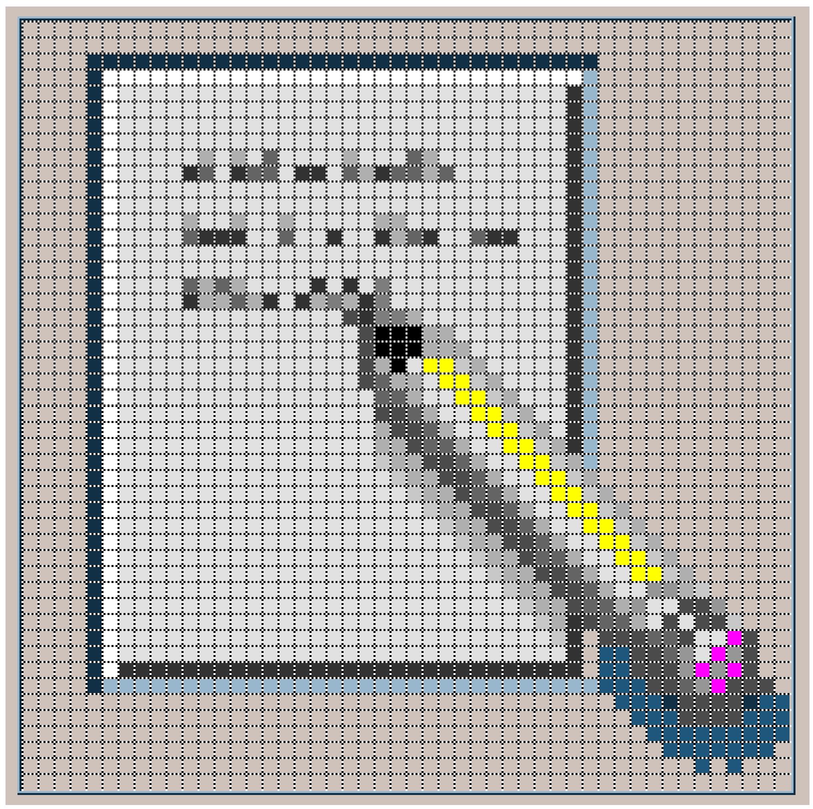


Figure 4-20 Example of anchoring page while letting pencil protrude from surface

The Help icon, for example, takes away the etch, made with the *topShadow* color, under and next to the right-hand book, and replaces it with a *select* color shadow. This makes it appear that one book is protruding slightly from the shelf. The printer icon has a protruding paper tray. In the Style Manager icon, the palette, letters, and mouse are above the etched-in window frame. The File

Manager icon has gone the furthest, as only the edge of the opening is etched, while the drawer front and the cocked folder protrude and even have a shadow.

The principle is to have something in the artwork anchored, yet let the 3-D nature of the objects come out as well. The variable content in an icon, like the printer page or the mail envelopes in the Mailer icon, should not be anchored.

Window and Session Control 5

Your application is presented to the user as a series of windows. Some of these windows present the main portion of the application. Others are dynamic, only appearing to the user when needed to accomplish certain tasks. All of these windows should contain menus, border decorations, and behavior styles appropriate to their function. The following chapter describes the guidelines that should be applied when designing the windows in your application.

Window Control Guidelines

The specifics of the appropriate window borders and decorations are outlined in “Window Decorations” on page 68, and the different window management behaviors are specified in “Window Management Actions” on page 67.

The fundamental user-visible characteristic of primary windows is that stacking, workspace placement, and minimization can be independent of other primary windows. Secondary window stacking, workspace placement, and minimization must be tied to the associated primary window.

Required **aa:** Application windows should be clearly distinguishable as primary or secondary windows based on appearance and behavior.—***page 187***

Window Management Actions

Required **aq:** Windows should follow Common Desktop Environment window management functionality conventions, as shown in Table 10-2.—***page 191***

Primary windows should provide Close, Move, Lower, and Minimize as the minimum set of capabilities. They should allow Resize and Maximize as appropriate. Secondary windows should be designed so that resizing and maximizing are neither necessary nor appropriate. Most secondary windows should only include the Close, Move, and Lower capabilities. In extraordinary cases, a secondary window may provide the Resize and Maximize capabilities. Secondary windows do not provide Minimize capability - they are minimized with the associated primary window.

Required **as:** Windows that have form factor constraints need to set Window Manager hints for minimum size, maximum size, aspect ratio, and resize increment as appropriate.***—page 191***

Required **at:** Maximizing a window should show more content (objects or controls) if appropriate (as opposed to scaling up the sizes of objects and controls).***—page 191***

Required **au:** Windows that have Close or Exit functionality need to support the window management protocol for Close if there is a window menu. In the case of dialog boxes, the Close item on the window menu corresponds to the Cancel functionality or dialog box dismissal with no further action taken.***—page 192***

Window Decorations

Window decorations are the user-visible controls in the frame of an application window. The Figure 5-1 on page 69 shows some sample decorations typically associated with a primary window.

Required **ab:** Windows that support particular window management functionality must request the corresponding window decoration (for example, a window that can be minimized should request the minimize button).***—page 188***

.

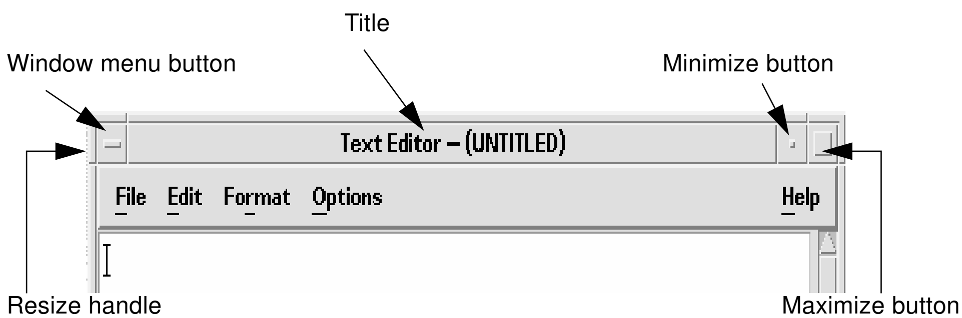


Figure 5-1 Example primary window decorations

In addition, windows that support any window management functionality (move, resize, minimize, maximize, close, and others) must have a window menu with the appropriate items for that functionality.

Required **ad:** Follow Common Desktop Environment window decoration conventions, as shown in Table 10-1.—***page 188***

Primary windows should have the following window decorations: Border, Title, Menu, and Minimize. If appropriate, primary windows should also include Maximize and Resize decorations.

Secondary windows should be designed so that resizing and maximizing are neither necessary nor appropriate. Most secondary windows should only include the Border, Title, and Menu decorations. If your secondary window allows resizing or maximizing, however, it must also include the appropriate decoration. The Figure 5-2 on page 69 shows a typical secondary window decoration.

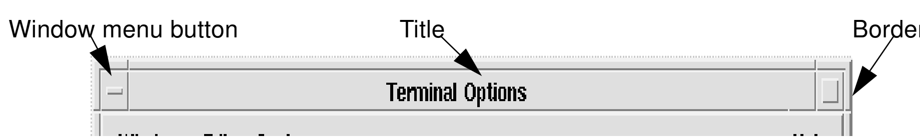


Figure 5-2 Example secondary window decorations

Window Menus

Windows have a menu that allows the user to perform various operations that affect the size and placement of the application window. Developers should use the following standard window menu items in their applications.

Required **ae:** Follow Common Desktop Environment window menu conventions. Items should appear in the window menu if they are applicable to the window or its minimized window icon.*—page 189*

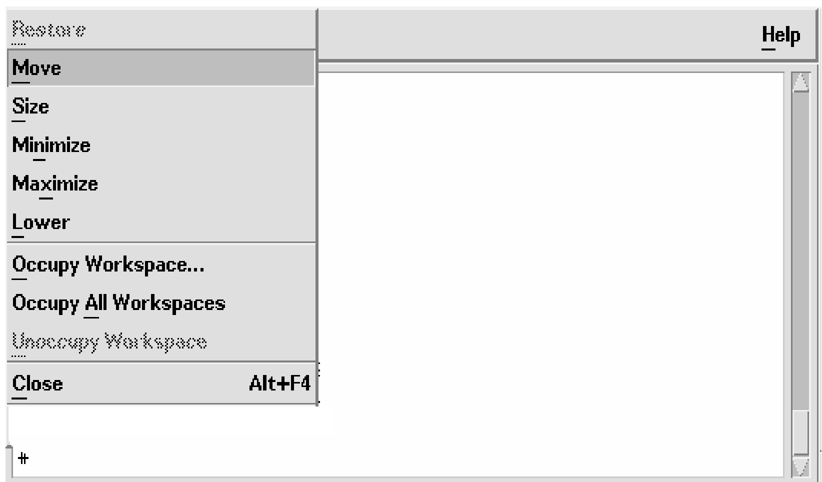


Figure 5-3 Sample window menu

The following items are valid English-language choices in the window menu (the mnemonics for each choice are listed in parentheses). They should be added to the menu in the order listed. Unless otherwise noted, the functionality of these menu items is as described in the *OSF/Motif Style Guide, Revision 1.2*.

- Restore (R)
- Move (M)
- Size (S)
- Minimize (n)

- Maximize (x)
- Lower (L)
- separator
- Occupy Workspace... (O)

  Allows a user to specify which workspaces the application occupies.
- Occupy All Workspaces (A)

  Enables the user to place the application in all available workspaces.
- Unoccupy Workspace (U)

  Removes the application from the current workspace. If the application is only occupying one workspace, the item should be made insensitive.
- separator
- Close (C)

Optional **af:** Applications should not add items to the window menu. If an extraordinary requirement has an application add items to the window menu, the items should be appended to the end of the menu with a separator between Close and the application items.***—page 189***

Optional **ag:** Accelerators, aside from Alt+F4 for Close, should not be used in the window menu (to minimize conflict with other uses of the Alt key for application accelerators, localization, and others).***—page 189***

Window Icons

Applications should use icons to represent themselves to the user when minimized on the desktop.

Optional **ah:** Applications should provide unique window icons for their primary windows. The window icon image should have a similar appearance to the associated file or Front Panel icon image.***—page 189***

Optional **ai:** The window icon label should contain the same text as the title of the corresponding primary window, or an abbreviated form of it. Refer to "Layout" on page 193 for window title guidelines.***—page 190***

Optional **aj:** The window icon image should have a similar appearance to the associated file or Front Panel icon image. Refer to "Design Philosophy and Helpful Hints" on page 49.—***page 190***

Window Placement

Window positioning should be left to the Window Manager or to user control.

Recommended **ak:** Applications should not require or force windows or window icons to be positioned at a particular screen location.—***page 190***

Recommended **al:** A secondary window is placed by the application relative to the associated primary window. It should be placed close to, but not obscuring, the component that caused it to be displayed and the information that is necessary to interact with the dialog box.—***page 190***

Recommended **an:** If a secondary window is allowed to be stacked below its associated primary window (not constrained to stay on top of the primary window), it should be placed such that it is not completely covered by the primary window. This recommendation takes precedence over other placement recommendations.—***page 190***

Recommended **ao:** If a menu or dialog box is already on display, reinvoking the command that caused it to be displayed automatically brings that window or menu to the front of the window stack without changing its position on the screen.—***page 190***

Workspace Management Guidelines

Desktop applications appear in one of several work areas called workspaces. A user may have several workspaces active on the desktop. The application should behave in certain ways in relation to those workspaces.

Recommended **av:** When your application creates a new window, it should come up in the user's current workspace and only occupy that single workspace.—***page 192***

Recommended **aw:** Application windows that are related to a particular task should move together between workspaces.—***page 192***

For example, a spreadsheet application may have one or more secondary windows open that allow the user to change the properties of data cells in the main window. If the user moves the main window to a different workspace, the properties windows should move with it.

On the other hand, a word processor may have several windows open, where each is used to edit a different document. In this case, when a user moves one of the windows to a different workspace, the other windows may remain where they are.

Session Management Support

When you design a desktop application, you must consider the following guidelines for session management.

Required **ax:** Applications should support Interclient Communications Conventions Manual (ICCCM) mechanisms for session management of their primary windows and key properties.—***page 192***

The ICCCM defines important relationships and behaviors between applications and the window manager, including protocols for saving and restoring application state across invocations.

Required **ay:** Applications should support ICCCM mechanisms for session management of all associated windows (that is, secondary windows that may include help windows).—***page 192***

Associated windows include multiple primary windows and secondary windows, such as online help windows.

Optional **az:** Applications should accept messages from the Common Desktop Environment Session Manager that inform them the user is logging out and should save their state at that time.—***page 192***

Optional **ba:** Applications that have a single primary window that is open at the time the user logs out should restore the primary window, in the workspace last occupied, when the user logs in again.—***page 192***

Optional **bb:** Save user context wherever possible. For example, applications that support the editing of files should save the state of the file at logout and should restore the file in the application window when users log in again.—***page 192***

Optional **bc:** Applications that have multiple primary windows that are open at the time the user logs out should restore all primary windows, in their respective workspaces, when the user logs in again.—***page 192***

Application Design Principles 6

You application should present its components to the user in a logical and task-organized manner. Menus should follow a common organization and naming convention to enable users to use the same rules and practices across the desktop. The following sections outline the Common Desktop Environment application design and menu structure requirements.

Component Layout Guidelines

When you design the physical organization of the controls within your application, you should use the following guidelines to ensure that users are presented with a consistent interface throughout the desktop.

Main Window Layout

Required **6-1:** Your application should be composed of at least one main window.—***page 193***

A main window contains a client area and, optionally, a menu bar, a command area, a message area, and scroll bars. The client area contains the framework of the application. The use of a main window ensures interapplication consistency.

Required **bd:** The default size of the application's main window must be large enough to accommodate a typical amount of data, but should not fill the entire physical display size to minimize visual conflicts with other applications.—***page 193***

Required **6-2:** If your application has multiple main windows that serve the same primary function, each window closes and iconifies separately.***—page 193***

For example, a text editor might allow the user to edit multiple documents, each in its own main window. Each window is then treated as a separate application and can be closed or iconified when it is not being used.

Required **6-3:** If your application has multiple main windows that serve different primary functions, each window should be able to iconify independently of the other windows.***—page 194***

For instance, a debugger might provide separate main windows for editing source code, examining data values, and viewing results. Each window can be iconified when it is not being used, but it is up to the application to decide whether each window closes separately or whether closing one window closes the entire application.

Required **be:** Resize corners should be included in any main window that incorporates a scrolling data pane or list.***—page 193***

Any changes to the overall size of the window should result in a corresponding increase or decrease in the size of the scrollable portion. Additionally, your application might reorganize elements within the window based on the increased or decreased amount of space (for example, it might reorganize a row of buttons into two rows).

Menu Bar Layout

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Required **6-4:** If your application has a menu bar, it is a horizontal bar at the top edge of the application, just below the title area of the window frame. A menu bar organizes the most common features of an application. It contains a list of menu topics in cascading buttons; each button is associated with a distinct pull-down menu containing commands that are grouped by common functionality. The use of a menu bar yields consistency across applications.***—page 195***

A menu bar organizes the most common features of an application. It contains a list of menu topics in buttons; each button is associated with a distinct pull-down menu containing commands that are grouped by common functionality. The use of a menu bar is not required, but it is strongly recommended.

Required **6-5:** The menu bar for your application contains only cascading buttons.***—page 195***

The use of other types of buttons in the bar precludes user browsing of the menu structure.

Recommended **bn:** There are several common menu operations that should be considered "standard". The standard menu bar entries are File, Edit, View, Options and Help. If your application provides that functionality to the user, it should be included in the menu bar under the appropriate name.The contents of these menu entries are discussed below in more detail.***—page 196***

Standard menu bar entries should be presented in the following order:

File Edit View Options Help

You should exclude from your menu bar any item shown in the preceding text if your application does not support the associated function. For example, if your application does not support the ability to display its data in different views, then you should not include a View menu.

You may add application-specific menus in between any of the standard menu items, with the following exceptions:

- The File menu, if present, is located in the first menu position on the left.

- The Help menu is located on the far right position.

- If File and Edit are present, they should be next to each other.

For example, your application may have:

File Edit <category1> <category2> View Options <category3> Help

Recommended **bo:** Applications that are not file-oriented in nature (or that manage files transparently, not exposing this activity to the user) should replace the File menu with one or more application-specific menus.—***page 196***

Possible replacements for the File menu:

Replacement1: <app-label> Selected

Replacement2: <app-label><obj-type>

Replacement3: <obj-type>

You may use Replacement1 if your application has more than one object type. Items on <app-label> would be used for global actions that are not specific to an object type. The items in Selected are actions that pertain to objects that are currently selected, and may change depending on what objects are selected. If nothing is selected, this menu should have a single item that says (none selected). If an item is selected, but there are no items that apply to that object, this menu should have a single item that says (none).

You may use Replacement2 if your application has a single object type. Actions that are global to the application are on <app-label>, and actions that are specific to the object type are on <obj-type>.

You may use Replacement3 if your application has a single object type, and does not require an <app-label> menu. For example, a Print Manager might contain a Printer menu.

All other menubar guidelines that apply to File-oriented applications also apply to non-File-oriented applications. Thus, the following menubar would be valid:

<app-label> Selected Edit <category1> View <category2> Help

Applications that are complex or are extremely domain-specific (for example, an application for medical imaging and diagnosis of cat scan data) may require other approaches to their menu bar design. For example,

<app-label><category1><category2> Selected Edit <object-type> Options Help

Recommended **bp:** Exit or Close should be located on the first (leftmost) menu of your menubar.—*page 197*

Common Menu Types

User actions fall into categories that are similar across a wide range of applications. Your application should use the following standard menus when possible to enable the user to easily locate desired functionality.

File Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You need to make the appropriate changes for other locales.

Required **6-7:** If your application uses a File menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application.—*page 198*.

Items should be presented to the user in the order listed below. In all cases where a dialog is recommended to be displayed to the user, and the dialog has functionality outlined in Chapter 7, "Common Dialogs", your application should use a dialog box.

- **New** [REQUIRED]
  Creates a new file. If the current client area will be used to display the new file, your application clears the existing data from the client area. If changes made to the current file will be lost, your application displays a dialog, asking the user about saving changes. The mnemonic is N.
- **Open...** [REQUIRED]
  Opens an existing file by prompting the user for a file name with a dialog box. If changes made to the current file will be lost, your application displays a dialog asking the user about saving changes. The mnemonic is O.
- **Save** [REQUIRED]
  Saves the currently opened file without removing the existing contents of the client area. If the file has no name, your application displays a dialog prompting the user to enter a file name. The mnemonic is S.

- **Save As...** [REQUIRED]
  Saves the currently opened file under a new name by prompting the user for a file name with a dialog box. If the user tries to save the file using an existing name, your application displays a dialog that warns the user about a possible loss of data. Does not remove the existing contents of the client area. The mnemonic is A.
- **Print** [RECOMMENDED]
  Schedules a file for printing. If your application needs specific information in order to print, it displays a dialog, requesting the information from the user. In this case, the menu entry is followed by an ellipsis (Print...). The mnemonic is P.
- **Close** [RECOMMENDED]
  Closes the current primary window and its associated secondary windows. This action does not terminate the application - Exit should be used for that purpose. If changes made to the current primary window will be lost, your application displays a dialog, asking the user about saving those changes. If your application uses only a single primary window or multiple dependent primary windows, this action is not supplied. The mnemonic is C.
- **Exit** [REQUIRED]
  Ends the current application and all windows associated with it. If changes made to the current file will be lost, your application displays a dialog, asking the user about saving changes. The mnemonic is x.

Required **bq:** If the user chooses Exit, or in any other manner indicates that the application should be terminated, but there are changes to the current file that have not been saved, your application displays a dialog box asking whether the changes should be saved before exiting.—*page 198*

The user must always be given the opportunity to explicitly state whether unsaved changes should be saved or discarded. A dialog box similar to the one described should also be displayed if the user chooses Open from the File menu, but has not saved changes to the current file.

<Object-type> and Selected Menu Contents

The <object-type> menu contains controls that allow the user to create instances of the object-type. Both the <object-type> and Selected menus allow the user to manipulate object instances. Additional items should be added to the <object-type> or Selected menus if they relate solely to the manipulation of objects managed by the application (as opposed to more generic services that the application might provide).

Recommended **br:** If your application uses an <object-type> menu or a Selected menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application. Items should be presented to the user in the order listed below.***—page 199***

- **New**... [RECOMMENDED]
  Creates a new instance of the object-type. A dialog box is presented allowing the user to specify the values for settings associated with that object. The mnemonic is N.
- **Move To**... [OPTIONAL]
  Allows the user to move the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder. The mnemonic is M.
- **Copy To...** [OPTIONAL]
  Allows the user to copy the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder. The mnemonic is C.
- **Put in Workspace** [OPTIONAL]
  Allows the user to put a link for the object onto the Common Desktop Environment in the current workspace. The mnemonic is t

Any of the preceding three menu choices should be provided only if the objects managed by your application are able to reside as separate entities outside of your application's main window. For example, a printer object created by a printer management application might be able to be placed in a Folder window and function as an application unto itself. Your application should also support drag and drop as a method for performing any of these actions.

- **Delete** [OPTIONAL]
  Removes the selected objects. A confirmation dialog box should be presented to the user before the object is actually deleted. The mnemonic is D.
- **Properties** [RECOMMENDED]
  Displays a Properties window that shows the current values for settings associated with the selected object. The mnemonic is P.

- **<Default Action>** [RECOMMENDED]
  This choice should enact the default action for the selected object. "Open" is a typical default.

Edit Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Required **6-8:** If your application uses an Edit menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application:*—page 201*

- **Undo** [RECOMMENDED]
  Reverses the most recently executed action. The mnemonic is U.
- **Cut** [RECOMMENDED]
  Removes the selected portion of data from the client area and puts it on the clipboard. The mnemonic is t.
- **Copy** [RECOMMENDED]
  Copies the selected portion of data from the client area and puts it on the clipboard. The mnemonic is C.
- **Copy Link** [OPTIONAL]
  Copies a link of the selected portion of data from the client area and puts it on the clipboard. The mnemonic is K.
- **Paste** [RECOMMENDED]
  Pastes the contents of the clipboard into the client area. The mnemonic is P.
- **Paste Link** [OPTIONAL]
  Pastes a link of the data represented by the contents of the clipboard into the client area. The mnemonic is L.
- **Clear** [RECOMMENDED]
  Removes a selected portion of data from the client area without copying it to the clipboard. The remaining data is not rearranged to fill in the gap left by the Clear operation. The mnemonic is E.

- **Delete** [RECOMMENDED]
  Removes a selected portion of data from the client area without copying it to the clipboard. The mnemonic is D.
- **Select All** [RECOMMENDED]
  Sets the primary selection to be all the selectable elements in the client area. The mnemonic is S.
- **Deselect All** [RECOMMENDED]
  Removes from the primary selection all the selectable elements in the client area. The mnemonic is l.
- **Select Pasted** [OPTIONAL]
  Sets the primary selection to the last element or elements pasted into a component of the client area. The mnemonic is a.
- **Reselect** [OPTIONAL]
  Sets the primary selection to the last selected element or elements in a component of the client area. This action is available only in components that do not support persistent selections and only when the current selection is empty. The mnemonic is R.
- **Promote** [OPTIONAL]
  Promotes to the primary selection the current selection of a component of the client area. This action is available only for components that support persistent selections. The mnemonic is m.

Recommended **bs:** If your application does not provide an <object-type> or Selected menu, but allows the user to select data within the window and manage settings for the selected data, then it provides a Properties ... choice as the last item in the Edit menu.—***page 202***

View Menu

Recommended **bt:** If your application provides a View menu, it only contains functions that affect the way the current data is presented. It does not contain any option that alters the data itself.—***page 202***

Options Menu

Recommended **bu:** If your application has global settings that control the way the application behaves, it provides an Options menu from which these can be set.—***page 202***

Help Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Recommended **bv:** If your application includes a Help menu, it contains the following set of choices, with the specified functionality, when the actions are actually supported by your application. The Help choices included here supercede those listed for Motif 1.2.—***page 203***

This is the Common Desktop Environment-recommended Help menu and should be used instead of the Motif 1.2 Help menu. Items should be presented to the user in the order listed.

- **Overview** [REQUIRED]
  Provides general information about the window from which help was accessed or about the application overall. The mnemonic is **v**. Place a separator after.
- **Index** [OPTIONAL]
  Provides an index listing topics for all help information available for your application.The mnemonic is I.
- **Table of Contents** [RECOMMENDED]
  Provides a table of contents listing topics for all help information available for your application.The mnemonic is C.
- **Tasks** [RECOMMENDED]
  Provides access to help information indicating how to perform different tasks using your application.The mnemonic is T.
- **Reference** [RECOMMENDED]
  Provides access to reference information. The mnemonic is R.
- **Tutorial** [OPTIONAL]
  Provides access to your application's tutorial.The mnemonic is l.
- **Keyboard** [OPTIONAL]
  Provides information about your application's use of function keys, mnemonics, and keyboard accelerators. Also provides information on general Common Desktop Environment use of such keys.The mnemonic is K.

- **Mouse** [OPTIONAL]
  Provides information about using a mouse with your application.The mnemonic is M.
- **Mouse and Keyboard** [OPTIONAL]
  Provides information about your application's use of function keys, mnemonics, keyboard accelerators and mouse operations. Also provides information on general Common Desktop Environment use of such keys.The mnemonic is M.
- **On Item** [OPTIONAL]
  Initiates context-sensitive help by changing the shape of the pointer to the question mark pointer. When the user moves the pointer to a component and presses BSelect, any available context-sensitive help for the component is presented.The mnemonic is O.
- **Using Help** [REQUIRED]
  Provides information on how to use the Common Desktop Environment Help Facility.The mnemonic is U.
- **About** *applicationname* [REQUIRED]
  Displays a dialog box indicating, minimally, the name and version of your application, and displaying its icon or some other signature graphic for your application.The mnemonic is A.

The Overview, Using Help and About items are required. The Table of Contents, Tasks and Reference items are recommended. You can choose to have separate Mouse and Keyboard topics, or have a single combined Mouse and Keyboard topic. You should not use all three items.

Attachment Menu Contents

See Chapter 3, "Drag and Drop" for information on menu recommendations your application should use if it supports attachments.

Pop-up Menus

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Pop-up menus provide access to frequently used functions and should be used pervasively throughout the Common Desktop Environment environment. A pop-up menu may contain a collection of options that appear in different menus available from the menu bar. For example, it may contain items from both the File and Edit menus.

Recommended **by:** Your application should provide a pop-up menu for any element that is selectable within its data pane.***—page 205***

Recommended **bx:** If your application provides functions that apply to a data pane and not any specific element therein, then a pop-up menu is provided that contains the frequently used data pane functions and is accessible by pressing BMenu when the mouse pointer is over the background of the pane or a nonselectable element within the pane.***—page 205***

Recommended **cb:** The functions accessible from within your application's pop-up menus are also accessible from buttons displayed within the window or menus accessed through the menu bar.***—page 206***

Recommended **ca:** Every pop-up menu in your application has a title that indicates the function the menu performs or the element on which it operates.***—page 206***

Optional **cd:** Choices within your pop-up menus are organized in the following manner:***—page 209***

<choices that manage the object such as Open, Save, and Properties>

----------- separator ----------------

<standard edit menu choices such as Cut, Copy and Paste>

----------- separator ----------------

<other choices>

Optional **6-11:** If your application uses any of the common pop-up menu actions, the actions function according to the following specifications. See item cc: for supplemental guidelines.***—page 206***

- **Properties**
  Displays a properties dialog box that the user can use to set the properties of the component.
- **Undo**
  Reverses the last executed action.

- **Primary Move**
  Moves the contents of the primary selection to the component. This action is available only in editable components.
- **Primary Copy**
  Copies the contents of the primary selection to the component. This action is available only in editable components.
- **Primary Link**
  Places a link to the primary selection in the component. This action is available only in editable components.
- **Cut**
  Cuts elements to the clipboard. If the menu is popped up in a selection, cuts the entire selection to the clipboard.
- **Copy**
  Copies elements to the clipboard. If the menu is popped up in a selection, copies the entire selection to the clipboard.
- **Copy Link**
  Copies a link of elements to the clipboard. If the menu is popped up in a selection, copies a link to the entire selection to the clipboard.
- **Paste**
  Pastes the contents of the clipboard to the component. This action is available only in editable components.
- **Paste Link**
  Pastes a link of the contents of the clipboard to the component. This action is available only in editable components.
- **Clear**
  Removes a selected portion of data from the client area without copying it to the clipboard. If the menu is popped up in a selection, deletes the selection.
- **Delete**
  Removes a selected portion of data from the client area without copying it to the clipboard. If the menu is popped up in a selection, deletes the selection.
- **Select All**
  Sets the primary selection to be all of the elements in the collection with the pop-up menu.

- **Deselect All**
  Deselects the current selection in the collection with the pop-up menu.
- **Select Pasted**
  Sets the primary selection to be the last element or elements pasted into the collection with the pop-up menu.
- **Reselect**
  Sets the primary selection to be the last selected element or elements in the component with the pop-up menu. This action is available only in components that do not support persistent selections and only when the current selection is empty.
- **Promote**
  Promotes the current selection to the primary selection. It is available only in components that support persistent selections.

Recommended **cc:** Pop-up menus for selectable objects contain the following set of choices, with the specified functionality, when the actions are actually supported by your application. These guidelines supplement item 6-11.—*page 208*

- **Move To...**
  Allows the user to move the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
- **Copy To...**
  Allows the user to copy the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
- **Put in Workspace**
  Allows the user to put a link for the selected objects onto the Common Desktop Environment in the current workspace.
- **Delete**
  Deletes the selected object. A confirmation is displayed to the user before actually removing the object.
- **Help...**
  Displays a help window pertaining to objects of the type selected.

Required **6-12:** When a pop-up menu is popped up in the context of a selection, any action that acts on elements acts on the entire selection.—***page 209***

In the context of a selection, pop-up menu actions affect the entire selection.

General Menu Design Rules

As you design your application-specific menu panes, follow these guidelines to ensure maximum usability and accessibility.

Recommended **ce:** If the selection of a menu item will result in the user being queried for more information, such as through the posting of a file selection dialog, the menu item should be followed by an ellipsis ("..."). This requirement does not apply to menu items that will result in a simple warning or confirmation dialog being displayed.*—page 209*

The use of an ellipsis helps set the user's expectation for the behavior of the interface. When they select an item without an ellipsis, they know that they can expect an immediate result.

Recommended **cf:** Menus accessed from within your application contain at least two menu items.*—page 209*

No menu should contain only one item. If your application provides a menu with only one item, you should consider moving that item into another menu or making it a button within the window. The longer the menu, the more effort is needed for the user to access choices near the bottom. If your menu has a lot of choices, break it up into two or more menus, or group some items into submenus.

Optional **cg:** Submenus accessed from within your application contain at least three menu items.*—page 210*

Submenus may be used to group like items into a single secondary cascading menu where putting the items into the primary cascading menu would make it too long. However, if your submenu contains only two options, you should strongly look at removing the secondary cascading menu and putting the options into the primary cascading menu since it takes more effort for the user to access options located in a submenu.

Optional **ci:** If your application contains a menu that is expected to be accessed frequently, then a tear-off menu option is provided in that menu.*—page 210*

The user should be able to tear-off frequently accessed menus so that these can remain posted on the desktop as the user uses your application.

Required **6-14:** If your application uses a tear-off button in a menu, the tear-off button is the first element in the menu.—***page 211***

Optional **cj:** Provide keyboard accelerators where appropriate.—***page 210***

If specific menu items within a menu are expected to be used frequently, not the menu as a whole, then your application provides keyboard accelerators for these items and displays the keyboard accelerators in the associated menu to the right of the item to which they relate. You should not use accelerators that have already been defined for system functions - refer to Appendix A, *Keyboard Functions*, for a list of pre-defined key assignments.

Recommended **ck:** The labels used for items in the menu bar do not appear as options within the menus themselves.—***page 210***

The names of items in the menu bar serve as titles for the options the menu contains. The name of the menu bar item should provide a term that accurately describes the concept of the category relating all of the menu items and should not be used as the name of any item within the menu itself.

Required **cl:** Any menu choice that is not currently an appropriate selection is dimmed (insensitive).—***page 211***

Dimmed controls cannot be activated by the user and should appear only when the inactive state is short-term (that is, there is something the user can do within the application or the desktop environment to make the control become active). When the control is persistently inactive (because of the current configuration of the application or system, or a particular set of companion software is not currently installed), the control should be removed from the menu rather than be dimmed.

Required **6-15:** All menus are wide enough to accommodate their widest elements.—***page 211***

Tool Bars

Tool bars are a method used to provide quick access to things that are already user-accessible in an application by other methods. For example, an application can provide access to frequently used features from its menus through its tool bar. Some common usages of tool bars are navigation, changing data views, accessing frequently used tools or editors, simplifying the number of steps to complete a common operation, and providing a fast path to frequently used menu items.

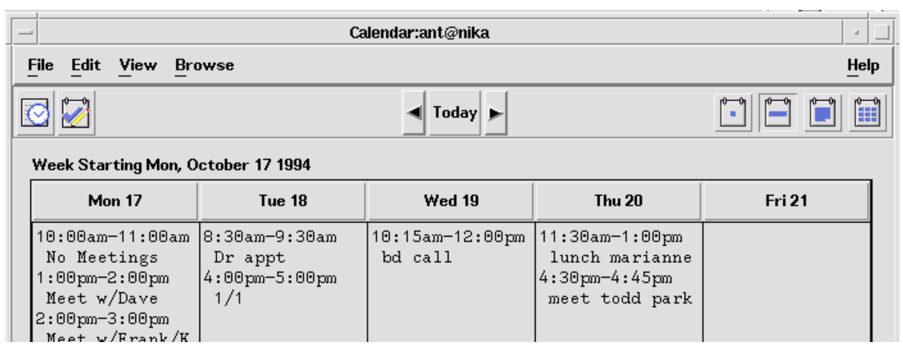


Figure 6-1 Example from Common Desktop Environment Calendar.

Tool Bar Guidelines

Required **fd:** If you use a tool bar, it should be used only in windows with a menu bar.—*page 227*

Required **fe:** Tool bars should contain only operations that are already available to the user in your application menus. All items in a tool bar should be redundant.—*page 227*

Items in a toolbar are meant to provide quick access to operations that are already accessible to the user by other methods.

Required **ff:** When an action represented by a tool bar icon is unavailable to the user, that icon should be made insensitive, with the associated stippled appearance. Whenever a menu item is made insensitive, the corresponding tool bar item must be made insensitive as well.—*page 227*

Recommended **fg:** Give users the option to hide the tool bar.—*page 227*

Design Issues for Tool Bars

When designing your application and an associated tool bar, consider the following issues.

- Would the usability of the application be improved by placing these items on the tool bar?

  Tool bars should only be used when they improve or enhance user access to common operations, such as in an application with several large menus.

- What kinds of operations are being placed on a tool bar? How are they grouped?

  Tool bars should present a natural organization of actions. Grouping items that are dissimilar can confuse users because they do not expect to find the item they are looking for in that context.

- Is the tool bar too crowded?

  Placing too many items in the tool bar can cause the user to have to search for the item they are looking for, rather then being able to quickly find it and use it. Keep the number of buttons to a minimum so that you don't increase the difficulty of your application when using a tool bar.

- Are the icons clearly representative of their associated action?

  Cryptic icons add to user confusion. Keep the pixmaps as simple as possible. Remember that all graphics must be international in scope. When designing a graphic to represent a command, such as Save, remember that the icon has to represent a verb, as opposed to a noun like most other icons. This can make the icon more confusing to users.

Tool Bar Components

A tool bar is typically constructed using the following Motif components.

Tool Bar Container

The tool bar uses a container component to provide a layout mechanism for the drawn buttons that make up a tool bar. You may choose most any container for the tool bar, as long as it allows for the specified behavior.

Required **fh:** The tool bar container is placed directly under the menu bar and should be the same width as the window, as well as similar height to the menu bar.—***page 227***

Recommended **fi:** If you use a tool bar in your application, then you should provide a status line in the same primary window as the tool bar.—***page 227***

This status line should provide immediate feedback to the user as to the purpose of the button that the mouse is currently over or that has the keyboard focus. When the arrow is over a tool bar icon, the status line should display a brief definition of what the icon represents or what will happen when the user clicks the icon.

Recommended **fj:** You may provide labels under tool bar icons. These labels should serve to explain the purpose of the icon.—***page 227***

Tool Bar Button

The Motif DrawnButton provides an appropriate medium for the graphic buttons in tool bars.

Recommended **fk:** Drawn buttons in the tool bar should be the same width and height. Similar or related items should be grouped, and groups should be evenly spaced across the tool bar.—***page 227***

Pixmap

The pixmap for the drawn button is the graphic that conveys the functionality to be expected by pushing a particular button.

Recommended **fl:** All pixmaps in the tool bar should be the same size.—***page 227***

This ensures that all the buttons will be the same size.

Recommended **fm:** The recommended size of the pixmap is 24x24. The default for the drawn button is to resize itself according to the size of its label type, which, in this case, would be a pixmap.—***page 227***.

Window Titles

The following guidelines should be followed when defining titles for your primary and secondary windows.

Optional **bf:** The title of your primary window (the main window your application displays to the user) should be the name of your application.***—page 194***

Note that this does not have to be the actual name of the executable invoked by the user.

Carefully consider how the title you choose for your primary window works when it is used in icons and pop-up windows. If the name of the pop-up window is too long, you may remove the application title, but remember that without the title users might have difficulty telling which pop-up windows belong with the originating primary window.

Optional **bg:** Use initial capital letters for each word in the title (in languages that support capitalization).***—page 194***

Optional **bh:** Follow the application name for each property window, as a minimum, with the title Properties and the name of the object it affects.***—page 194***

Optional **bi:** Begin the title of each pop-up window with the application title followed by a colon, then the title of the pop-up window. The colon should have a space both before and after it for readibility.***—page 194***

Pop-up windows should always indicate which primary window they are associated with (which primary window invoked that pop-up).

Optional **bj:** Use a hyphen to denote the current file name, when the application has files that can be loaded or saved. The hyphen should have a space before and after it. Only the base name of the file should be displayed, not the entire path.***—page 194***

The hyphen is used to denote specific instances of a window or data. The colon serves to delimit general categories or commands. For example, a file manager might have the following title for a Properties dialog box:

File Manager : Properties - myfile

Recommended **bk:** Follow the application name for each command window with the same title that is on the window button or window item users choose to display that window.***—page 195***

Optional **bl:** In the case of multiple primary windows, include the application name at the beginning of each window title, and add a name that uniquely identifies that primary window. No separator should be provided for these names (for example, Calendar Manager Multibrowse, Catalog Search, Admintool Databases).—***page 195***

Optional **bm:** An abbreviated name for the application may be used on other windows, so long as it is done on all windows.—***page 195***

Work-in-Progress Feedback

Recommended **gt:** If any command chosen by the user is expected to take longer than 2 seconds to complete, but less than 10 seconds, your application displays the standard busy pointer as feedback that the command is executing.—***page 233***

The user must receive assurance that your application has "heard" the request and is working on it. If the results of the request cannot be displayed immediately, some feedback must be provided. The busy cursor should be displayed within 0.5 seconds of execution of the command.

Recommended **gu:** If any command chosen by the user is expected to take longer than 10 seconds to complete, your application displays a working dialog box or other feedback of similar character that indicates that the application is working on the request. The feedback should reveal progress toward completion of the activity.—***page 234***

If an activity is expected to take a significant amount of time (10 seconds or more), your application should display feedback stronger than the busy pointer. Displaying the busy pointer for long amounts of time may lead the user to conclude that the application has become "hung." A progress indicator should be displayed in these scenarios that indicates that the application is still functioning and is working on the user's request. The progress indicator should show how much of the activity has been completed and what amount remains.

Recommended **gv:** When your application displays work-in-progress feedback to the user, it does not block access to other applications and services within the desktop environment.—***page 234***

Multitasking should always be supported and, as such, your application should allow the user to access other services while it is busy performing some activity. Preferably, the user is also able to access other features within your application even though it is currently working on another request. When this is supported, your application should display an enhanced busy pointer that indicates that the application is busy but still willing to accept input.

General Application Design Rules

Required **ep:** There is always exactly one control within any window of your application that has the input focus if the window in which it resides has the input focus.*—page 224*

If any window within your application has focus, some control within that window must have focus. The user should not have to explicitly set focus to a control within the window.

Optional **eq:** When a text field within your application does not have the input focus, the text cursor is not displayed within that field.*—page 224*

Although use of inactive text cursors is allowed within the Motif style, it is better to hide the text cursor on focus out rather than display the inactive text cursor. This makes it easier for the user to quickly scan the screen or a window and determine which text field currently has focus.

Optional **er:** Your application provides keyboard mnemonics for all buttons, menus, and menu items displayed within the application.*—page 224*

Once the user becomes adept at using your application, keyboard mnemonics provide the user a quick way to access functionality. Mnemonics also facilitate access to functionality from within keyboard-centric applications or windows. The user need not frequently switch between use of the mouse or use of the keyboard. Mnemonics should be provided pervasively throughout the user interface.

Optional **es:** Your application provides keyboard accelerators for those functions that are expected to be used frequently by the user.*—page 225*

Keyboard accelerators provide the user who has become expert at using your application a quick way to access application functionality without having to go through menus and dialog boxes.

Required **ev:** If your application does not use the values of global environment settings, such as multiclick timeout intervals, drag thresholds, window color settings, mouse left- or right-handedness, and so on, but instead uses its own values for these settings, then your application provides one or more Options dialog boxes that allow the user to change the values for these settings.*—page 225*

In general, you should not override the value of settings treated as global environment settings. These settings are controlled by the user through the Common Desktop Environment Style Manager. If you choose to ignore these settings and specify your own settings, then your application will be have inconsistently with other applications in the Common Desktop Environment. If you nevertheless choose to provide your own values, then you must provide the user a way to make your settings consistent with the rest of the desktop.

Application Installation

Required **em:** Applications should be installed to folders in the Application Manager not directly to the Front Panel or subpanels. For consistency, only Common Desktop Environment components will install to these locations. Users may choose to rearrange their Front Panel, but applications should not do this without user consent.*—page 223*

Common Dialogs 7

Use dialog boxes (secondary windows) to support user tasks that require detailed interaction from the user and that do not lend themselves well to direct manipulation in the main window. For example, you may not require a dialog box to support the task of setting a margin if the task can be performed by directly moving a margin stop on a ruler. On the other hand, you might require a dialog to support formatting a document if the task requires that the user specify several formatting options.

Dialog Box Design and Layout

Optional **ct:** Keep the size of your dialog boxes to a minimum. Remember that on low-resolution displays, dialogs may take up most of the screen real estate, and may even run off the edge of the screen if not designed correctly.**—<Default ¶ <Bold>Font>*page 214***

Optional **cu:** Avoid complexity in your dialog boxes. If your dialog box must support many functions, consider using an expandable dialog box (see "Expandable Windows" on page 228), or use more than one dialog in a nested fashion.**—<Default ¶ <Bold>Font>*page 214***

Optional **cv:** Avoid the use of resize handles in your dialog box. However, you may use resize handles when resizing is useful in allowing users to see more information; for example, when your dialog contains a scrolling list that is likely to be quite long, and users will frequently need to search the list.**—<Default ¶ <Bold>Font>*page 214***

Recommended **cp:** The title of dialog boxes used within your application adheres to the conventions listed in Table 10-3.**—<Default ¶ <Bold>Font>*page 212***

Required **cq:** Every dialog box in your application has at least one button that either performs the dialog box action and dismisses it or dismisses the dialog box without taking any action.**—<Default ¶ <Bold>Font>*page 212***

Recommended **cr:** If your application uses common dialog box actions, the actions have the following specified functionality and labels:**—<Default ¶ <Bold>Font>*page 212***

Label	**Functionality**
Yes	Indicates affirmative response to a question posed in the dialog box.
No	Indicates negative response to a question posed in the dialog box.
OK	Applies any changes made to components in the dialog box and dismisses the dialog box.
<command>	Applies any changes made to the components in the dialog box, performs the action associated with <command>, and optionally dismisses the dialog box.

The <command> button should be used in lieu of OK, Yes, or No as a button label when it provides more meaning to the user as to the action that will be performed when that button is clicked.

Apply	Applies any changes made to components in the dialog box and does not dismiss it.
Retry	Causes the task in progress to be attempted again.
Stop	Ends the task in progress at the next possible break point.
Pause	Causes the task in progress to pause.
Resume	Causes a task that has paused to resume.
Save As Defaults	Saves the current settings as the default settings that will appear the next time the window is displayed. The settings are not applied to any selected object and the dialog box is not dismissed.

A Save As Defaults button should be provided if it is expected that a user would want to use different default values for a set of controls within a dialog box than those that you provide as the factory settings. For example, a Save As Defaults button might be provided in a "New <object type>" window, allowing the user to indicate that whenever a new instance of that object-type is created, the current values should be displayed as the default settings instead of the values given by the application.

Reset — Cancels any changes that have not yet been applied by your application. The controls within the dialog box are reset to their state since the last time the dialog box action was applied. If no changes have been applied within the current invocation of the dialog box, the controls are reset to the state when the dialog box was first displayed.

Reset to Factory — Cancels any changes that have not yet been applied. Components in the dialog box are reset to their default state and value as specified by the vendor that delivered the application (that is, the controls are restored to the original factory settings).

Cancel — Dismisses the dialog box without performing any actions not yet applied.

Help — Provides help for the dialog box.

Recommended **cs:** Any visible control that is not currently active or whose setting is currently invalid is dimmed.**—<Default ¶ <Bold>Font>***page 214*

Dimmed controls cannot be activated by the user and should appear only when the inactive state is short-term (that is, there is something the user can do within the application or the desktop environment to make the control become active). When the control is persistently inactive (because of the current configuration of the application or system, or a particular set of companion software is not currently installed), the control should be removed rather than dimmed.

Optional **cw:** Every dialog box in your application has exactly one default button that is activated when the Return key is pressed.**—<Default ¶ <Bold>Font>***page 214*

The default button should be associated with the most likely response from the user and should not be potentially destructive or irreversible. Some applications may have dialog boxes that do not reveal a default button until a specific set of fields has been filled out or otherwise manipulated.

Optional **cx:** If a dialog box displayed by your application has controls that are considered to be advanced features, use an expandable dialog box, or use a multiple page dialog box that provides a <category> option menu that allows a user to navigate to each page.**—<Default ¶ <Bold>Font>***page 215*

Controls that relate to advanced features should not be displayed with the set of options initially displayed to the user. The typical user should be presented with only those options that are necessary to use the basic functionality of the application. Users looking to access advanced functionality within the dialog box may use the Category option button (see Figure 7-1). If the number of advanced controls is very few, or the settings for these controls are highly related to the settings of basic controls displayed in the dialog box (that is, the settings of the advanced controls change when the user changes settings for basic controls), you might choose to provide an expandable dialog box (see the section on Expandable Windows and Dialog Boxes).

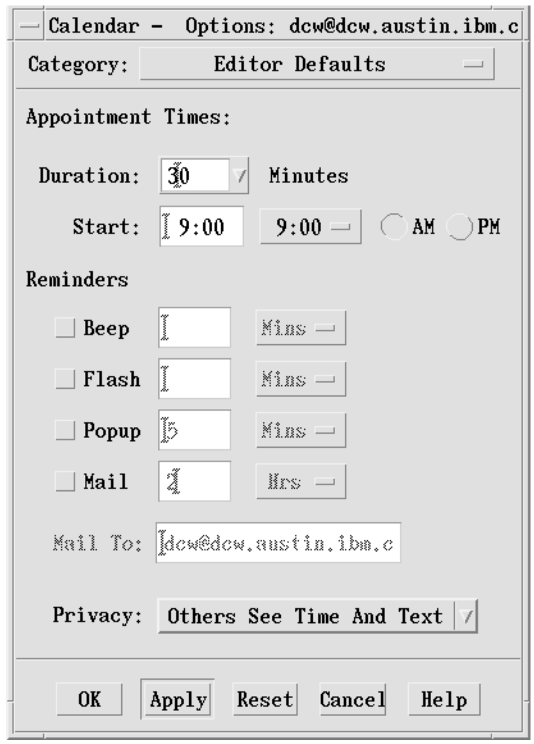


Figure 7-1 An example of using a Category option menu in a dialog

Optional **dl:** Controls within your dialog box are placed in a left-right, top-down layout based on the order in which the user is expected to fill out or choose options within the dialog box.**—<Default ¶ <Bold>Font>***page 218*

This assumes that your application is being designed for a left-to-right language environment. Alternate design approaches may be necessary for other locales.

Required **dm:** Push buttons that affect the dialog box as a whole, either by modifying its contents or layout, invoking the action of the dialog box, or dismissing the dialog box, are located at the bottom of the dialog box.**—<Default ¶ <Bold>Font>***page 219*

In general, there should only be one row of buttons at the bottom of a dialog box. If your application has dialog boxes that contain several global buttons, it may be necessary to create two or more rows of buttons at the bottom of the dialog box. The last row should contain the standard dialog box buttons (OK, Reset, Cancel, and Help). If a dialog box contains buttons that are not related to the dialog box as a whole, but instead, relate to a specific control within the dialog box, the buttons should be located near the control to which they relate.

Required **dn:** If your application provides an Apply button within a dialog box, it also provides an OK button or command button that performs the dialog box action then dismisses it.**—<Default ¶ <Bold>Font>*page 219***

Optional **do:** Your application does not use cascading buttons within dialog boxes unless there is absolutely no other design alternative that can be used without a negative impact on the layout of your dialog box.**—<Default ¶ <Bold>Font>*page 219***

In general, cascade buttons should only be used within menus and menu bars. You should avoid their use in all other locations unless absolutely necessary.

Dialog Box Placement

Recommended **al:** A secondary window is placed by the application relative to the associated primary window. It should be placed close to, but not obscuring, the component that caused it to be displayed and the information that is necessary to interact with the dialog box.**—<Default ¶ <Bold>Font>*page 190***

Some suggestions are given in section 6.2.4.3, "Determining Dialog Box Location and Size," of the OSF/Motif Style Guide, Revision 1.2. Additional or modified recommendations include:***—page 190***

Optional **am:** If a dialog box does not relate to specific items in the underlying window, it should be placed below the menu bar (if there is one) and centered (horizontally) over the work area.***—page 188***

Recommended **an:** If a secondary window is allowed to be stacked below its associated primary window (not constrained to stay on top of the primary window), it should be placed such that it is not completely covered by the primary window. This recommendation takes precedence over other placement recommendations.**—<Default ¶ <Bold>Font>*page 190***

Recommended **ao:** If a menu or dialog box is already on display, reinvoking the command that caused it to be displayed automatically brings that window or menu to the front of the window stack without changing its position on the screen.**—<Default ¶ <Bold>Font>*page 190***

Dialog Box Interaction

All of the navigation and selection guidelines that apply to applications in general should apply to your dialogs. In addition, as you design your application-specific dialog boxes, you should follow these guidelines to ensure maximum usability and accessibility.

Required **en:** When your application displays a dialog box, it places the input focus at the first text field into which the user is allowed to type an entry, or at the first control within the dialog box with which the user should interact.**—<Default ¶ <Bold>Font>*page 224***

Input focus should always be placed at a predictable and intuitive location. The user should not be forced to set focus at the control most likely to be used when the window is displayed.

Recommended **eo:** As the user presses the Tab key within dialog boxes of your application, the input focus moves to different controls within the window in a left-right, top-down order.**—<Default ¶ <Bold>Font>*page 224***

This assumes that your application is being designed for a left-to-right language environment. Alternate design approaches may be necessary for other locales.

Required **et:** Dialog boxes displayed by your application never block input to other applications within the desktop (that is, they are not system modal) unless it is absolutely essential that the user perform no other action in the desktop until the user responds to the dialog box.**—<Default ¶ <Bold>Font>*page 225***

Applications must allow the user the freedom to access information and tools within the user's desktop environment. Only in the most dire circumstances should an application ever block access to other applications and services within the environment.

Required **eu:** Dialog boxes displayed by your application never block access to other functionality within the application (application modal) unless it is essential that the state of the application remains unchanged until the user responds to the dialog box.**—<Default ¶ <Bold>Font>***page 225*

Required **6-18:** A warning dialog box allows the user to cancel the destructive action about which the dialog box is providing a warning.**—<Default ¶ <Bold>Font>***page 223*

Expandable Windows and Dialog Boxes

This section describes a standard method for providing expandable windows or dialogs in Common Desktop Environment. Expandable windows allow users to selectively display advanced or application-specific functionality in a separate portion of the window that is normally not visible when the window is initially displayed. Users can choose to display the entire window, or only the core functionality, according to their own needs and preferences. Applications can implement expandable windows fairly easily using existing toolkit components.

Use expandable windows only when your application needs to present a limited set of additional dialog box options. Consider using an alternate method if your dialog would grow unmanageably large, for example, larger than a typical low-resolution display could handle. Keep in mind also how your dialog will expand when translated into other languages. An alternative method for expandable dialogs is to use a multipage dialog and provide a Category button for switching pages.

Guidelines for Expandable Windows and Dialog Boxes

Recommended **fn:** The primary pane of the dialog box or window should contain all of the controls needed to complete the task. This should include all critical and frequently used functionality.**—<Default ¶ <Bold>Font>***page 228*

Recommended **fo:** It is assumed that infrequently used features are placed in the secondary pane. The core functionality of the application should not depend on any controls placed in secondary panes.**—<Default ¶ <Bold>Font>***page 228*

Required **fp:** Command buttons are aligned along the bottom of the dialog box. When the window is expanded to show a secondary pane, then buttons are moved to the bottom of the secondary pane. See Chapter , "Application Design Principles" for information about layout of action buttons in dialog boxes.**—<Default ¶ <Bold>Font>***page 228*

Recommended **fq:** If important controls must be placed in the secondary pane, the application can specify that the window in question should be displayed in its expanded state by default. Users should still be able to shrink the window by pressing the Contract button.**—<Default ¶ <Bold>Font>***page 228*

Components of Expandable Windows

To create an expandable window or dialog box, use the standard Motif widgets in conjunction with state variables and some simple rules that govern its behavior. In addition to the application-defined controls and displays that make up the content of the window itself, use a primary and secondary pane in the following way.

Primary and Secondary Panes

The primary pane should contain the core or base functionality required by nearly all end-users of the application. The primary pane is a standard Motif container that is the main component of the window or dialog. Only the primary pane is visible when the expandable window is initially displayed. An "expand button" allows the user to display a secondary pane providing access to the full functionality of the window or dialog. [See Figure 7-2]

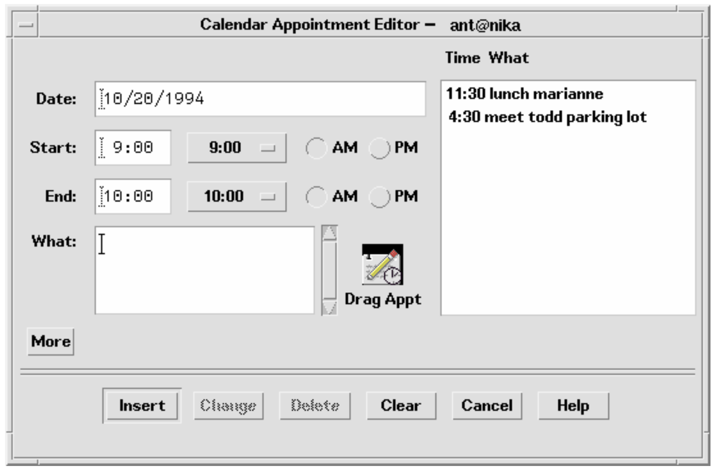


Figure 7-2 Calendar Appointment Editor primary pane with Expand button (More)

Expanding the Secondary Pane

The secondary pane provides space for additional options or advanced functionality without increasing the difficulty of the core functionality in the primary pane. The secondary pane can be expanded in a vertical or a horizontal direction. To determine the appropriate direction in which to expand, consider the following questions:

- What are the reading patterns in the countries that will be using the applications?
- What makes the most sense based on the information in the dialog?

Recommended **fr:** The secondary pane should expand in the direction most consistent with users' expectations, the reading pattern of the language in which it will be displayed, and the content of the information displayed.**—<Default ¶ <Bold>Font>***page 228*

Recommended **fs:** If possible, the panes should have the same default width.**—<Default ¶ <Bold>Font>***page 228*

Required **ft:** A separator should be used to separate the primary pane from the secondary pane.**—<Default ¶ <Bold>Font>***page 228*

The user needs to have clear visual feedback as to which elements are part of the primary pane and which elements are part of the secondary pane of the expandable window.

Resizing the Expanded Window or Dialog

Required **fu:** If a window is resizable, any sizing changes should be allocated to the pane containing scrolling lists or text fields whose displayed length is less than their stored length. If both panes contain scrollable controls, size changes should be distributed evenly between the two panes. If neither pane contains scrollable controls, the window should not be resizable.**—<Default ¶ <Bold>Font>***page 228*

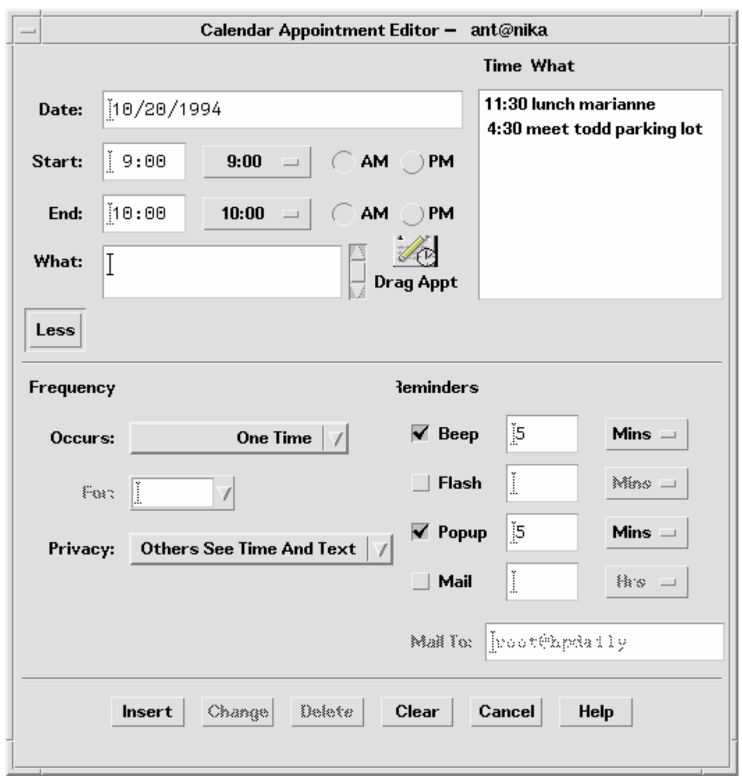


Figure 7-3 Calendar Appointment Editor with expand button (Less) and primary and secondary panes

Expand Button

The expand button is used to display the secondary pane. The expand button is a standard Motif drawn button with a label that changes depending on the state of the window. The button labels for the two states should tell the user what will happen. For example, the button might say Options when the window is closed and Basic when the window is open. Clicking the Options button displays the bottom pane; clicking the Basic button hides the bottom pane. Labels should be opposites such as Expand and Contract, More and Less, or Basic and Options.

Required **fv:** The expandable window should have one button that changes its label based on the state of the window.**—<Default ¶ <Bold>Font>***page 229*

Required **fw:** The expand button should have two labels that reflect the two states of the expandable window accurately. The current label should indicate to the user what will happen if the user clicks the button.**—<Default ¶ <Bold>Font>***page 229*

Examples of possible labels are Basic and Options, Expand and Contract, and More and Less.

Optional **fx:** The expand button may contain a graphic in addition to the label. This graphic should indicate the direction in which the window will expand or contract.**—<Default ¶ <Bold>Font>***page 229*

Placing the Expand Button

Recommended **fy:** The button should appear in the lower left-hand corner of the window or dialog box for expansion in the vertical direction and in the lower-right hand corner for expansion in the horizontal direction.**—<Default ¶ <Bold>Font>***page 229*

Required **fz:** If the window or dialog box contains a scrolling list positioned to the far right side of the pane, do not align the drawn button with the scroll bar. For example, the button should be aligned with the list, not the scroll bar.**—<Default ¶ <Bold>Font>***page 229*

Window State

Required **ga:** Applications must remember the state of each window or dialog box (expanded or not expanded) independently (not collectively). The state should be changed only by the user and should always be preserved until explicitly altered by the user.**—<Default ¶ <Bold>Font>***page 229*

Recommended **gb:** Applications should remember the state of each expandable window or dialog box across sessions, so that users don't have to manually configure the expandable windows each time the application is run.**—<Default ¶ <Bold>Font>***page 229*

If appropriate, applications can provide a mechanism to allow users to set the state of an expandable window on a global basis in the application. This would be part of the application's Options.

File Selection Dialog Boxes

The Common Desktop Environment file selection dialog box is a subclass of the Motif file selection dialog box that has enhancements for improved usability. As long as your application uses the standard Motif file selection dialog box calls, the Common Desktop Environment enhanced version will appear in the Common Desktop Environment environment. There are additional guidelines to follow to make your dialog consistent and easy to use. Use the file selection dialog box whenever your application supports a task that involves choosing a file or directory. Examples are: Open, Include, Save As, and Copy To.

Contents

Required **7-10:** If your application uses a file selection dialog box, it contains the following components:**—<Default ¶ <Bold>Font>***page 237*

- A directory Text component showing the current directory path. The user can edit the directory Text component and press <Return> or <Enter> to change the current directory.
- A group of push buttons, including a command button, and Update, Cancel, and Help buttons. The command button is typically labeled Open or Save, but if there is another label that better describes the

resulting action (such as Include), that label should be used. Activating the command button carries out the corresponding action and dismisses the file selection dialog box.

- For applications that allow saving to different formats, an option button allowing users to specify the format when saving a file.
- A file name Text component for displaying and editing a file name. This component is optional when the file selection dialog box is used to choose an existing file or directory.

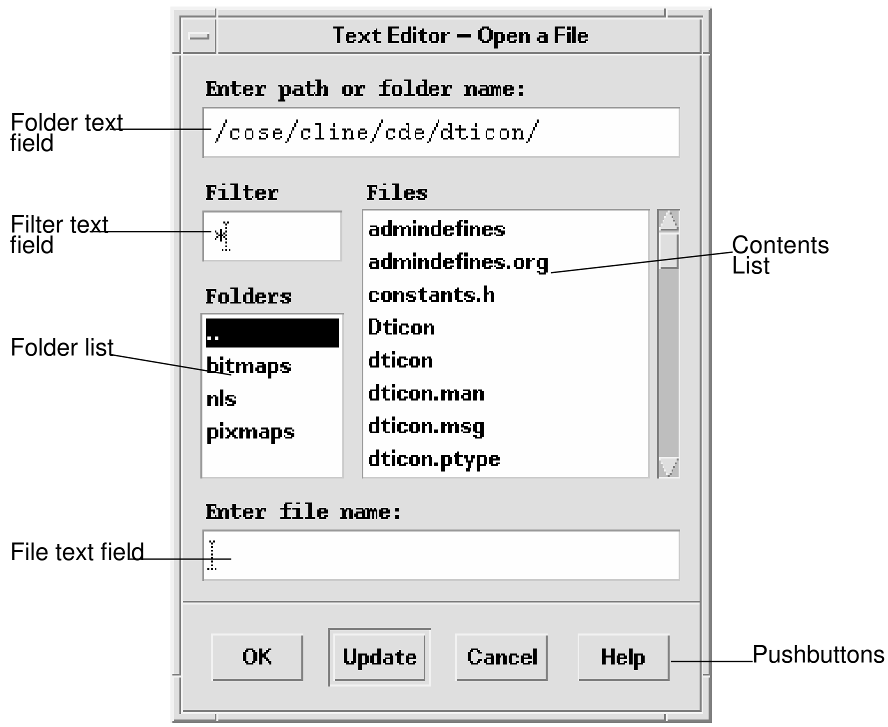


Figure 7-4 Example of an Open file selection dialog box

Required **7-17:** The file selection box displays the contents of a directory in the contents list when the file selection box is initialized, when the user presses Enter or Return in the directory text component, and when the user opens a directory in the contents list. The contents list is updated each time the contents of the directory changes.**—<Default ¶ <Bold>Font>***page 240*

This specification ensures the consistent operation of a directory and file search in a file selection dialog box.

Optional **ht:** Directory and file name lists should be presented alphabetically, case insensitive. The first item on the directory list should be the parent directory and it should be labeled "..".**—<Default ¶ <Bold>Font>***page 241*

Recommended **de:** The file selection dialog box should not display hidden (dot) directories or files, unless your users depend on using these types of files. If your application does support displaying hidden files, you should supply a check box allowing users to toggle between showing and not showing hidden files, or else allow users to toggle between showing and hiding files at a global level in your application.**—<Default ¶ <Bold>Font>***page 217*

Recommended **df:** The file selection dialog box should not show the full path names for files and directories, but should only show the relative names, except for the directory text field**—<Default ¶ <Bold>Font>***page 217*.

The global Common Desktop Environment setting should be:

```
XmFileSelectionBox.fullPathMode: false
```

Unless your application overrides this behavior, your file selection dialog box should not show full path names in the list boxes.

Required **dg:** In general, the file selection dialog box should recall the directory location that was previously set by the user.**—<Default ¶ <Bold>Font>***page 217*

For example, if the user brings up Save As and navigates to `/users/jay/letters` to save the file, the next time the user brings up Save As, the file selection box should be in the directory `/users/jay/letters`. This information, however, should *not* be recalled once the user has closed the primary window, but should resort to the default directory.

If your application supports multiple primary windows, each window should recall the directory location that was set for that window.

File Selection Dialog Box Behavior

Optional **hq:** The file selection dialog box should come up in a directory that makes sense for the task. For example, when saving a new file from an editor, the file selection box should come up in the user’s home directory. If the user navigates to some other directory within the file selection box, the application should remember that directory the next time it is brought up.—**<Default ¶ <Bold>Font>***page 240*

Optional **hr:** Users should never be allowed to overwrite an existing file through the file selection box without a warning dialog box prompt.—**<Default ¶ <Bold>Font>***page 241*

Optional **hs:** Keyboard focus should be placed in the file name field each time users bring up a file selection dialog box.—**<Default ¶ <Bold>Font>***page 241*

Labeling

Optional **hu:** Labels should be clear. In the English language, use the following labels for the file selection dialog box fields and lists:—**<Default ¶ <Bold>Font>***page 241*

Component	Label
Directory text field	Enter Path or Folder Name
Filter text Field	Filter
Directory list	Folder
Contents list	Files
File text field	Enter Filename:*

Optional **hv:** Optionally, application developers can make this label more instructive and specific, such as Enter File to Open for Open dialog boxes.—**<Default ¶ <Bold>Font>***page 241* (see following sections for specific recommendations).

These labels should be the default labels. If they are not set by default, you need to set them via resources in your application’s app-defaults file.

Recommended **he:** When the file selection box is used to specify an existing file (for example, to open a document), the command button is normally labeled Open and it should be the default action.—**<Default ¶ <Bold>Font>***page 237*

Required **hj:** When the file selection dialog box is used to specify a new file name (for example, a Save As dialog box), the command button is normally labeled Save and is the default action. This specification ensures the uniform appearance of a file selection box across applications.—**<Default ¶ <Bold>Font>***page 238*

Button Activation

Recommended **hf:** If the Update button is activated while a directory is selected in the contents list, the directory is opened, its contents are displayed in the contents list, and the directory text is updated.—**<Default ¶ <Bold>Font>***page 238*

Required **hg:** If the Open button is activated while the appropriate file is selected in the contents list, the file is utilized by the application and the file selection box is dismissed.—**<Default ¶ <Bold>Font>***page 238*

Selection and Navigation

Required **7-12:** Double-clicking BSelect on an item in the contents list selects that item and activates the default action. In all cases, double-clicking BSelect on a directory in the contents list opens that directory and displays its contents in the contents list (the default action is Open).—**<Default ¶ <Bold>Font>***page 239*

- When the file selection dialog box is used to choose an existing file, double-clicking BSelect on an appropriate file in the contents List chooses that file and dismisses the file selection dialog box (the default action is Open).
- When the file selection dialog box is used to choose an existing directory or to specify a new directory or file, the files list should not appear.

Required **7-13:** The normal text navigation and editing functions are available in the text components for moving the cursor within each text component and changing the contents of the text.—**<Default ¶ <Bold>Font>***page 239*

These actions provide a convenient way to choose a directory or file name from the corresponding List while focus remains in the Text component.

Optional **7-15:** Your application allows the user to select a file by scrolling through the list of file names and selecting the desired file or by entering the file name directly into the file selection text component. Selecting a file from the list causes that file name to appear in the file selection text area.**—<Default ¶ <Bold>Font>***page 239*

This method for selecting a file needs to be consistent across applications.

Required **7-16:** Your application makes use of the selection when one of the following occurs:**—<Default ¶ <Bold>Font>***page 239*

- The user activates the command push button while an appropriate item is selected in the contents List.
- The user double-clicks BSelect on an appropriate file in the contents list.
- The user presses Return or Enter while the file name text component has the keyboard focus and contains an appropriate item.

Guidelines for Specific File Selection Dialog Box Uses

The following guidelines apply for specific uses of the file selection dialog box, and should be observed in addition to the more general guidelines.

Open Dialog

Recommended **hm:** If the user has opened the application without supplying a file name argument, the Open dialog box should use the user's home directory as the default directory.**—<Default ¶ <Bold>Font>***page 240*

An exception to this rule might be made if a clearly more useful directory can be identified; for example, the icon editor might default to `$HOME/.dt/icons`. For applications that allow editing, never default to a directory in which the user does not have read and write permission, such as `/usr/dt/bin`.

Required **hn:** If the user has opened the application with a file name argument, the Open dialog box should default to the directory in which that file resides.**—<Default ¶ <Bold>Font>*page 240***

Save As Dialog

Optional **ho:** When using the file selection dialog box in Save As capacity, provide a default name of Untitled, place the location cursor in the file name field and highlight the file name text to create a "delete pending type-in" mode. If the current directory already has a file of that name, create a name Untitled2, and so forth.**—<Default ¶ <Bold>Font>*page 240***

Optional **hp:** When using the file selection dialog box in a Save As capacity, add a file name extension if the application supports file typing by extension, and make this extension visible in the file name field. Do not highlight the extension to create a "delete pending type-in" mode, but allow users to modify the extension or delete it explicitly.**—<Default ¶ <Bold>Font>*page 240***

After saving a file using "Save As", the application should use that newly saved file as the current file, and all subsequent edits and saves should apply to that newly created file.

The Save As dialog should use the same directory in which the current file resides.

Directory Selection Dialog

Recommended **hh:** When the file selection dialog box is used to choose an existing directory (for example, to install a set of files into the chosen directory) or to specify a new directory, the command button should be given an appropriate label, such as Install, Choose, Create, or OK. If this button is activated while the appropriate directory is selected in the contents list, the directory is utilized by the application and the file selection box is dismissed.**—<Default ¶ <Bold>Font>*page 238***

Required **hi:** When the file selection dialog box is used to choose an existing directory, there must also be an additional button, labeled Update, that is enabled whenever a directory is selected in the contents list, and opens the directory. This Update button is the default action.**—<Default ¶ <Bold>Font>*page 238***

Optional **hk:** When the file selection dialog box is used to choose an existing file, files are shown in the contents list but they are all disabled. Double-clicking BSelect on a disabled file name has no effect.**—<Default ¶ <Bold>Font>***page 238*

Print Dialog Box

These are guidelines for a common look and feel print dialog box, which may be used wherever a print action is available. The print dialog is not a widget. Developers are encouraged to use these guidelines as a starting point and to add functionality as appropriate for their applications. It is important, however, to remember that users expect consistency from one print dialog to another; therefore, the common area should be left as unchanged as possible. Use a print dialog box whenever users would want to select options for printing a file, a selection, or other type of object. If your application supports printing, you should use a print dialog box, and you may provide an optional nondialog method of printing directly, that is, “silent” printing.

Standard Print Menu Items for Applications

Print...: brings up a print dialog so the user can choose from the available options before printing the selected objects.

Print One: prints one copy of the selected objects, using the default print methods previously defined by the user. The user is not prompted for further information through a dialog.

Guidelines for Common Print Dialog Functions

Applications are expected to provide many different types of printing functions and capabilities. This section provides guidelines for the most commonly used types of print options so that appearance and behavior for these items is consistent across applications. These common items are grouped into a common area that is located in the top portion of the print dialog. Figure 7-5 on page 119 shows a typical print dialog. The common area is the area above the separator line.

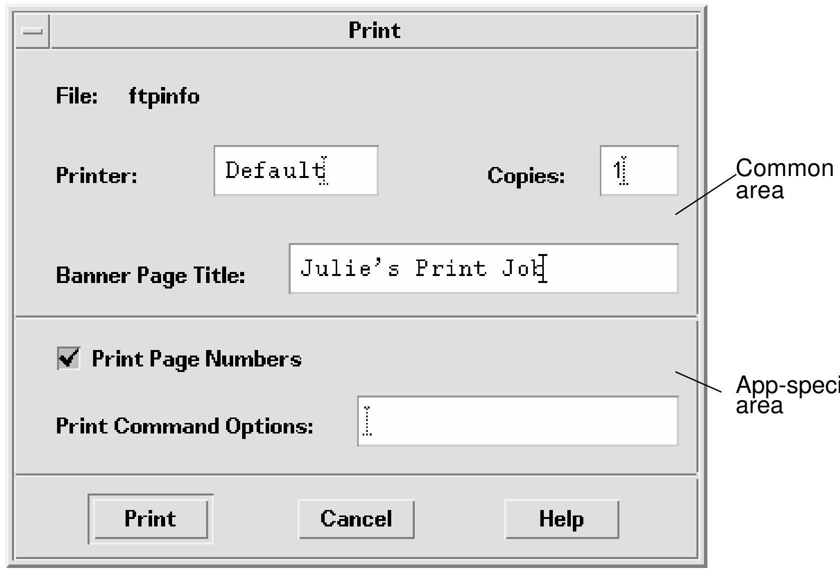


*Figure 7-5*A basic print dialog. box

The common area contains the following components:

- Dialog Title: Print
- File: this is a noneditable field. It displays the file name (if available). If the user is printing a nonfile object, this field should display the object type if possible (for example, mail message, calendar appointment).
- Printer: A combination box; could also be a text field. It contains the name of the printer destination. The default entry is labeled Default, that is, whatever printer is the default destination. The user may select or type any other valid printer name. If it is a combination box, the list of printers could reflect what is appropriate for that printing job. The dialog should retain the last user entry or selection made.

- Copies: A spin box (numeric widget) where the user selects or types the number of output copies desired. Optionally, this could be a text field.
- Banner Page Title: A text field where the user may enter the text the user wants to appear on the banner page (that is, cover page) of their output. This field should pick up the default banner title if the user has set it elsewhere. Optionally, you could add a check box to turn the banner page off completely.
- Separator lines: Used between the common fields, the application-specific fields, and the buttons.

The print dialog contains the following standard buttons:

- Print: Accepts the user's choices in the dialog, prints the selected objects, and exits the dialog.
- Cancel: Ignores the user's choices in the dialog, prints nothing, and exits the dialog.
- Help: Brings up an associated Help window.
- Optional buttons could include Reset, Print Preview.

Guidelines for Application-Specific Print Dialog Box Functions

The standard print dialog application-specific area is the bottom half of the dialog box illustrated in Figure 7-6 on page 122.

Depending upon the application or function, developers may choose to add more fields to the common Print dialog. The controls in the dialog are laid out horizontally; if more fields are needed, it is suggested that you add another separator line, then place the additional controls below it, as illustrated. If any additional push buttons are needed, they should go between the Print and Cancel buttons.

Optional Fields

Some possible optional fields include:

Print Page Numbers
This checkbox applies only when ASCII files are being printed. If the objects selected for printing are non-ASCII, this control should be dimmed. When turned on, output pages will be numbered.

Print Command Options
A text field where the user may type an `lp` command or script name to override the instructions in the other fields. If you want to provide print methods besides `lp`, rename this field (Print Method, Use Print Command, and so on.).

Priority This might be an option menu containing values for High, Medium, and Low, or might be a spin box containing numbers.

Orientation An option menu containing the values Portrait and Landscape (see Figure 7-6 on page 122).

Resolution An option menu or a spin box containing numeric values, in dpi (see Figure 7-6 on page 122).

Sides An option menu containing the values Single and Double (see Figure 7-6 on page 122).

Paper Size An option menu containing the values Letter, Legal, and so on. (Figure 7-6 on page 122).

Paper Source — An option menu containing the values Upper Tray, Lower Tray, and so on. (see Figure 7-6 on page 122).

If the dialog box is to be used for an application, consider:

Page Range — Two text fields, from x to y (see Figure 7-7 on page 123).

Reduce/Enlarge — A spin box containing values for percentages (see Figure 7-7 on page 123).

Print Preview — A button that brings up a WYSIWYG representation of the output.

Some Sample Layouts

Figure 7-6 on page 122 and Figure 7-7 on page 123 show some example Print dialog box layouts.

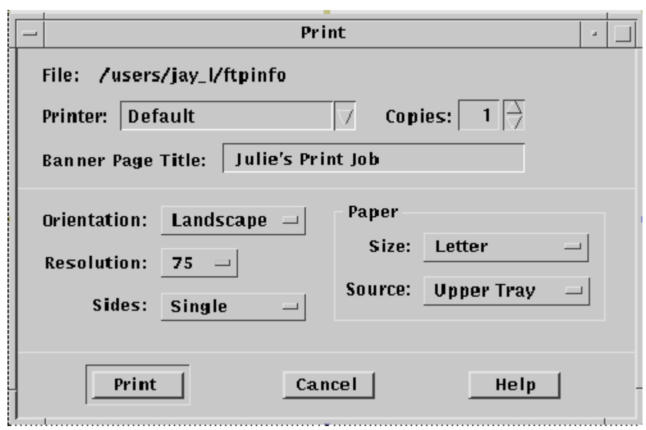


Figure 7-6 Example print dialog box layout for general printing

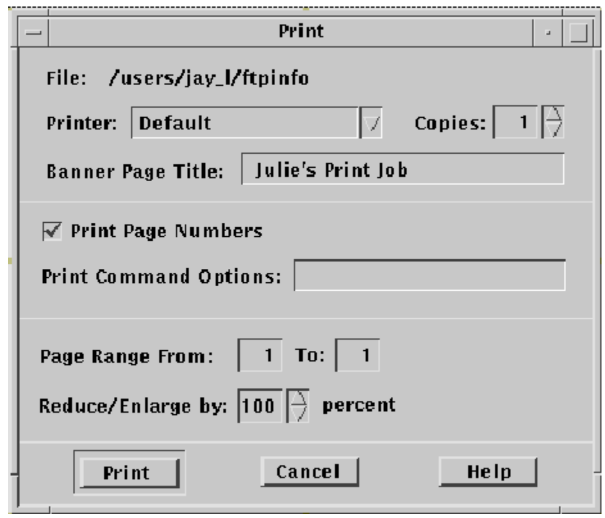


Figure 7-7 Example print dialog box layout for printing from applications

The Properties Dialog

User a properties dialog if your application provides settings that control the behavior of an application or the characteristics of an object.

Guidelines

Recommended **cz:** If your application manages objects and allows the user to see or modify settings for these objects, these settings are displayed in an object properties window that is accessible from a Properties ... choice in the Edit, <object-type>, or Selected menus, as well as from the pop-up menu associated with the object.**—<Default ¶ <Bold>Font>*page 215***

Recommended **da:** If your application provides access to a Properties or Options window, this window includes the following set of buttons in the order listed, with the specified functionality, when supported by your application.**—<Default ¶ <Bold>Font>*page 215***

OK	Applies any changes made to components in the DialogBox and dismisses it. OK may be replaced by a more appropriate label (for example, Add). The alternate label should be a verb phrase.
Apply	Applies any changes made to components in the DialogBox and does not dismiss it.
Reset	Cancels any changes that have not yet been applied by your application. The controls within the DialogBox are reset to their state since the last time the DialogBox action was applied. If no changes have been applied within the current invocation of the DialogBox, the controls are reset to their state as of when the DialogBox was first displayed.
Reset to Factory	Cancels any changes that have not yet been applied. Components in the dialog box are reset to their default state or value as specified by the vendor that delivered the application (that is, the controls are restored to the original factory settings).
Cancel	Dismisses the dialog box without performing any actions not yet applied.
Help	Provides help for the dialog box.

Recommended **db:** If your application provides a Properties window that displays settings for a selected object, the Properties window tracks the current selection and modifies the state of any controls to accurately reflect the properties of the currently selected object.**—<Default ¶ <Bold>Font>***page 216*

The About Dialog Box

The About dialog box is used to present version and other information about your application. Use as the dialog that comes up when the user chooses About <application name> from the Help menu.

Guidelines for the About Dialog Box

The About dialog box should contain a minimum set of information about the application that is visible in a single text pane.

That minimum set should be:

- Application name
- Version number
- Release date
- Copyright

Required **di:** The About dialog box should contain a Close button. Other buttons are optional, such as Help and More.**—<Default ¶ <Bold>Font>***page 218*

The following information might also be contained in the About box:

Recommended **dj:** Information about the operating system or other aspects required to run the application, for example, TriTeal Enterprise Desktop 4.0.**—<Default ¶ <Bold>Font>***page 218*

Optional **dk:** A More Information dialog box for additional information such as development team credits, licensing, client or xhost information.**—<Default ¶ <Bold>Font>***page 218*

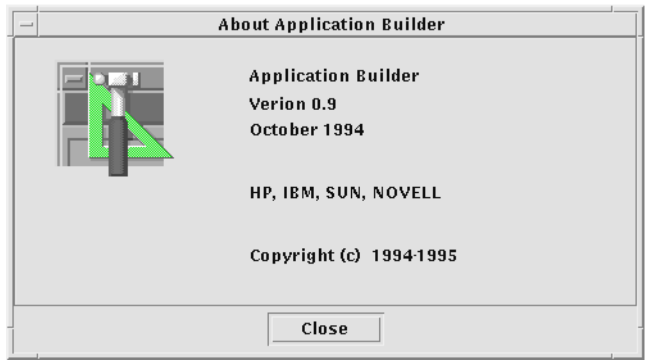


Figure 7-8 Example About dialog box

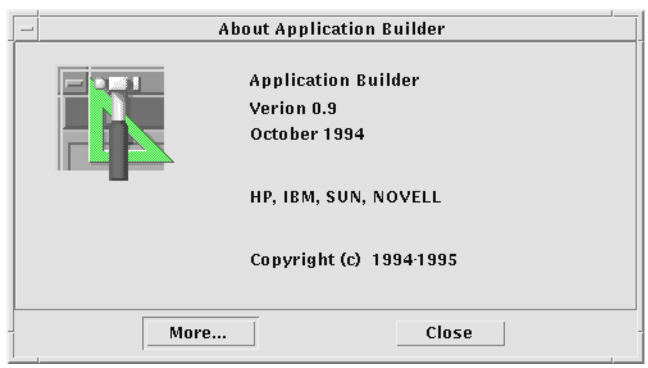


Figure 7-9 About dialog box with a More button

Application Messages 8

From time to time, an application needs to present feedback to keep the user informed about the progress of ongoing activities and to alert the user to situations that require intervention. The Common Desktop Environment Motif interface provides many ways to provide such feedback to the user. This section describes the use of error messages, informational messages, and other message dialog boxes.

Error Messages and Informational Messages

Error messages and informational messages are appropriate in different situations. Present an error message when it is crucial to bring the information to the user’s attention because the intended action cannot be carried out without user intervention. Present informational messages to describe progress, indicate short-term status, or give helpful suggestions. Informational messages should be gentle and nondisruptive to the flow of activity. Therefore, you must assume that the user may not notice informational messages. If it is important that your users see a particular piece of information, present it in an error dialog box or in another type of message dialog box.

Error Messages

Use error messages to present crucial messages to the user when user intervention is necessary for the successful completion of an operation. Error messages are intended to bring a problem to the user’s attention and to help the user resolve the problem.

Guidelines for Error Messages

Use a Motif error dialog box to present application error messages. Figure 8-1 shows a typical error dialog box. Keep in mind the basic three-part structure for error messages. Whenever possible, each message should tell the user:

- What happened
- Why it happened
- What should be done to correct the problem

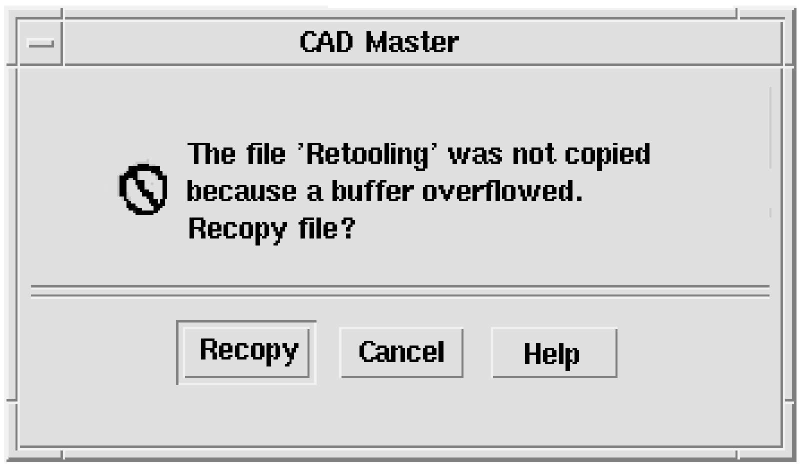


Figure 8-1 An error dialog box.

The text describes the error, why it happened, and offers to retry the operation. A Help button in the lower right will bring up appropriate online documentation.

Recommended **gd:** Error messages displayed by your application indicate the possible cause of the error and indicate the possible actions the user can take in response.—***page 230***

Keep the information in the ErrorDialog clear and concise. The idea is to alert users to the problem, have them quickly understand the problem, and learn how to recover from it. A large amount of text makes it more difficult for the user to focus on the critical information.

Recommended **gc:** Messages displayed by your application do not assume that the user has any expert knowledge about computer systems in general, or the UNIX system in particular.—***page 229***

It is appropriate to assume that the user has knowledge about basic terms used within the desktop, such as files or programs. Such knowledge can be assumed to have been learned by the user through tutorials, online help and user documentation. However, terminology that is typically understood only by an expert or frequent computer user should be avoided unless the application is specifically targeted at computer professionals. Likewise, messages returned to your application by the underlying operating system should not be passed through to the user, but instead, should be “translated” into language that can be understood by the novice user.

Try to make the error message specific to the immediate situation and as helpful as possible. For example, if the user has entered an invalid name, the error message should not simply state that an invalid name was entered. Instead, tell the user which character was invalid. If the rules for valid names are simple, describe them in the error message; otherwise, describe the rules in the online help and give the user access through the Help button.

In many cases, the only user response to an error dialog box will be to click the OK button to dismiss the dialog box. However, often it may be possible to offer to resolve the problem for the user. If you have buttons for user actions, be sure to also include a Cancel button.

Optional **ge:** Your application uses audio feedback, in addition to any messages displayed, to signal error conditions and events.—***page 230***.

Optional **gh:** Urgent conditions that require immediate attention by the user, no matter which application or desktop service the user is currently accessing, are brought to the user's attention using audiovisual notification. The alarm is signaled in the current workspace regardless of the workspace in which the application resides.—***page 230***

Some applications, such as network monitors or stock watch programs, may need to grab the user’s immediate attention to some event. Both visual and audio alarms should be used to signal the user. The user should be able to acknowledge the alarm and cause it to cease.

Keep in mind that the user may be running several applications at once, and may be focused on another application while your application is running in the background and encounters an error. The application name should appear in the title bar, to help the user identify the source of the error message.

Once users have read an error message, they may need to access other parts of their system to resolve the problem. Whenever reasonable, posting an error dialog box should not block further interaction with your application and, if at all possible, it should not block other applications.

Optional **gq:** Your application writes error messages to the Common Desktop Environment error log when it is not appropriate to display these to the user in a message dialog box, but when the message may nevertheless be useful in diagnosing problems.*—page 233*.

You might also write error messages that are displayed to the user in the error log if it would be valuable to the user or an administrator to refer to these messages at some later time. Messages written to the error log should provide additional information about the error and should state the context in which the error occurred.

Access to Online Help

Recommended **gj:** Your application provides a Help button in all message dialog boxes, except those that contain self-explanatory messages.*—page 231*

Applications should be designed with both the expert and novice user in mind. The novice user must be able to access additional information clarifying the message, the circumstances under which it might have been displayed, and what the user should do in response to the message.

The brief description of the problem in the ErrorDialog should be sufficient for the experienced user, but may not contain enough information to enable less experienced users to resolve the problem. Rather than confusing the dialog with additional text, use the Help button to take the user directly to the online documentation for a more detailed description of the error, its causes, and methods for resolving the issue. Users who still need additional help can browse the general online help facilities from there. A few notices may not need Help buttons because the text of the message will cover the condition sufficiently.

For more information on how to access online help directly from the error dialog box see the *Help System Author's and Programmer's Guide*.

Helpful Hints

Optional **gh:** Urgent conditions that require immediate attention by the user, no matter which application or desktop service the user is currently accessing, are brought to the user's attention using audiovisual notification. The alarm is signaled in the current workspace regardless of the workspace in which the application resides.***—page 230***

Applications should not send messages to command entry windows or to the UNIX console (that is, applications should not write to the default UNIX files `stdout` or `stderr`). Applications are often launched by double-clicking icons in the File Manager, Front Panel, or Application Manager, which means users will not see messages that are written to `stdout` or `stderr`. Even if the application is launched from a terminal window, the user may subsequently close the terminal window, and messages appearing there will often not be seen. Worse, if the user does not have a console window running (the console window is difficult to launch in Common Desktop Environment), messages intended for the console may blast across the screen and make everything look ugly.

Informational Messages

Use informational messages in the window footer to present progress, status, or helpful information to the user. Informational messages should *not* be used to present crucial information, because informational messages are deliberately designed to be nonobtrusive and many users may not notice them.

Guidelines for Informational Messages

Recommended **gi:** Your application uses footer messages only to communicate status, progress, or information (help) messages. It does not use the footer to present error messages.***—page 231***

Motif provides a message area at the bottom of the main window, but this is rather clumsy and ugly. A more elegant approach is to provide a wider margin below the data area of the main window where status information can be unobtrusively displayed, as shown in Figure 8-2. For other examples of the use of informational messages, see the status message area in the Common Desktop Environment Mailer.

Figure 8-2 An informational message in the lower margin of a window.

The text "Loading earth.gif..." is displayed at the start of the load and the text "Done" is added when the load completes. The entire message is removed 5 seconds later.

Informational messages in the footer area should be left-justified and displayed in a light font in keeping with their unobtrusive nature. Note that the margin where informational messages are displayed should *not* accept mouse focus. Progress messages in the footer area should normally be displayed only while the operation is in progress. Notices and other information that is no longer valid should be removed within a few seconds to avoid confusion about whether the information is current.

Other Message Dialogs

Recommended **gk:** Your application uses the appropriate style dialog box for the display of messages to the user.***—page 231***

Optional **gl:** An information dialog box is used to display status, completion of activity, or other informative types of messages to which the user need not necessarily respond other than to acknowledge having read the message.***—page 231***.

Minimally, information dialog boxes should have an OK button so that the user can dismiss the dialog box. If there is additional information available about the situations under which the message is displayed or other references for the topic to which the message relates, then a Help button should be included.

Optional **gn:** A question dialog box is used to ask questions of the user. The question is clearly worded to indicate what a Yes response or a No response means. The buttons displayed are Yes, No, and Help. Help provides additional information as to what the application will do in response to a Yes or No choice.***—page 232***

Where possible, you should replace the label for the Yes and No buttons to make it clear what action will be performed as a result of choosing either option. For example, if the user has made changes to a document and has not saved these but has chosen the application's Exit option, you might display a question dialog box that asks, “Changes have not been saved. Do you want to save these before exiting?” The buttons should be Save, Discard, Cancel, and Help. These labels allow the more experienced user to click the correct button without having to carefully read the question and relate it to the button labels.

Optional **go:** A warning dialog box is used to communicate the consequences of an action requested by the user that may result in the loss of data, system or application accessibility, or some other undesirable event. The dialog box is presented before the action is performed and offers the user the opportunity to cancel the requested operation. The buttons displayed are Yes, No, and Help, or Continue, Cancel, and Help. Help provides additional information on the consequences of performing the action requested.—***page 232***

The use of Yes and No or Continue and Cancel depends on the wording of your message. The labels for Yes and No should be replaced as suggested previously. Continue may be replaced with a label more specific to the action that will be performed.

Optional **gp:** A working dialog box is used to display in-progress information to the user when this information is not displayed in the footer of your application's window. The dialog box contains a Stop button that allows the user to terminate the activity. The operation is terminated at the next appropriate breakpoint, and a confirmation might be displayed asking whether the user really wants to stop the activity. The confirmation message might state the consequences of stopping the action.—***page 232***

Work-in-Progress Feedback

Recommended **gt:** If any command chosen by the user is expected to take longer than 2 seconds to complete, but less than 10 seconds, your application displays the standard busy pointer as feedback that the command is executing.—***page 233***

The user must receive assurance that your application has "heard" the request and is working on it. If the results of the request cannot be displayed immediately, some feedback must be provided. The busy cursor should be displayed within 0.5 seconds of execution of the command.

Recommended **gu:** If any command chosen by the user is expected to take longer than 10 seconds to complete, your application displays a working dialog box or other feedback of similar character that indicates that the application is working on the request. The feedback should reveal progress toward completion of the activity.—***page 234***

If an activity is expected to take a significant amount of time (10 seconds or more), your application should display feedback stronger than the busy pointer. Displaying the busy pointer for long amounts of time may lead the user to conclude that the application has become "hung." A progress indicator should be displayed in these scenarios that indicates that the application is still functioning and is working on the user's request. The progress indicator should show how much of the activity has been completed and what amount remains.

Recommended **gv:** When your application displays work-in-progress feedback to the user, it does not block access to other applications and services within the desktop environment.—***page 234***

Multitasking should always be supported and thus your application should allow the user to access other services while it is busy performing some activity. Preferably, the user is also able to access other features within your application even though it is currently working on another request. When this is supported, your application should display an enhanced busy pointer that indicates that the application is busy but still willing to accept input.

Designing for Accessibility 9

This chapter provides guidelines for making software applications accessible to people with disabilities.

Accessibility

Accessibility means removing barriers that can prevent people with disabilities from participating in substantial life activities, including the use of services, products, and information.

Removing barriers to access often results in benefits for a wide range of people—not only those with disabilities. For example, until curb cut ramps were placed on sidewalks, it was difficult or impossible for people in wheelchairs to cross a street. In addition to providing a wheelchair solution, curb cuts have benefited people on bicycles, as well as those pushing shopping carts and baby carriages.

Designing accessible software has similar beneficial consequences for a wide range of users. Solutions that allow use of the keyboard instead of the mouse aid users involved in keyboard-intensive tasks. Users of portables or those in open offices with telephones ringing may not be able to use or hear sounds. Providing visual cues to augment or replace audible cues assists these users, in addition to assisting hearing impaired users.

There is a growing market for accessible computer products. Approximately 40 million Americans have a disability of some type, and as the population ages, more and more people will develop age-related disabilities (25% by age 55, jumping to 50% at age 65).

Like all computer users, users with disabilities vary in age, computer experience, interests, and education. When barriers are removed, the computer gives them a tool to compete with all other users on an equal basis. Users with disabilities are engineers, artists, scientists, designers, lawyers, administrative assistants, and software engineers. The common thread among these diverse users is that computers play an important role in their daily work.

Not only does providing access provide benefits for a wide range of users, but it is also a requirement in all current federal contracts under section 508 of the Federal Rehabilitation Act. In the commercial sector, the Americans with Disabilities Act (ADA) calls for similar considerations when reasonably accommodating current and prospective employees.

Access and the Style Guide

Many users with disabilities can use application software without any adjunct adaptive software or hardware, while others may use additional technology such as screen readers or speech recognition. In either case, it is important to follow required style guidelines because those guidelines provide standard methods that make it possible for users with disabilities to access applications either directly or through adaptive software and hardware.

The easiest way to ensure accessible applications is to follow the style guidelines, and to read and follow the advice offered in this chapter.

Physical Disabilities

Physical disabilities can be the result of congenital conditions, accidents, or excessive muscular strain. Examples include spinal cord injuries, degenerative nerve diseases, stroke, and repetitive stress injuries.

While physical capabilities vary greatly between and within the disability examples cited, they have a common requirement for keyboard access to all controls, features, and information in application software. Providing comprehensive keyboard access is essential to ensure that the user who cannot utilize a mouse can productively use Motif applications.

Full keyboard access to an application is necessary, but not sufficient to make applications accessible. The other central requirement is to follow the key mapping guidelines found throughout this style guide. Consistent use of these mappings not only provides more usable applications for all users by reducing learning across applications, but also increases the effectiveness of alternate I/O technology such as speech control and screen reading software.

Guideline

Recommended **id:** All application functions are accessible from the keyboard.—*page 254*

Visual Disabilities

Visual disabilities may require use of tools ranging from reading glasses, to large-sized displays and fonts, to screen reading software that enables completely blind users to navigate and hear what is on the screen.

Reading small fonts can be challenging for users with low vision. All fonts, including those in text panes, menus, labels, and information messages should be easily configured by the user—font size and type should never be hard coded.

Interpreting information that depends upon color (for example, red = stop, green = go) can be difficult for people with visual impairments. A significant number of people are color blind and are unable to see differences between some colors. For these reasons, never use color as the only source of information.

In addition to being difficult to interpret, some background and text color combinations can result in text that is difficult to read for users with visual impairments. Again, the key is to provide choice. Never hard code color choices. Users should always have the capability to override default colors, so they can choose the colors that work best for them.

Provide meaningful names for every widget instance. Meaningful names help screen reading software give useful information to users with visual impairments. Rather than naming an eraser graphic widget5, for example, call it eraser or some other such descriptive name.

Without such descriptive information, blind or low-vision users cannot interpret unlabeled, graphically labeled, or custom widgets. Providing this information is a requirement for access in such cases. As an added bonus, meaningful widget names make for code that is easier to debug.

Finally, remember that many users with visual disabilities depend upon keyboard navigation and control, and they will not be using a pointing device.

Guidelines

Recommended **ie:** Colors should not be hard coded.—***page 254***

Recommended **if:** Graphic attributes, such as line, border, and shadow, should not be hard coded.—***page 254***

Recommended **ig:** Font sizes and styles should not be hard coded.—***page 254***

Recommended **ih:** All application code uses descriptive names for widgets. Such descriptive names for widgets using graphics instead of text (for example, palette items and icons) allow screen reading software to provide descriptive information to blind users.—***page 254***

Hearing Disabilities

People with hearing disabilities either cannot detect sound or may have difficulty distinguishing audio output from typical background noise.

Never assume that users will hear an auditory notice. Remember that users sitting in airplanes, in noisy offices, or in other public places where sound must be turned off need the same types of visual notification as hearing impaired users. Additionally, some users are able to hear audible cues only at certain frequencies or volumes. Volume and frequency of audio feedback should be easily configured by the user. Never hard code these parameters.

Sounds unaccompanied by visual notification, such as a beep indicating that a print job is complete, are of no value to users with hearing impairments or others who are not using sound. While such sounds can be valuable, never create a design that assumes sounds will be heard.

On the other hand, it would be intrusive for most users to see a warning window whenever a printout is ready. Visual notices can take the form of changing an icon, posting a message in an information area, or providing a message window as appropriate. Anyone using a system in a public area will benefit from the option of choosing to see rather than hear such notices.

The key point is to provide users with a choice. When appropriate, provide visual as well as audio notification. If visual notification does not make sense as the default behavior, then be sure to provide it as an option.

Guidelines

Recommended **ii:** Interactions do not depend upon the assumption that a user will hear an audible notification.—***page 254***

Recommended **ij:** Where appropriate, users can choose to receive cues as audio or visual information.—***page 254***

Recommended **ik:** The application does not overuse or rely exclusively on audible information.—***page 254***

Recommended **il:** Users can choose to configure the frequency and volume of audible cues.—***page 254***

Language, Cognitive, and Other Disabilities

The access guidelines outlined for visual, hearing, and physical disabilities typically benefit users with cognitive, language, and other disabilities by allowing them to choose effective means of communication, sometimes through the use of adaptive technology.

Guidelines

Recommended **im:** Tear-off menus and user configurable menus for key application features may be provided for users with language and cognitive disabilities.—***page 254***

Existing Keyboard Access Features

When designing TED Motif applications, be aware of existing system-level key mappings used by access features in the X Window System™ server. These server features, known as AccessX, provide basic workstation accessibility, typically used by people with mobility impairments. AccessX became a supported part of the X Windows server in version X11R6.

The built-in, server-level access features include:

StickyKeys Provides locking or latching of modifier keys (for example, Shift, Control) so that they can be used without simultaneously pressing the keys being modified. StickyKeys allow single-finger operation of multiple key combinations.

RepeatKeys Delays the onset of key repeat, allowing users with limited coordination time to release keys before multiple characters are sent.

SlowKeys Requires a key to be pressed and held for a set period before keypress acceptance. This allows users with limited coordination to accidentally press keys without sending keypress events.

MouseKeys An alternative to the mouse which provides keyboard-based explicit control of cursor movement and all mouse button press and release events.

ToggleKeys Indicates locking key state with a tone when pressed; for example, Caps Lock.

BounceKeys Requires a delay between keystrokes before accepting the next keypress so users with tremors can prevent the system from accepting inadvertent keypresses.

Guideline

Recommended **in:** Application keymappings do not conflict with existing system-level key mappings reserved for access features in the X Windows server as shown in Table 10-6.—***page 255***

Resources for More Information on Accessibility

For more information about software accessibility, consult the following organizations, conferences, and books.

Organizations

Clearinghouse on Computer Accommodation (COCA)
18th & F Streets, NW
Room 1213
Washington, DC 20405
(202) 501-4906

A central clearinghouse of information on technology and accessibility. COCA documentation covers products, government resources, user requirements, legal requirements, and much more.

Sensory Access Foundation
385 Sherman Avenue, Suite 2
Palo Alto, CA 94306
(415) 329-0430

A nonprofit organization that consults on application of technology "to increase options for visually and hearing impaired persons." Publishes newsletters on adaptive technology.

Special Needs Project
3463 State Street
Santa Barbara, CA 93105
(805) 683-9633

Vendor of books for professionals and families on a wide variety of disability issues.

Trace Research and Development Center
S-151 Waisman Center
1500 Highland Avenue
Madison, WI 53528
(608) 262-6966

A central source for the current information on assistive technologies as well as a major research and evaluation center. Trace distributes databases and papers on adaptive technology and resources.

Conferences

CSUN
Conference on Technology and Persons with Disabilities
Every spring in Los Angeles, California
(818) 885-2578

Closing the Gap
Conference on Microcomputer Technology in Special Education and Rehabilitation
Every fall in Minneapolis, Minnesota
(612) 248-3294

Bibliography

Brown, Carl. Computer Access in Higher Education for Students with Disabilities, 2nd Edition. George Lithograph Company, San Francisco. 1989.

Cornsweet, T.N. Visual Perception. Academic Press, New York. 1970.

Edwards, A., Edwards, E., and Mynatt, E. Enabling Technology for Users with Special Needs (InterCHI '93 Tutorial). 1993.

Johnson, M, and Elkins, S. Reporting on Disability. Advocado Press, Lousville, KY, 1989.

Managing Information Resources for Accessibility, U.S. General Services Administration Information Resources Management Service, Clearinghouse on Computer Accommodation, 1991.

Vanderheiden, G.C., Thirty-Something Million: Should They Be Exceptions?, Human Factors, 32(4), 383-396. 1990.

Vanderheiden, G.C. Making Software More Accessible for People with Disabilities, Release 1.2. Trace Research & Development Center, 1992.

Walker, W.D., Novak, M.E., Tumblin, H.R., Vanderheiden, G.C. Making the X Window System Accessible to People with Disabilities. Proceedings: 7th Annual X Technical Conference. O'Reilly & Associates, 1993.

Part 2 — Certification Checklist

Certification Checklist 10

The Common Desktop Environment Certification Checklist provides the list of requirements for Common Desktop Environment application-level certification. Common Desktop Environment requirements consist of the OSF/Motif Release 1.2 requirements with Common Desktop Environment-specific additions.

How to Use the Checklist

You certify your own application by comparing its behavior with that specified in the checklist. For each checklist item, check *Yes* only if your application performs exactly as described for that item. If you have not implemented a specified type of behavior in any manner anywhere in your application, check *N/A* (not applicable) for the items pertaining to that behavior.

The checklist describes keys using a model keyboard mechanism. Wherever keyboard input is specified, the keys are indicated by the engravings that they have on the OSF/Motif model keyboard. Mouse buttons are described using a virtual button mechanism to better describe behavior independent from the number of buttons on the mouse. For more information on the model keyboard and virtual button mechanisms, consult the Preface and Section 2.2.1, “Pointing Devices” of the *OSF/Motif Style Guide, Revision 1.2*.

Note – This checklist uses Common Desktop Environment typographical conventions for keyboard and mouse inputs. These conventions differ from those used in the *OSF/Motif Style Guide, Revision 1.2*. For information, see"What Typographic Changes and Symbols Mean" on page xv.

By default, this checklist assumes that your application is being designed for a left-to-right language environment in an English-language locale. Some sections of the checklist may require appropriate changes for other locales.

As you compare the behavior of your application to the requirements in the checklist, we recommend that you follow along in the *OSF/Motif Style Guide, Revision 1.2*. Each item in this checklist contains the corresponding section number from the OSF/Motif checklist, if the item came from that list. Each item in the checklist is also followed by a brief explanation or justification. If you do not understand a particular item, refer to the appropriate section in the OSF/Motif guide and check the glossary for any terms that are unclear.

The headings used in this checklist correspond to the headings in the *OSF/Motif Style Guide, Revision 1.2* and the checklist items are labeled with the numbers used in that book. The Common Desktop Environment-specific additions are labeled with alphabetic identifiers.

Each checklist item also has a priority label: Required, Recommended, or Optional. The Required items must be followed for an application to be Common Desktop Environment compliant. Recommended items should be followed where feasible. Optional items are alternative implementations which the interface designer can choose.

Preface

y n/a n

Required ❑ ❑ ❑ **1-1:** Each of the nonoptional keys described on the OSF/Motif model keyboard is available either as specified or by using other keys or key combinations if the specified key is unavailable (Preface).

The model keyboard does not correspond directly to any existing keyboard; rather, it assumes a keyboard with an ideal set of keys. However, to ensure consistency across applications, the nonoptional keys or substitutes for them must always be available.

Input Models

Keyboard Focus Model

	y	n/a	n		
Required	❑	❑	❑	**2-1:**	Only one window at a time has the keyboard focus. The window that has the focus is highlighted. Within the window that has the keyboard focus, only one component at a time has the focus. The keyboard focus determines which component on the screen receives keyboard events. This rule prevents confusion about which window and component have the focus.
Required	❑	❑	❑	**2-2:**	When your application uses an explicit focus policy, pressing BSelect does not move focus to a component that is not traversable or does not accept input. An explicit focus policy requires the user to explicitly select which window or component receives the keyboard focus. Generally, the user gives the focus to a window or component by pressing BSelect over it. However, this policy must not allow the user to give focus to a component that is not traversable or does not accept input.
Required	❑	❑	❑	**2-3:**	When your application uses an explicit focus policy, the component with the keyboard focus is highlighted by a location cursor. The user needs to know the location of the keyboard focus to be able to control an application.

Input Device Model

Required	❑	❑	❑	**2-4:**	Your application supports methods of interaction for keyboard-only users. All features of your application are available from the keyboard. Some users may not have access to a pointing device. These users need to be able to access the full functionality of the application from the keyboard. Additionally, advanced users will be able to use the keyboard to perform some tasks more quickly than with a pointing device.
Required	❑	❑	❑	**2-5:**	Your application uses the following bindings for mouse buttons:

BSelect Used for selection, activation, and setting the location cursor, and is the leftmost button, except for left-handed users, where it can be the rightmost button.

BTransfer Used for moving and copying elements, and is the middle mouse button, unless dragging is integrated with selection or the mouse has fewer than three buttons.

BMenu Used for popping up menus, and is the rightmost button, except for left-handed users, where it can be the leftmost button, or unless the mouse has fewer than three buttons. If the mouse has one button, BMenu is bound to Alt+BSelect.

These bindings ensure a consistent interface for using standard mouse-based operations across applications.

y n/a n

Required ❑ ❑ ❑ **2-6:** Your application does not warp the pointer unless you have given the user a means of disabling the behavior.

The pointer position is intended only as input to applications, not as an output mechanism. An application warps the pointer when it changes the pointer's position. This practice is confusing to users and reduces their sense of control over an application. Warping the pointer can also cause problems for users of absolute location pointing devices.

Required ❑ ❑ ❑ **a:** Components and applications that have functions corresponding to the Motif/Common Desktop Environment virtual keys must support those keys.

If these virtual keys are available, the following mappings should be used. Priorities indicate the importance of implementing these functions in your application.

Required ❑ ❑ ❑ **Help = F1** Pressing the Help key provides the user with help information in a window or in the status area.

Required ❑ ❑ ❑ **Properties = Control+I**
Pressing the Properties key invokes a dialog box for making object-specific settings.

	y	n/a	n	
Required	❑	❑	❑	**Undo = Control+Z** Pressing the Undo key reverses the effect of the last applied operation. This is the primary key mapping for Undo.
Optional	❑	❑	❑	**Undo = Alt+Backspace** This is a secondary key mapping for Undo. It should be supported in addition to Control+Z to help users migrating from previous versions of Motif, Microsoft Windows, or OS/2.
Required	❑	❑	❑	**Cut = Control+X** Pressing the Cut key removes the selected object and places it in the clipboard. This is the primary key mapping for Cut.
Optional	❑	❑	❑	**Cut = Shift+Delete** This is a secondary key mapping for Cut. It should be supported in addition to Control+X to help users migrating from previous versions of Motif, Microsoft Windows, or OS/2.
Required	❑	❑	❑	**Copy = Control+C** Pressing the Copy key places a copy of the selected object in the clipboard. This is the primary key mapping for Copy.
Optional	❑	❑	❑	**Copy = Control+Insert** This is a secondary key mapping for Copy. It should be supported in addition to Control+C to help users migrating from previous versions of Motif, Microsoft Windows, or OS/2.
Required	❑	❑	❑	**Paste = Control+V** Pressing the Paste key places the contents of the clipboard at the selected location. This is the primary key mapping for Paste.

	y	n/a	n		
Optional	❏	❏	❏		**Paste = Shift+Insert** This is a secondary key mapping for Paste. It should be supported in addition to Control+V to help users migrating from previous versions of Motif, Microsoft Windows, or OS/2.
Optional	❏	❏	❏		**Open = Control+O** Pressing the Open key opens the object, which is typically the default action.
Optional	❏	❏	❏		**Stop = Control+S** Pressing the Stop key cancels an operation.
Optional	❏	❏	❏		**Again = Control+A** Pressing the Again key repeats the last operation.
Optional	❏	❏	❏		**Print = Control+P** Pressing the Print key initiates printing.
Optional	❏	❏	❏		**Save = Control+S** Pressing the Save key saves the current file.
Optional	❏	❏	❏		**New = Control+N** Pressing the New key will create a new object.

Navigation

Mouse-Based Navigation

Required	❏	❏	❏	**3-1:**	When the keyboard focus policy is explicit, pressing BSelect on a component moves focus to it, except for components, such as scroll bars, that are used to adjust the size and location of other elements. BSelect provides a convenient mechanism for using the mouse to move focus when the keyboard focus policy is explicit.

	y	n/a	n		
Required	❑	❑	❑	**3-2:**	When the pointer is on a menu, your application uses BSelect Press to activate the menu in a spring-loaded manner. A spring-loaded menu is one that appears when the user presses a mouse button, remains on the screen for as long as the button is pressed, and disappears when the user releases the button. BSelect, mouse button 1, provides a means of activating spring-loaded menus that is consistent across applications.
Required	❑	❑	❑	**3-3:**	When the pointer is in an element with an inactive pop-up menu and the context of the element allows the pop-up menu to be displayed, your application uses BMenu Press to activate the pop-up menu in a spring-loaded manner. The availability of a pop-up menu can depend on the location of the pointer within an element, the contents of an element, or the selection state of an element. BMenu, mouse button 3, provides a consistent means of activating a spring-loaded pop-up menu.
Required	❑	❑	❑	**3-4:**	If the user takes an action to post a pop-up menu, and a menu can be posted for both an inner element and an outer element that contains the inner element, the pop-up menu for the internal element is posted. This specification ensures that the pop-up menu for an internal element is always accessible.
Required	❑	❑	❑	**3-5:**	Once a pop-up menu is posted, BMenu behaves just as BSelect does for any menu system. The specified operation of BMenu is for manipulating pop-up menus.
Required	❑	❑	❑	**3-6:**	BSelect is also available from within posted pop-up menus and behaves just as in any menu system. Once a pop-up menu is posted, the user can select an element from it using the standard selection mechanism, BSelect.
Required	❑	❑	❑	**3-7:**	When a menu is popped up or pulled down in a posted manner, your application places the location cursor on the menu’s default entry, or on the first entry in the menu if there is no default entry.

A posted menu remains visible until it is explicitly unposted. Placing the location cursor on the default entry allows the user to select the default operation easily. When there is no default entry, placing the location cursor on the first entry yields uniform behavior across applications.

	y	n/a	n		
Required	❑	❑	❑	**3-8:**	Your application removes a spring-loaded menu system when the mouse button that activated it is released, except when the button is released on a cascading button in the menu hierarchy.

The concept of a spring-loaded menu system requires that the menu disappear when the mouse button is released.

Required ❑ ❑ ❑ **3-9:** While a spring-loaded menu system is popped up or pulled down, moving the pointer within the menu system moves the location cursor to track the pointer.

Once a spring-loaded menu system has appeared on the screen, the user needs to be able to maneuver the location cursor through the menu system using the mouse.

Required ❑ ❑ ❑ **3-10:** When a spring-loaded menu system is popped up or pulled down and the pointer rests on a cascading button, the associated menu is pulled down and becomes traversable. The associated menu is removed, possibly after a short delay, when the pointer moves to a menu item outside of the menu or its cascading button.

The user needs to be able to use the mouse to access all of the associated menus of a menu system. This feature allows the user to move quickly to any menu in a menu system.

Required ❑ ❑ ❑ **3-11:** When a spring-loaded menu system that is part of the menu bar is pulled down, moving the pointer to any other element on the menu bar unposts the current menu system and posts the pull-down menu associated with the new element.

This feature of a spring-loaded menu system allows the user to browse quickly through all of the menus attached to a menu bar.

Required ❑ ❑ ❑ **3-12:** When a spring-loaded menu system is popped up or pulled down, and the button that activated the menu system is released within a component in the menu system, that component is activated. If the release is on a cascading button or an option button, the associated menu is activated in a posted manner if it was not posted prior to the associated button press.

Releasing the mouse button that activated a spring-loaded menu provides a means of activating a menu element that is consistent across applications.

	y	n/a	n		
Required	❑	❑	❑	**3-13:**	When the pointer is in an area with a pop-up menu, your application uses BMenu Click to activate the menu in a posted manner if it was not posted prior to the BMenu Click. BMenu Click provides a means of posting a pop-up menu that is consistent across applications.
Required	❑	❑	❑	**3-14:**	Once a pull-down or option menu is posted, BSelect Press in the menu system causes the menu to behave as a spring-loaded menu. This feature of a posted pull-down or option menu allows the user to switch easily between using a posted menu and a spring-loaded menu.
Required	❑	❑	❑	**3-15:**	If a button press unposts a menu and that button press is not also passed to the underlying component, subsequent events up to and including the button release are not passed to the underlying component. When a button press unposts a menu, the press can be passed to the underlying component. Whether or not it is passed to the underlying component, the press can have additional effects, such as raising and giving focus to the underlying window. If the press is not passed to the underlying component, events up to and including the release must not be passed to that component.
Required	❑	❑	❑	**3-16:**	Once a pop-up menu is posted, BSelect Press or BMenu Press in the menu system causes the menu to behave as a spring-loaded menu. This feature of a posted pop-up menu allows the user to switch easily between using a posted menu and a spring-loaded menu.
Optional	❑	❑	❑	**b:**	BMenu Press or BMenu Click on a menu bar item displays the menu.
Required	❑	❑	❑	**c:**	BMenu Press or BMenu Click on an option button displays the option menu.
Required	❑	❑	❑	**d:**	BSelect Press on a text field causes the text cursor to be inserted at the mouse cursor position.

Keyboard-Based Navigation

	y	n/a	n		
Required	❑	❑	❑	**3-17:**	In a text component, the text cursor is shown differently when the component does and does not have the keyboard focus. In a text component, the text cursor serves as the location cursor and, therefore, must indicate whether the component has keyboard focus.
Required	❑	❑	❑	**3-18:**	If a text component indicates that it has lost the keyboard focus by hiding the text cursor and if the component subsequently regains the focus, the cursor reappears at the same position it had when the component lost focus. To ensure predictability, it is important that the text cursor not change position when a text component loses and then regains the keyboard focus.
Required	❑	❑	❑	**3-19:**	If a small component, such as a sash, indicates that it has the keyboard focus by filling, no other meaning is associated with the filled state. This rule reduces possible confusion about the significance of filling in a small component.
Required	❑	❑	❑	**3-20:**	All components are designed and positioned within your application so that adding and removing each component's location cursor does not change the amount of space that the component takes up on the screen. For visual consistency, the sizes and positions of components should not change when keyboard focus moves from one component to another.
Required	❑	❑	❑	**3-21:**	Control+Tab moves the location cursor to the next field, and Control+Shift+Tab moves the location cursor to the previous field. Unless Tab and Shift+Tab are used for internal navigation within a field, Tab also moves the location cursor to the next field, and Shift+Tab also moves the location cursor to the previous field. These keys provide a consistent means of navigating among fields in a window.
Required	❑	❑	❑	**3-22:**	Tab (if not used for internal navigation) and Control+Tab move the location cursor forward through fields in a window according to the following rules:

- If the next field is a control, Tab (if not used for internal navigation) and Control+Tab move the location cursor to that control.
- If the next field is a group, Tab (if not used for internal navigation) and Control+Tab move the location cursor to a traversable component within the group.
- If the next field contains no traversable components, Tab (if not used for internal navigation) and Control+Tab skip the field.

These rules ensure the consistent operation of Tab (if not used for internal navigation) and Control+Tab across applications.

y n/a n

Required ❑ ❑ ❑ **3-23:** Shift+Tab (if not used for internal navigation) and Control+Shift+Tab move the location cursor backward through fields in the order opposite to that of Tab (if not used for internal navigation) and Control+Tab.

These rules result in the uniform operation of Shift+Tab (if not used for internal navigation) and Control+Shift+Tab across applications.

Required ❑ ❑ ❑ **3-24:** When a window acquires focus, the location cursor is placed on the control that last had focus in the window, providing that all the following conditions are met:

- The window uses an explicit keyboard focus policy.
- The window acquires the focus through keyboard navigation or through a button press other than within the client area of the window.
- The window had the focus at some time in the past.
- The control that last had focus in the window is still traversable.

This rule ensures that when the user returns to a window after navigating away, the focus returns to the component where the user left it.

Required ❑ ❑ ❑ **3-25:** Field navigation wraps between the first and last fields in the window.

This feature of field navigation provides the user with a convenient way to move through all of the fields in a window.

Required ❑ ❑ ❑ **3-26:** When the Down Arrow and Up Arrow keys are used for component navigation within a field, they behave according to the following rules:

In a left-to-right language environment, the Down Arrow key moves the location cursor through all traversable controls in the field, starting at the upper left and ending at the lower right, then wrapping to the upper left. If the controls are aligned in a matrix-like arrangement, Down Arrow first traverses one column from top to bottom, then traverses the column to its right, and so on. In a right-to-left language environment, Down Arrow moves the location cursor through all traversable controls, starting at the upper right and ending at the lower left.

- Up Arrow moves the location cursor through all traversable controls in the field in the order opposite to that of Down Arrow.

These rules ensure a consistent means of navigating among components using the directional keys.

y n/a n

Required ❑ ❑ ❑ **3-27:** When the Right Arrow and Left Arrow keys are used for component navigation within a field, they behave according to the following rules:

- In a left-to-right language environment, the Right Arrow moves the location cursor through all traversable controls in the field, starting at the upper left and ending at the lower right, then wrapping to the upper left. If the controls are aligned in a matrix-like arrangement, the Right Arrow first traverses one row from left to right, then traverses the row below it, and so on. In a right-to-left language environment, the Right Arrow moves the location cursor through all traversable controls, starting at the lower left and ending at the upper right.
- Left Arrow moves the location cursor through all traversable controls in the field in the order opposite to that of the Right Arrow.

These rules ensure a consistent means of navigating among components using the directional keys.

Required ❑ ❑ ❑ **3-28:** If a control uses the Right Arrow and Left Arrow for internal navigation, Begin moves the location cursor to the leftmost edge of the data or the leftmost element in a left-to-right language environment. In a right-to-left language environment, Begin moves the location cursor to the rightmost edge of the data or the rightmost element.

This rule permits convenient navigation to the left or right edge of the data or the left or right element in a control.

y n/a n

Required ❑ ❑ ❑ **3-29:** If a control uses the Right Arrow and Left Arrow keys for internal navigation, the End key moves the location cursor to the rightmost edge of the data or the rightmost element in a left-to-right language environment. In a right-to-left language environment, End moves the location cursor to the leftmost edge of the data or the leftmost element.

This rule permits convenient navigation to the left or right edge of the data or the left or right element in a control.

Required ❑ ❑ ❑ **3-30:** If a control uses the Up Arrow and Down Arrow keys for internal navigation, Control+Begin moves the location cursor to one of the following:

- The first element.
- The topmost edge of the data.
- In a left-to-right language environment, the topmost left edge of the data; in a right-to-left language environment, the topmost right edge of the data.

This rule permits convenient navigation to the beginning of the data in a control.

Required ❑ ❑ ❑ **3-31:** If a control uses the Up Arrow and Down Arrow keys for internal navigation, Control+End moves the location cursor to one of the following:

- The last element.
- The bottommost edge of the data.
- In a left-to-right language environment, the bottommost right edge of the data; in a right-to-left language environment, the bottommost left edge of the data.

This rule permits convenient navigation to the end of the data in a control.

Optional ❑ ❑ ❑ **e:** Each time a new window is opened, keyboard focus is placed in the first field or location within the window or in a default location, if this is appropriate for the particular window.

Required ❑ ❑ ❑ **f:** The Tab key moves input focus between push buttons within a group.

The arrow keys also move the selected focus per the *OSF/Motif Style Guide, Revision 1.2*.

y n/a n

Required ❑ ❑ ❑ **g:** Use the Control, Shift, and Alt keys only to modify the function of other keys or key combinations.

Optional ❑ ❑ ❑ **h:** Use the Alt key only to provide access to mnemonics.

Menu Traversal

Required ❑ ❑ ❑ **3-32:** If the user traverses to a menu while the keyboard focus policy is implicit, the focus policy temporarily changes to explicit and reverts to implicit whenever the user traverses out of the menu system.

Menus must always be traversable, even when the keyboard focus policy is generally implicit.

Required ❑ ❑ ❑ **3-33:** Your application uses the F10 key to activate the menu bar system if it is inactive. The location cursor is placed on the first traversable cascading button in the menu bar. If there are no traversable cascading buttons, the key does nothing.

F10 provides a consistent means of traversing to the menu bar using the keyboard.

Required ❑ ❑ ❑ **3-34:** When the keyboard focus is in an element with an inactive pop-up menu and the context of the element allows the pop-up menu to be displayed, your application uses the menu key to activate the pop-up menu. The location cursor is placed on the default item of the menu, or on the first traversable item in the pop-up menu if there is no default item.

The Menu key provides a uniform way of activating a pop-up menu from the keyboard.

Required ❑ ❑ ❑ **3-35:** When the keyboard focus is in an option button, your application uses the Select key or the Spacebar to post the option menu. The location cursor is placed on the previously selected item in the option menu; or, if the option menu has been pulled down for the first time, the location cursor is placed on the default item in the menu. If there is an active option menu, the Return, Select, or Spacebar keys select the current item in the option menu, unpost the menu system, and return the location cursor to the option button.

These keys provide a means of posting an option menu from the keyboard that is consistent across applications.

	y	n/a	n		
Required	❑	❑	❑	**3-36:**	Your application uses the Down Arrow, Left Arrow, Right Arrow, and Up Arrow keys to traverse through the items in a menu system.

The Down Arrow, Left Arrow, Right Arrow, and Up Arrow directional keys provide a consistent means of navigating among items in a menu system.

Required ❑ ❑ ❑ **3-37:** When a menu traversal action traverses to the next or previous component in a menu or menu bar, the order of traversal and the wrapping behavior are the same as that of the corresponding component navigation action within a field.

This specification provides consistency between menu traversal and component navigation within a field.

Required ❑ ❑ ❑ **3-38:** If your application uses any two-dimensional menus, they do not contain any cascading buttons.

Cascading buttons in a two-dimensional menu would restrict the user's ability to navigate to all of the elements of the menu using the keyboard.

Required ❑ ❑ ❑ **3-39:** When focus is on a component in a menu or menu bar system, the Down Arrow key behaves in the following way:

- If the component is in a vertical or two-dimensional menu, traverse down to the next traversable component, wrapping within the menu if necessary.
- If the component is in a menu bar, and the component with the keyboard focus is a cascading button, post its associated pull-down menu and traverse to the default entry in the menu or, if the menu has no default, to the first traversable entry in the menu.

This rule results in consistent operation of the directional keys in a menu or menu bar system.

Required ❑ ❑ ❑ **3-40:** When focus is on a component in a menu or menu bar system, the Up Arrow key behaves in the following way:

If the component is in a vertical or two-dimensional menu, this action traverses up to the previous traversable component, wrapping within the menu if necessary, and proceeding in the order opposite to that of the Down Arrow key.

This rule results in consistent operation of the directional keys in a menu or menu bar system.

y n/a n

Required ❑ ❑ ❑ **3-41:** When focus is on a component in a menu or menu bar system, the Left Arrow key behaves in the following way:

- If the component is in a menu bar or two-dimensional menu, but not at the left edge, traverse left to the previous traversable component.
- If the component is at the left edge of a menu bar, wrap within the menu bar.
- If the component is at the left edge of a vertical or two-dimensional menu that is the child of a vertical or two-dimensional menu, unpost the current menu and traverse to the parent cascading button.
- If the component is at the left edge of a vertical or two-dimensional menu that is the child of a menu bar, unpost the current menu and traverse left to the previous traversable entry in the menu bar. If that entry is a cascading button, post its associated pull-down menu and traverse to the default entry in the menu or, if the menu has no default, to the first traversable entry in the menu.

This rule results in consistent operation of the directional keys in a menu or menu bar system.

Required ❑ ❑ ❑ **3-42:** When focus is on a component in a menu or menu bar system, the Right Arrow key behaves in the following way:

- If the component is a cascading button in a vertical menu, post its associated pull-down menu and traverse to the default entry in the menu or, if the menu has no default, to the first traversable entry in the menu.
- If the component is in a menu bar or two-dimensional menu, but not at the right edge, traverse right to the next traversable component.
- If the component is at the right edge of a menu bar, wrap within the menu bar.

- If the component is not a cascading button and is at the right edge of a vertical or two-dimensional menu, and if the current menu has an ancestor cascading button (typically in a menu bar) from which the Down Arrow key posts its associated pull-down menu, unpost the menu system pulled down from the nearest such ancestor cascading button and traverse right from that cascading button to the next traversable component. If that component is a cascading button, post its associated pull-down menu and traverse to the default entry in the menu or, if the menu has no default, to the first traversable entry in the menu.

This rule results in consistent operation of the directional keys in a menu or menu bar system.

y n/a n

Required ❑ ❑ ❑ **3-43:** All menu traversal actions, with the exception of menu posting, traverse to tear-off buttons in the same way as for other menu entries.

Traversal of tear-off buttons needs to be consistent with traversal of other menu items.

Required ❑ ❑ ❑ **3-44:** If your application uses the F10, Menu, or Cancel key to unpost an entire menu system and an explicit focus policy is in use, the location cursor is moved back to the component that had it before the menu system was posted.

Returning the location cursor to the component that had it previously allows the user to resume a task without disruption.

Scrollable Component Navigation

Required ❑ ❑ ❑ **3-45:** Any scrollable components within your application support the appropriate navigation and scrolling operations. Your application uses the page navigation keys Page Up, Page Down, Control+Page Up (for Page Left), and Control+Page Down (for Page Right) for scrolling the visible region by a page increment.

A user needs to be able to view and access the entire contents of a scrollable component.

Required ❑ ❑ ❑ **3-46:** When scrolling by a page, your application leaves at least one unit of overlap between the old and new pages.

The overlap between one page and the next yields visual continuity for the user.

	y	n/a	n		
Required	❑	❑	❑	**3-47:**	Any keyboard operation that moves the cursor to or in the component, or that inserts, deletes, or modifies items at the cursor location scrolls the component so that the cursor is visible when the operation is complete.

The user needs to be able to see the results of moving the location cursor or operating on the contents of the scrollable component.

Required ❑ ❑ ❑ **3-48:** If a mouse-based scrolling action is in progress, the Cancel key cancels the scrolling action and returns the scrolling device to its state prior to the start of the scrolling operation.

The Cancel key provides a convenient way for the user to cancel a scrolling operation.

Selection

Selection Models

Required ❑ ❑ ❑ **4-1:** Your system supports five selection models: single selection, browse selection, multiple selection, range selection, and discontiguous selection.

Each collection has one or more appropriate selection models. The model limits the kinds of choices the user can make in the collection. Some collections enforce a selection model, while others allow the user or application to change it.

Mouse-Based Single Selection

Required ❑ ❑ ❑ **4-2:** In a collection that uses single selection, when BSelect is clicked in a deselected element, the location cursor moves to that element, that element is selected, and any other selection in the collection is deselected.

Single selection is the simplest selection model, used to select a single element. BSelect, the first mouse button, provides a consistent means of selecting an object within a group using the mouse.

Mouse-Based Browse Selection

y n/a n

Required ❑ ❑ ❑ **4-3:** In a collection that uses browse selection, when BSelect is released in a selectable element, that element is selected, and any other selection in the collection is deselected. As BSelect is dragged through selectable elements, each element under the pointer is selected, and the previously selected element is deselected. The selection remains on the element where BSelect is released, and the location cursor is moved there.

Browse selection is used to select a single element. It also allows the user to browse through the collection by dragging BSelect. See "Mouse-Based Multiple Selection" on page 163.

4-4: This item has been deleted.

Mouse-Based Multiple Selection

Required ❑ ❑ ❑ **i:** If your application contains collections that follow the multiple selection model, BAdjust is supported and behaves just like BSelect, when the BTransfer button is currently configured to behave as BAdjust.

On a three-button mouse, button 2 is typically used for the BTransfer (or BSelect) function. However, in a Common Desktop Environment environment, the user may change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. BAdjust can be used to toggle the selection state of elements under the multiple selection model.

Required ❑ ❑ ❑ **j:** In a collection that uses multiple selection, clicking BSelect or BAdjust on an unselected element adds that element to the current selection. Clicking BSelect or BAdjust on a selected element removes that element from the current selection. Clicking BSelect or BAdjust moves the location cursor to that element.

Mouse-Based Range Selection

4-5: This item has been replaced by items **k:** and **l:**.

Required ❑ ❑ ❑ **k:** In a collection that uses range selection, pressing BSelect on an unselected element sets an anchor on the element, or at the position where BSelect was pressed, and deselects all elements in the collection. If BSelect is released before the drag threshold has been exceeded, then the element under the pointer should be selected. If BSelect Motion exceeds the drag threshold, then a new selection should begin. The anchor and the current position of the pointer determine the current range. As BSelect is dragged through the collection, the current range is highlighted. When BSelect is released, the anchor does not move, and all the elements within the current range are selected.

Range selection allows the user to select multiple contiguous elements of a collection by pressing and dragging BSelect.

y n/a n

Required ❑ ❑ ❑ **l:** In a collection that uses range selection, pressing BSelect on an currently selected element should *not* cause all other elements in the selection set to be deselected. If BSelect is released before the drag threshold is exceeded, then, at that point, all other elements should be deselected and the element under the pointer should remain selected. If BSelect Motion exceeds the drag threshold, then no element should be deselected and a drag operation should begin.

Required ❑ ❑ ❑ **4-6:** In a text-like collection that uses range selection, the anchor point is the text pointer position when BSelect is pressed, and the current range consists of all elements between the anchor point and the current text pointer position.

In text-like collections, elements are ordered linearly, and a text pointer is always considered to be between elements at a point near the actual pointer position.

Required ❑ ❑ ❑ **4-7:** In a graphics-like or list-like collection that uses a marquee to indicate the range of a range selection, the current range consists of those elements that fall completely within the marquee. If there is an anchor element, the marquee is always made large enough to enclose it completely. Otherwise, an anchor point is used and is the point at which BSelect was pressed; the anchor point determines one corner of the marquee. If the collection is not arranged as a list or matrix, the marquee is extended to the pointer position. If the collection is arranged as a list or matrix, the marquee is either extended to completely enclose the element under the pointer or extended to the pointer position. Clicking BSelect on a selectable element makes it an anchor element, selects it, and deselects all other elements.

A marquee, or highlighted rectangle, is often used to indicate the range of a selection in graphics-like and list-like collections.

4-8: This item has been deleted.

	y	n/a	n		
Required	❑	❑	❑	**m:**	If your application contains collections that follow the range selection model, BAdjust is supported and behaves just like Shift+BSelect, when the BTransfer button is currently configured to behave as BAdjust.

On a three-button mouse, button 2 is typically used for the BTransfer function. However, in a Common Desktop Environment environment, the user may change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. BAdjust can be used to extend the selection set in the same manner as Shift+BSelect.

	y	n/a	n		
Required	❑	❑	❑	**n:**	In a collection that uses range selection, when the user presses Shift+BSelect, or BAdjust, the anchor remains unchanged, and an extended range for the selection is determined, based on one of the extension models.
Optional	❑	❑	❑	**Reselect**	The extended range is determined by the anchor and the current pointer position, in exactly the same manner as when the selection was initially made.
Optional	❑	❑	❑	**Enlarge Only**	The selection can only be enlarged. The extended range is determined by the anchor and the current pointer position, but then is enlarged to include the current selection.
Optional	❑	❑	❑	**Balance Beam**	A balance point is defined at the midpoint of the current selection. When the user presses Shift+BSelect or BAdjust on the opposite side of the balance point from the anchor, this model works exactly like the reselect model. When the user presses Shift+BSelect, BAdjust, or starts a navigation action modified by Shift on the same side of the balance point as the anchor, this model moves the anchor to the opposite end of the selection and then works exactly like the reselect model.

When the user releases BSelect or BAdjust, the anchor does not move, all the elements within the extended range are selected, and all the elements outside of it are deselected.

Mouse-Based Discontiguous Selection

y n/a n

Required ❑ ❑ ❑ **4-9:** In a collection that uses discontiguous selection, the behavior of BSelect is exactly the same as in the range selection model. After the user sets the anchor with BSelect, Shift+BSelect works exactly as in the range selection model.

Discontiguous selection is an extension of range selection that allows the user to select multiple discontiguous ranges of elements.

Required ❑ ❑ ❑ **4-10:** In a collection that uses discontiguous selection, when the current selection is not empty and the user clicks Control+BSelect, the anchor and location cursor move to that point. If the current selection is not empty and the user clicks Control+BSelect on an element, the selection state of that element is toggled, and that element becomes the anchor element.

In discontiguous selection, Control+BSelect Click provides a convenient means of moving the anchor and toggling the selection state of the element under the pointer.

Required ❑ ❑ ❑ **4-11:** In a collection that uses discontiguous selection, Control+BSelect Motion toggles the selection state of a range of elements. The range itself is determined exactly as for BSelect Motion. Releasing Control+BSelect toggles the selection state of the elements in the range according to one of two models:

Anchor Toggle Toggling is based on an anchor element. If the range is anchored by a point, and is not empty, the anchor element is set to the element within the range that is nearest to the anchor point. Toggling sets the selection state of all elements in the range to the inverse of the initial state of the anchor element.

Full Toggle The selection state of each element in the extended range is toggled.

In discontiguous selection, Control+BSelect provides a convenient means of toggling the selection state of elements in a range.

Required ❑ ❑ ❑ **4-12:** In a collection that uses discontiguous selection, after Control+BSelect toggles a selection, Shift+BSelect or Control+Shift+BSelect extends the range of toggled elements. The extended range is determined in exactly the same way as when Shift BSelect is used to extend a range selection. When the user releases Control+Shift+BSelect, the selection state of elements added to the range is determined by the toggle model in use (either Anchor Toggle or Full Toggle). If elements are removed from the range, they either revert to their state prior to the last use of Control+BSelect or change to the state opposite that of the elements remaining within the extended range.

Shift+BSelect and Control+Shift+BSelect provide a convenient means of extending the range of toggled elements.

	y	n/a	n	
Required	❑	❑	❑	**o:** In a collection that uses discontiguous selection, BAdjust can be used to extend the range of a discontiguous selection. The extended range is determined in exactly the same way as when BAdjust is used to extend a range selection.

On a three-button mouse, mouse button 2 is typically used for the BTransfer function. However, in a Common Desktop Environment environment, the user may change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. BAdjust can be used to extend the selection set in the same manner as Shift+BSelect.

Keyboard Selection

Required ❑ ❑ ❑ **4-13:** The selection models support keyboard selection modes according to the following rules:

- Single selection supports only add mode.
- Browse selection supports only normal mode.
- Multiple selection supports only add mode.
- Range selection supports normal mode. If it also supports add mode, normal mode is the default.
- Discontiguous selection supports both normal mode and add mode. Normal mode is the default.

Selection must be available from the keyboard. In normal mode, used for making simple contiguous selections from the keyboard, the location cursor is never disjoint from the current selection. In add mode, used for making more complex and possibly disjoint selections, the location cursor can move independently of the current selection.

	y	n/a	n	
Required	❑	❑	❑	**4-14:** If a collection supports both normal mode and add mode, Shift+F8 switches from one mode to the other. Mouse-based selection does not change when the keyboard selection mode changes. In editable components, add mode is a temporary mode that is exited when the user performs an operation on the selection or deselects the selection.

Shift+F8 provides a convenient means of switching between normal mode and add mode.

Keyboard-Based Single Selection

Required ❑ ❑ ❑ **4-15:** In a collection that uses single selection, the navigation keys move the location cursor independently from the selected element. If the user presses the Select key or the Spacebar on an unselected element, the element with the location cursor is selected, and any other selection in the collection is deselected.

Single selection supports only add mode. Pressing the Select key or the Spacebar is similar to clicking BSelect.

Keyboard-Based Browse Selection

Required ❑ ❑ ❑ **4-16:** In a collection that uses browse selection, the navigation keys move the location cursor and select the cursored element, deselecting any other element. If the application has deselected all elements or if the cursor is left disjoint from the selection, the Select key or the Spacebar selects the cursored element and deselects any other element.

Browse selection supports only normal mode. A navigation operation is similar to dragging BSelect.

Keyboard-Based Multiple Selection

	y	n/a	n	
Required	❑	❑	❑	**4-17:** In a collection that uses multiple selection, the navigation keys move the location cursor independently from the current selection. The Select key or the Spacebar on an unselected element adds the element to the current selection. Pressing the Select key or the Spacebar on a selected element removes the element from the current selection. Multiple selection supports only add mode. Pressing the Select key or the Spacebar is similar to clicking BSelect.

Keyboard-Based Range Selection

Required	❑	❑	❑	**4-18:** In a collection that uses range selection and is in normal mode, the navigation keys move the location cursor and deselect the current selection. If the cursor is on an element, it is selected. The anchor moves with the location cursor. Text-like collections can use a different model in which the navigation keys leave the anchor at its current location, except that, if the current selection is not empty, it is deselected and the anchor is moved to the location of the cursor prior to navigation. Range selection supports normal mode, and, if the collection also supports add mode, normal mode is the default.
Required	❑	❑	❑	**4-19:** In a collection that uses range selection, whether in normal mode or add mode, the Select key or Spacebar (except in a text component) moves the anchor to the cursor, deselects the current selection, and, if the cursor is on an element, selects the element. Unless the anchor is on a deselected item, Shift+Select or Shift+Spacebar (except in text) extends the selection from the anchor to the cursor, based on the extension model used by Shift+BSelect (Reselect, Enlarge Only, or Balance Beam). In range selection, pressing the Select key or Spacebar is similar to clicking BSelect, and pressing Shift+Select or Shift+Spacebar extends the range as with Shift+BSelect.
Required	❑	❑	❑	**4-20:** In a collection that uses range selection and is in normal mode, using Shift in conjunction with the navigation keys extends the selection, based on the extension model used by Shift+BSelect. If the current selection is

empty, the anchor is first moved to the cursor. The cursor is then moved according to the navigation keys, and the selection is extended based on the extension model used by Shift+BSelect.

In range selection, shifted navigation extends the selection in a similar manner to dragging Shift+BSelect.

y n/a n

Required ❑ ❑ ❑ **4-21:** In a collection that uses range selection and is in add mode, the navigation keys move the location cursor but leave the anchor unchanged. Shifted navigation moves the location cursor according to the navigation keys, and the selection is extended based on the extension model used by Shift+BSelect.

Shifted navigation in add mode is similar to shifted navigation in normal mode, except that when the selection is empty the anchor does not move to the cursor prior to navigation.

Keyboard-Based Discontiguous Selection

Required ❑ ❑ ❑ **4-22:** In a collection that uses discontiguous selection and is in normal mode, all keyboard operations have the same effect as in the range selection model.

Normal mode does not permit multiple discontiguous selections.

Required ❑ ❑ ❑ **4-23:** In a collection that uses discontiguous selection and is in add mode, the Select key or Spacebar moves the anchor to the location cursor and initiates toggling. If the cursor is on an element, the selection state of that element is toggled, but the selection state of all other elements remains unchanged. Shift+Select or Shift+Spacebar and shifted navigation operations extend the selection between the anchor and the location cursor, based on the toggle mechanism used by Control+BSelect (Anchor Toggle or Full Toggle).

Add mode permits use of the keyboard to make multiple discontiguous selections.

Canceling a Selection

Required ❑ ❑ ❑ **4-24:** Your application uses the Cancel key to cancel or undo any incomplete motion operation used for selection. Once the user presses the Cancel key to cancel a motion operation, the application ignores subsequent key and button releases until after all buttons and keys are released. Pressing the Cancel key while extending or toggling leaves the selection state of all elements as they were prior to the button press.

The Cancel key allows the user to cancel an incomplete selection operation quickly and consistently.

Autoscrolling and Selection

y n/a n

Required ❑ ❑ ❑ **4-25:** If the user drags the pointer out of a scrollable collection during a motion-based selection operation, autoscrolling is used to scroll the collection in the direction of the pointer. If the user presses the Cancel key with BSelect pressed, the selection operation is canceled.

Autoscrolling provides a convenient means of extending a selection to elements outside the viewport of a scrollable collection.

Selecting and Deselecting All Elements

Required ❑ ❑ ❑ **4-26:** In a collection that uses multiple, range, or discontiguous selection, Control+/ selects all the elements in the collection, places the anchor at the beginning of the collection, and leaves the location cursor at its previous position.

Control+/ provides the user with a convenient means of selecting all of the objects in a collection.

Required ❑ ❑ ❑ **4-27:** In a collection that is in add mode, Control+\ deselects all the elements in the collection. In a collection that is in normal mode, Control+\ deselects all the elements in the collection, except the element with the location cursor if the location cursor is being displayed. In either mode, Control+\ leaves the location cursor at its current position and moves the anchor to the location cursor.

Control+\ allows the user to deselect all of the selected objects quickly and uniformly.

y n/a n

Using Mnemonics for Elements

Required ❑ ❑ ❑ **4-28:** If your application supports mnemonics associated with selectable elements, typing a mnemonic while the collection has the keyboard focus is equivalent to moving the location cursor to the element and pressing the Select key or Spacebar.

Mnemonics within a collection of selectable elements provide an additional selection method.

Selection Actions

Required ❑ ❑ ❑ **4-29:** When the keyboard focus policy is explicit, the destination component is the editable component that last had the keyboard focus. When the keyboard focus policy is implicit, the destination component is the editable component that last received mouse button or keyboard input.

The destination component is used to identify the component on which certain operations, primarily data transfer operations, act. There is only one destination component at a time.

Required ❑ ❑ ❑ **4-30:** If the keyboard focus is in a component (or a pop-up menu of a component) that supports selections, operations that act on a selection act on the selection in that component.

A selection operation acts on the component that has focus, if that component supports selections.

Required ❑ ❑ ❑ **4-31:** If the keyboard focus is in a component (or a pop-up menu of a component) that supports some operation that does not act on a selection, invoking the operation acts on that component.

An operation that does not act on a selection acts on the component that has focus, if that component supports the operation.

Required ❑ ❑ ❑ **4-32:** Inserting or pasting elements into a selection, except for a primary transfer operation at the bounds of the primary selection, first deletes the selection if pending delete is enabled.

Pending delete controls the conditions under which the selection is deleted. It is enabled by default.

	y	n/a	n		
Required	❑	❑	❑	**4-33:**	In normal mode, inserting or pasting elements disjoint from the selection also deselects the selection, except for primary transfer operations whose source and destination are in the same collection. In add mode, the selection is not deselected. In add mode, a transfer operation that is disjoint from the selection does not affect the selection.
Required	❑	❑	❑	**4-34:**	In editable list-like and graphics-like collections, Delete deletes the selected elements. Delete provides a consistent means of deleting the selection.
Required	❑	❑	❑	**4-35:**	In editable text-like collections, Delete and Backspace behave as follows: • If the selection is not empty and the control is in normal mode, the selection is deleted. • If the selection is not empty, the control is in add mode, and the cursor is not disjoint from the selection, the selection is deleted. • If the selection is not empty and the control is in add mode, but the cursor is disjoint from the selection, Delete deletes one character forward, and Backspace deletes one character backward. • If the selection is empty, Delete deletes one character forward, and Backspace deletes one character backward. In text, Delete and Backspace provide a convenient way to delete the entire selection or single characters.

Transfer Models

	y	n/a	n		
Required	❑	❑	❑	**4-36:**	If the move, copy, or link operation the user requests is not available, the transfer operation fails. Three transfer operations are generally available: copy, move, and link. The user requests one of these operations by pressing the buttons or keys appropriate for the type of transfer. In general, for mouse-based operations, the modifier Control forces a copy, Shift forces a move, and Control+Shift forces a link. However, any requested transfer operation must fail if that operation is not available.

y n/a n

Required ❑ ❑ ❑ **4-37:** If a collection does not have a fixed insertion point or keep elements ordered in a specific way, the insertion position for transferred data is determined as follows:

- For BTransfer-based (or BSelect) primary and drag transfer operations, excepted as noted below for text collections, the insertion position is the position at which the user releases BTransfer (or BSelect).
- In a text-like collection, when the user drops selected text, the insertion position is the position at which the user releases BTransfer (or Bselect). When the user drops an icon, the insertion position is the text cursor and the data is pasted before it.
- In a list-like collection, the insertion position for other transfer operations is the element with the location cursor, and the data is pasted before it.

The insertion position is the position in the destination where transferred data is placed. Some mouse-based transfer operations place data at the pointer position if possible. Other operations, including keyboard-based transfer, generally place the data at the location cursor.

Required ❑ ❑ ❑ **p:** Your application supports the use of mouse button 1 to perform drag-and-drop operations.

In Motif 1.2, drag and drop is typically performed using button 2 on a three-button mouse (BTransfer). However, in the Common Desktop Environment environment, mouse button 1 (BSelect) should be supported for drag and drop to support mouse usage compatible with other graphical user interface (GUI) environments. A drag can be initiated with either mouse button 1 or mouse button 2.

Required ❑ ❑ ❑ **q:** When button 2 of a three-button mouse is configured to operate as BAdjust, your application does not perform any BTransfer operations when clicking mouse button 2.

On a three-button mouse, button 2 is typically used for the BTransfer (or BSelect) function. However, in a Common Desktop Environment environment, the user can change an environment setting indicating that mouse button 2 should be used for the BAdjust function instead. When this is the case, BAdjust click should not result in the transfer of any data.

y n/a n

Required ❑ ❑ ❑ **r:** BSelect should always initiate a drag if the drag is started on a selected item. The drag starts once the drag threshold has been reached. This is true for text regions, scrolling lists, and other similar elements.

Clipboard Transfer

Required ❑ ❑ ❑ **4-38:** Keyboard-based clipboard selection actions are available in every editable collection in your application.

Clipboard selection actions need to be available from the keyboard.

Required ❑ ❑ ❑ **4-39:** Your application uses the Cut key (or Shift+Delete) and the Cut entry on the Edit menu to cut the selected elements from an editable component to the clipboard.

The Cut key (or Shift+Delete) and the Cut entry on the Edit menu offer a consistent means of cutting the selection to the clipboard from the keyboard.

Required ❑ ❑ ❑ **4-40:** Your application uses the Copy key (or Control+Insert) and the Copy entry on the Edit menu to copy the selected elements to the clipboard.

The Copy key or (Control+Insert) and the Copy entry on the Edit menu offer a consistent means of copying the selection to the clipboard from the keyboard.

Required ❑ ❑ ❑ **4-41:** Your application uses the Paste key (or Shift+Insert) to paste the contents of the clipboard into an editable component.

The Paste key (or Shift+Insert) offers a consistent way of pasting the contents of the clipboard from the keyboard.

Required ❑ ❑ ❑ **4-42:** If Paste or Paste Link is invoked using a component's pop-up menu, the data is pasted at the insertion position of the component. However, if the pop-up menu is popped up over a selection, the selection is first deleted, even if pending delete is disabled, and the pasted data replaces it, if possible.

Popping up a pop-up menu over a selection indicates that a Paste or Paste Link operation should replace the selection.

y n/a n

Required ❑ ❑ ❑ **4-43:** If Paste or Paste Link is invoked from the Edit menu or by a keyboard operation, and the insertion position in the target component is not disjoint from a selection, the pasted data replaces the selection contents if pending delete is enabled.

Pending delete determines whether the selection is deleted when the insertion position is not disjoint from the selection and Paste or Paste Link is invoked from the Edit menu or by a keyboard operation.

Primary Transfer

Required ❑ ❑ ❑ **4-44:** In an editable collection, BTransfer Click, Control+BTransfer Click, Alt, Copy, and Control+Alt+Insert copy the primary selection to the insertion position. (Note that the insertion position is usually different for mouse and keyboard operations.)

These operations provide a convenient way for the user to force a copy operation.

Required ❑ ❑ ❑ **4-45:** In an editable collection, Shift+BTransfer Click, Alt+Cut, and Alt+Shift+Delete move the primary selection to the insertion position. (Note that the insertion position is usually different for mouse and keyboard operations.)

These operations provide a convenient way for the user to force a move operation.

Required ❑ ❑ ❑ **4-46:** In an editable collection, Control+Shift+BTransfer Click places a link to the primary selection at the insertion position.

Control+Shift+BTransfer provides a convenient way for the user to force a link operation.

Required ❑ ❑ ❑ **4-47:** A Primary Move moves the primary selection as well as the elements selected; that is, the element moved to the destination becomes selected as the primary selection. Primary Copy and Primary Link do not select transferred data at the destination.

This rule provides the expected treatment of the selection in a move, copy, and link operation.

Quick Transfer

	y	n/a	n	
Required	❑	❑	❑	**4-48:** All text components support quick transfer. Quick transfer is used to make a temporary selection and then immediately move, copy, or link that selection to the insertion position of the destination component. In text, quick transfer provides a convenient way to move, copy, or link text without disturbing the primary selection.
Required	❑	❑	❑	**4-49:** If a component supports quick transfer, Alt+BTransfer Motion or Control+Alt+BTransfer Motion temporarily selects elements in the specified range and, on release, copies them to the insertion position of the destination component. These operations provide a convenient way to perform a quick copy.
Required	❑	❑	❑	**4-50:** If a component supports quick transfer, Alt+Shift+BTransfer Motion temporarily selects elements in the specified range and, on release, moves them to the insertion position of the destination component. This operation provides a convenient way to perform a quick cut.
Required	❑	❑	❑	**4-51:** If a component supports quick transfer, Control+Alt+Shift+BTransfer Motion temporarily selects elements in the specified range and, on release, places a link to them at the insertion position of the destination component. This operation provides a convenient way to perform a quick link.
Required	❑	❑	❑	**4-52:** Quick transfer does not disturb the primary selection or affect the clipboard, except when the destination of the transfer is within or on the boundaries of the primary selection and pending delete is enabled. In this case, quick transfer deletes the contents of the primary selection, leaving an empty primary selection, before pasting the transferred elements. Quick transfer is a secondary selection mechanism, so it cannot disrupt the primary selection. When the destination of the transfer is in the primary selection, quick transfer replaces the primary selection with the secondary selection.
Required	❑	❑	❑	**4-53:** With quick transfer, the range of the temporary selection is determined by using the same model as when BSelect Motion determines the range of a primary selection.

This rule provides consistency between primary selection and quick transfer operations.

y n/a n

Required ❑ ❑ ❑ **4-54:** If the user drags the pointer out of a scrollable collection while making the temporary selection, autoscrolling is used to scroll the collection in the direction of the pointer. If the user releases BTransfer with the pointer outside of the collection, or if the user presses the Cancel key with BTransfer pressed, the highlighting is removed and a transfer is not performed.

Autoscrolling provides a convenient means of extending a temporary selection to elements outside the viewport of a scrollable collection.

Drag Transfer

Required ❑ ❑ ❑ **4-55:** In a collection that supports selection, Shift+BTransfer Release or Shift+BSelect Release forces a drag move operation. If a move is not possible, the operation fails.

This mechanism offers a convenient way for the user to force a move operation.

Required ❑ ❑ ❑ **4-56:** In a collection that supports selection, Control+BTransfer Release or Shift+BSelect Release forces a drag copy operation. If a copy is not possible, the operation fails.

This mechanism offers a convenient way for the user to force a copy operation.

Required ❑ ❑ ❑ **4-57:** In a collection that supports selection, Control+Shift+BTransfer Release pr Shift+BSelect Release forces a drag link operation. If a link is not possible, the operation fails.

This mechanism offers a convenient way for the user to force a link operation.

Required ❑ ❑ ❑ **4-58:** When a drag move operation moves a selection within the same component, the selection moves along with the elements selected.

In other words, when selected elements are moved with a drag operation, they should stay selected after the move. This mechanism offers a convenient way to move the selection within a component.

	y	n/a	n		
Required	❏	❏	❏	**4-59:**	In text-like collections, initiating a drag within a selected region drags the entire text selection. To be consistent, drag-and-drop actions need to operate on the entire selection.
Required	❏	❏	❏	**4-60:**	In list-like and graphics-like collections, initiating a drag with either BSelect or BTransfer on a selected element drags the entire selection. To be consistent, drag-and-drop actions need to operate on the entire selection.
Required	❏	❏	❏	**4-61:**	In list-like and graphics-like collections, initiating a drag with BTransfer or BSelect on an unselected element drags just that element and leaves the selection unaffected. Unselected elements can be dragged without affecting the selection.
Required	❏	❏	❏	**4-62:**	When a drag is initiated in an unselected region and the pointer is over two possible draggable elements, the drag uses the draggable element highest in the stacking order. This guideline ensures the consistency of drag operations.
Required	❏	❏	❏	**4-63:**	When your application starts a drag operation, the pointer is replaced with a drag icon. A drag icon provides visual feedback that a drag operation is in progress.
Required	❏	❏	❏	**4-64:**	All drag icons used by your application include a source indicator. A source indicator gives a visual representation of the elements being dragged.
Required	❏	❏	❏	**4-65:**	Pressing the Cancel key ends a drag-and-drop operation by canceling the drag in progress. The Cancel key provides a consistent way for the user to cancel a drag operation.
Required	❏	❏	❏	**4-66:**	Releasing BTransfer ends a drag-and-drop operation. Releasing BTransfer offers a consistent means of ending a drag operation.

	y	n/a	n		
Required	❑	❑	❑	**4-67:**	When BTransfer (or BSelect) is released, the drop operation ordinarily occurs at the location of the hot spot of the drag icon pointer and into the highest drop zone in the stacking order. However, if a drop occurs within a selection and pending delete is enabled, the transferred data replaces the contents of the entire selection. This rule provides consistency in the treatment of mouse-based transfer operations.
Required	❑	❑	❑	**4-68:**	After a successful transfer, the data is placed in the drop zone, and any transfer icon used by your application is removed. A transfer icon can be used to represent the type of data being transferred during a drop operation. A successful drop operation results in the transfer of data.
Required	❑	❑	❑	**4-69:**	After a failed transfer, the data remains at the drag source and is not placed in the drop zone. Any transfer icon used by your application is removed. A failed drop operation does not result in the transfer of data.
Recommended	❑	❑	❑	**s:**	In a collection that supports selection, if BTransfer Motion (or BSelect Motion) results in the start of a drag operation, feedback is presented to the user that indicates that a copy, move, or link operation is in progress. Whether the operation is a copy, move, or link depends on the type of object created at the drop zone and whether the source object is removed. Although, typically, an unmodified drag results in a move operation, depending on the location of the source object and the target drop zone, the drag may in fact result in a copy or link operation. For example, dragging an icon representing an attachment to a mail message typically results in a copy of the attachment being created as opposed to the original being removed from the mail message. Any feedback presented should incorporate use of a drag icon that portrays the source object being manipulated.

	y	n/a	n		
Recommended	❑	❑	❑	**t:**	In a collection that supports selection, if Control+BTransfer Motion or Control+BSelect Motion results in the start of a drag operation, feedback is presented to the user that indicates that a copy operation is in progress. The feedback presented should incorporate use of a drag icon that portrays the source object being copied.
Recommended	❑	❑	❑	**u:**	In a collection that supports selection, if Control+Shift+BTransfer Motion or Control+Shift+BSelect Motion results in the start of a drag operation, feedback is presented to the user that indicates that a link operation is in progress. The feedback presented should incorporate use of a drag icon that portrays the source object being linked.
Recommended	❑	❑	❑	**v:**	In a collection that supports copy, move, or link operations that can be performed by dragging, the feedback presented to the user during the drag operation indicates whether a single object or multiple objects are being manipulated. Feedback provided during the drag operation should ensure that the user feels confident that the desired set of objects is being dragged. The drag icon used for multiobject drag operations should integrate the feedback used to indicate whether the operation is a move, copy, or link.
Optional	❑	❑	❑	**w:**	If your application allows the user to paste data into its data pane, it allows the user to drag and drop files from the File Manager into the data pane. The user should be able to drag and drop files into application data panes. The result should be the inclusion of some element of the file, or the display of an error message indicating that the file selected cannot be incorporated into the application's data. Drag transfers that are accepted can result in a number of different responses from your application: 1) the icon image for the file might be inserted at the drop point; 2) the application might perform some activity using the data contained within the file as its input; 3) the data contained within the file might be inserted at the drop point; or 4) the name of the file might be inserted at the drop point.

Component Activation

Basic Activation

	y	n/a	n		
Required	❑	❑	❑	**5-1:**	Your application uses BSelect to activate a button. BSelect, the first mouse button, provides a consistent means of activating a button using the mouse.
Required	❑	❑	❑	**5-2:**	When a button has the focus, your application uses the Select key or Spacebar to activate the button. The Select key and Spacebar provide a uniform way of selecting a button. Selecting a button is equivalent to activating the button.
Required	❑	❑	❑	**5-3:**	When an activatable menu entry has the focus, your application uses the Select, Spacebar, Enter, or Return key to activate the entry. The Select, Spacebar, Enter, and Return keys offer a consistent means of activating a menu entry using the keyboard.
Required	❑	❑	❑	**5-4:**	When BSelect is pressed over a button, the appearance of the button changes to indicate that releasing BSelect will activate the button. If, while BSelect is pressed, the pointer is moved outside of the button, the visual state is restored. If, while BSelect is still pressed, the pointer is moved back inside of the button, the visual state is again changed to indicate the pending activation. If BSelect is pressed and released within a button, the button is activated, regardless of whether the pointer has moved out of the button while it was pressed. The visual state of a button offers a cue to the user about whether the button will be activated when the mouse button is released.
Required	❑	❑	❑	**5-5:**	If a selectable element of a collection is activatable, BSelect Click, the Select key, and Spacebar (except in text) select it. BSelect Click 2 selects and activates it. This rule provides for consistent integration of activation and selection in a collection where elements can be both selected and activated.
Required	❑	❑	❑	**x:**	The time allowed to detect a double click (`*doubleClickTime: 500`) should be no less than 500 milliseconds.

Accelerators

	y	n/a	n		
Required	❑	❑	❑	**5-6:**	If your application uses accelerators, the component with the accelerator displays the accelerator key or key combination following the label of the component.

An accelerator is a key or key combination that invokes the action of some component regardless of the position of the location cursor when the accelerator is pressed. So that the user knows that there is an accelerator associated with a component, the accelerator needs to be displayed.

Required	❑	❑	❑	**5-7:**	If a button with an accelerator is within a primary or secondary window, or within a pull-down menu system from its menu bar, it is activatable whenever the input focus is in the window or the menu bar system. If a button with an accelerator is within a pop-up menu system, it is activatable whenever the focus is in the pop-up menu system or the component with the pop-up menu.

An accelerator must be activatable from the window or component associated with the accelerator.

Mnemonics

Required	❑	❑	❑	**5-8:**	If your application uses mnemonics, the label for the component with the mnemonic contains the character that is its mnemonic. If the label does not naturally contain the character, the mnemonic is placed in parentheses following the label.

A mnemonic is a single character that can be associated with any component that contains a text label. Mnemonics provide a fast way of selecting a component from the keyboard. To let the user know that there is a mnemonic associated with a selection, the mnemonic is underlined in the label of the selection by the toolkit. For a mnemonic to be underlined, the label for a selection needs to contain the mnemonic character. If the label does not contain the mnemonic, putting the mnemonic in parentheses following the label provides visual consistency.

y n/a n

Required ❑ ❑ ❑ **y:** Mnemonic characters must be chosen for ease-of-location within the text of a label. Wherever possible, use the first character of the label. If that is not possible, try to use the last character of the label, or if there is more than one word, the first character of the second word. After that, go through the label from the second character on until a unique mnemonic is found.

Required ❑ ❑ ❑ **5-9:** All mnemonics are case insensitive for activation.

The user must be able to activate a mnemonic by pressing either the lowercase or the uppercase variant of the mnemonic key.

Required ❑ ❑ ❑ **5-10:** When the location cursor is within a menu or a menu bar, pressing the mnemonic key of a component within that menu or menu bar moves the location cursor to the component and activates it. If a mnemonic is used for an option button or for a cascading button in a menu bar, pressing Alt and the mnemonic anywhere in the window or its menus moves the cursor to the component with that mnemonic and activates it.

A mnemonic is generally activatable when the location cursor is within the component that contains the mnemonic. Pressing Alt and the mnemonic provides a way to activate a visible mnemonic when the location cursor is within the window that contains the mnemonic.

Tear-off Activation

Required ❑ ❑ ❑ **5-11:** Activating a tear-off button tears off the menu that contains the button.

A tear-off button is like a push button with the special interaction of tearing off the menu from its cascading button. Tear-off buttons use the same basic activation as other buttons.

Required ❑ ❑ ❑ **5-12:** When a menu with a tear-off button is posted, pressing BTransfer in the tear-off button starts a tear-off action. As long as BTransfer is held, a representation of the menu follows the movement of the pointer. Releasing BTransfer ends the tear-off action by unposting the menu system, creating a new window at the current pointer location that contains the contents of the menu, and giving focus to the new window in explicit pointer mode.

Help Activation

	y	n/a	n		
Required	❑	❑	❑	**5-13:**	Your application uses the Help key on a component to invoke any context-sensitive help for the component or its nearest ancestor with context-sensitive help available. The Help key offers the user a consistent mechanism for invoking context-sensitive help.
Required	❑	❑	❑	**z:**	Your application provides context-sensitive help at all locations. The user should never get a “help not available” message.

Default Activation

Required	❑	❑	❑	**5-14:**	If your application uses default push buttons in a window, the current default push button is highlighted. When the focus is on a push button, its action is the default action, and the push button shows default highlighting. If the default action in a window varies, some push button always has default highlighting, except when there is no current default action. Placing emphasis on the default push button in a dialog box provides the user with a visual cue about the expected reply to the dialog box.
Required	❑	❑	❑	**5-15:**	When focus is in a window with a default action and an activatable menu does not have the focus, the Enter key and Control+Return invoke the default action. If focus is in a component other than multiline text or an activated menu, Return also invokes the default action. These actions have no other effect on the component with the focus, unless the default action has some effect on that component. These rules ensure that the means of invoking a default action are consistent across applications.

y n/a n

Required ❑ ❑ ❑ **5-16:** Except in the middle of a button motion operation, pressing the Cancel key anywhere in a dialog box is equivalent to activating the Cancel push button in the dialog box.

The Cancel key provides a uniform means of canceling dialog box from the keyboard.

Expert Activation

Required ❑ ❑ ❑ **5-17:** If your application supports expert activation, expert actions exist only as shortcuts to application features that are available through another mechanism.

Expert activation, using mouse double-clicking on buttons, provides a convenient way for experienced users to perform certain tasks quickly. However, new users and keyboard-only users need to be able to perform the same tasks.

Required ❑ ❑ ❑ **5-18:** When the focus is on a button used for expert activation, no default action is available, unless the default and expert actions are the same.

This rule minimizes possible confusion between default and expert activation.

Required ❑ ❑ ❑ **5-19:** If a component with an expert action is selectable, activating the expert action first selects the component and then performs the expert action.

A user needs to be able to select a component, even if it has an expert action associated with it.

Previewing and Autorepeat

Required ❑ ❑ ❑ **5-20:** If your application supports activation preview using BSelect, the previewing information is removed when the user releases BSelect.

Activation preview presents the user with additional information that describes the effect of activating a button. This information cannot interfere with the normal operation of the application.

Cancel Activation

y n/a n

Required ❑ ❑ ❑ **5-21:** Pressing the Cancel key stops current interaction in the following contexts:

- During a mouse-based selection or drag operation, it cancels the operation.
- During a mouse-based scrolling operation, it cancels the scrolling action and returns the system to its state prior to the start of the scrolling operation.
- Anywhere in a dialog box that has a Cancel push button, it is equivalent to activating that push button, except during a mouse-based selection or drag operation.
- In a pull-down menu, it either dismisses the menu and moves the location cursor to the cascading button used to pull it down, or unposts the entire menu system. In a pop-up menu, option menu, tear-off menu, or menu bar, it unposts the menu system.
- When the focus is in a torn off menu window, it closes the torn off menu window.

These guidelines for the Cancel key ensure the consistent operation of the key across applications.

Window Management

Window Support

This section corresponds to section 7.2 of the *OSF/Motif Style Guide, Revision 1.2*. The different window types are discussed throughout the *OSF/Motif Style Guide, Revision 1.2* and this book. In particular, see Chapter , "Application Design Principles."

Required ❑ ❑ ❑ **aa:** Application windows should be clearly distinguishable as primary or secondary windows based on appearance and behavior.

Primary Window:

- Primary window decoration (see "Window Decorations" on page 188).

- Primary window management (see “Window Management Actions” on page 191).
- Window stacking, workspace placement, and minimization can be independent of other primary windows.

Secondary Window:

- Secondary window decoration (see “Window Decorations” on page 188).
- Secondary window management (see “Window Management Actions” on page 191).
- Window stacking, workspace placement, and minimization tied to the associated primary window.

Window Decorations

y n/a n

Required ❑ ❑ ❑ **ab:** Windows that support particular window management functionality must request the corresponding window decoration (for example, a window that can be minimized should request the minimize button).

Required ❑ ❑ ❑ **ac:** Windows that support any window management functionality (move, resize, minimize, maximize, close, and others) must have a window menu with items for that functionality.

Required ❑ ❑ ❑ **ad:** Follow Common Desktop Environment window decoration conventions, as shown in Table 10-1.

Table 10-1 Common Desktop Environment Window Decoration Conventions

	Border	Title	Menu	Min	Max	Resize
Primary Window:						
Default	Yes	Yes	Yes	Yes	Yes[1]	Yes[1]
Front Panel	Yes[2]	No	Yes[2]	Yes	No	No
Secondary Window:						
Default	Yes	Yes	Yes	No	No[3]	No[3]
Front Panel	No	Yes	Yes	No	No	No

[1]Decorations for resize and maximize should be provided for primary windows if appropriate.

[2]The Front Panel has custom visuals for the window decorations.

[3]Secondary windows should be designed such that resizing and maximization are not necessary or appropriate. If a secondary window must be resizable and maximizable, the associated decorations should be displayed.

	y	n/a	n		
Required	❑	❑	❑	**ae:**	Follow Common Desktop Environment window menu conventions. Items should appear in the window menu if they are applicable to the window or its minimized window icon.

- Restore (R)
- Move (M)
- Size (S)
- Minimize (n)
- Maximize (x)
- Lower (L)
- Occupy Workspace ... (O)
- Occupy All Workspaces (A)
- Unoccupy Workspace (U)
- Close (C) Alt+F4

Optional	❑	❑	❑	**af:**	Applications should not add items to the window menu. If an extraordinary requirement has an application add items to the window menu, the items should be appended to the end of the menu with a separator between Close and the application items.
Optional	❑	❑	❑	**ag:**	Accelerators, aside from Alt+F4 for Close, should not be used in the window menu (to minimize conflict with other uses of the Alt key for application accelerators, localization, and others).

Window Navigation

This section corresponds to section 7. 4 of *OSF/Motif Style Guide, Revision 1.2*. There are no checklist items for application developers.

Icons

Optional	❑	❑	❑	**ah:**	Applications should provide unique window icons for their primary windows. The window icon image should have a similar appearance to the associated file or Front Panel icon image.

	y	n/a	n		
Optional	❑	❑	❑	**ai:**	The window icon label should contain the same text as the title of the corresponding primary window, or an abbreviated form of it. Refer to "Layout" on page 193 for window title guidelines.
Optional	❑	❑	❑	**aj:**	The window icon image should have a similar appearance to the associated file or Front Panel icon image. Refer to "Design Philosophy and Helpful Hints" on page 49.

Application Window Management

Window Placement

Recommended	❑	❑	❑	**ak:**	Applications should not require or force windows or window icons to be positioned at a particular screen location.
Recommended	❑	❑	❑	**al:**	A secondary window is placed by the application relative to the associated primary window. It should be placed close to, but not obscuring, the component that caused it to be displayed and the information that is necessary to interact with the dialog box.

Some suggestions are given in section 6.2.4.3, "Determining Dialog Box Location and Size," of the *OSF/Motif Style Guide, Revision 1.2*. Additional or modified recommendations include:

Optional	❑	❑	❑	**am:**	If a dialog box does not relate to specific items in the underlying window, it should be placed below the menu bar (if there is one) and centered (horizontally) over the work area.
Recommended	❑	❑	❑	**an:**	If a secondary window is allowed to be stacked below its associated primary window (not constrained to stay on top of the primary window), it should be placed such that it is not completely covered by the primary window. This recommendation takes precedence over other placement recommendations.
Recommended	❑	❑	❑	**ao:**	If a menu or dialog box is already on display, reinvoking the command that caused it to be displayed automatically brings that window or menu to the front of the window stack without changing its position on the screen.

Window (Document) Clustering

y n/a n

Optional ❑ ❑ ❑ **ap:** Windows that are closely related in supporting a particular task should be placed in a window cluster. Secondary windows are automatically placed in a window cluster with the associated primary window. Windows in a window cluster are stacked together, minimized or normalized together, and kept in the same workspace.

Note – Currently the only mechanism for forming a window cluster that is supported by the Window Manager is to indicate a primary-secondary relationship.

Window Management Actions

Required ❑ ❑ ❑ **aq:** Windows should follow Common Desktop Environment window management functionality conventions, as shown in Table 10-2.

Table 10-2 Common Desktop Environment Window Management Conventions

	Close	**Move**	**Lower**	**Min**	**Max**	**Resize**
Primary Window:						
Default	Yes	Yes	Yes	Yes	Yes[1]	Yes[1]
Front Panel	No	Yes	Yes	Yes	No	No
Secondary Window:						
Default	Yes	Yes	Yes	No	No[2]	No[2]
Subpanel	Yes	Yes	Yes	No	No	No

[1]Resize and maximize functionality should be provided for primary windows if appropriate.

[2]Secondary windows can contain the Maximum and Resize window manager functions, if appropriate.

Required ❑ ❑ ❑ **ar:** Windows that support particular window management functionality should request corresponding window decoration (for example, a window that can be minimized should request the minimize button).

Required ❑ ❑ ❑ **as:** Windows that have form factor constraints need to set Window Manager hints for minimum size, maximum size, aspect ratio, and resize increment as appropriate.

Recommended ❑ ❑ ❑ **at:** Maximizing a window should show more content (objects or controls) if appropriate (as opposed to scaling up the sizes of objects and controls).

	y	n/a	n		
Required	❑	❑	❑	**au:**	Windows that have Close or Exit functionality need to support the window management protocol for Close if there is a window menu. In the case of dialog boxes, the Close item on the window menu corresponds to the Cancel functionality or dialog box dismissal with no further action taken.
Recommended	❑	❑	❑	**av:**	When your application creates a new window, it should come up in the user's current workspace and only occupy that single workspace.
Recommended	❑	❑	❑	**aw:**	Application windows that are related to a particular task should move together between workspaces.

Session Management Support

Required	❑	❑	❑	**ax:**	Applications should support *Interclient Communications Conventions Manual* (ICCCM) mechanisms for session management of their primary windows and key properties.
Required	❑	❑	❑	**ay:**	Applications should support ICCCM mechanisms for session management of all associated windows (that is, secondary windows that may include help windows).
Optional	❑	❑	❑	**az:**	Applications should accept messages from the Common Desktop Environment Session Manager that inform them the user is logging out and should save their state at that time.
Optional	❑	❑	❑	**ba:**	Applications that have a single primary window that is open at the time the user logs out should restore the primary window, in the workspace last occupied, when the user logs in again.
Optional	❑	❑	❑	**bb:**	Save user context wherever possible. For example, applications that support the editing of files should save the state of the file at logout and should restore the file in the application window when users log in again.
Optional	❑	❑	❑	**bc:**	Applications that have multiple primary windows that are open at the time the user logs out should restore all primary windows, in their respective workspaces, when the user logs in again.

Application Design Principles

Layout

Main Window

	y	n/a	n		
Required	❑	❑	❑	**6-1:**	Your application should be composed of at least one main window. A main window contains a client area and, optionally, a menu bar, a command area, a message area, and scroll bars. The client area contains the framework of the application. The use of a main window ensures interapplication consistency.
Required	❑	❑	❑	**bd:**	The default size of the application's main window must be large enough to accommodate a typical amount of data, but should not fill the entire physical display size to minimize visual conflicts with other applications. Each application potentially must share the display with other applications. The default window size should not take up all the available screen space.
Required	❑	❑	❑	**be:**	Resize corners should be included in any main window that incorporates a scrolling data pane or list. Resize corners should be included in any main window that incorporates a scrolling data pane or list. Any changes to the overall size of the window should result in a corresponding increase or decrease in the size of the scrollable portion. Additionally, your application might reorganize elements within the window based on the increased or decreased amount of space (for example, it might reorganize a row of buttons into two rows).
Required	❑	❑	❑	**6-2:**	If your application has multiple main windows that serve the same primary function, each window closes and iconifies separately. For example, a text editor might allow the user to edit multiple documents, each in its own main window. Each window is then treated as a separate application and can be closed or iconified when it is not being used.

y n/a n

Required ❑ ❑ ❑ **6-3:** If your application has multiple main windows that serve different primary functions, each window should be able to iconify independently of the other windows.

For example, a debugger might provide separate main windows for editing source code, examining data values, and viewing results. Each window can be iconified when it is not being used, but it is up to the application to decide whether each window closes separately or whether closing one window closes the entire application.

Window Titles

Optional ❑ ❑ ❑ **bf:** The title of your primary window (the main window your application displays to the user) should be the name of your application.

Note that this does not have to be the actual name of the executable invoked by the user.

Carefully consider how the title you choose for your primary window works when it is used in icons and pop-up windows. If the name of the pop-up window is too long, you may remove the application title; however, without the title, users might have difficulty telling which pop-up window belongs with the originating primary window.

Optional ❑ ❑ ❑ **bg:** Use initial capital letters for each word in the title (in languages that support capitalization).

Optional ❑ ❑ ❑ **bh:** Follow the application name for each property window, as a minimum, with the title Properties and the name of the object it affects.

Optional ❑ ❑ ❑ **bi:** Begin the title of each pop-up window with the application title followed by a colon, then the title of the pop-up window. The colon should have a space both before and after it for readibility.

Pop-up windows should always indicate which primary window they are associated with (which primary window invoked that pop-up).

Optional ❑ ❑ ❑ **bj:** Use a hyphen to denote the current file name, when the application has files that can be loaded or saved. The hyphen should have a space before and after it. Only the base name of the file should be displayed, not the entire path.

The hyphen is used to denote specific instances of a window or data. The colon serves to delimit general categories or commands. For example, a file manager might have the following title for a Properties dialog box:

File Manager : Properties - myfile

	y	n/a	n		
Recommended	❑	❑	❑	**bk:**	Follow the application name for each command window with the same title that is on the window button or window item users choose to display that window.
Optional	❑	❑	❑	**bl:**	In the case of multiple primary windows, include the application name at the beginning of each window title, and add a name that uniquely identifies that primary window. No separator should be provided for these names (for example, Calendar Manager Multibrowse, Catalog Search, Admintool Databases).
Optional	❑	❑	❑	**bm:**	An abbreviated name for the application may be used on other windows, so long as it is done on all windows.

Menu Bar

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Required	❑	❑	❑	**6-4:**	If your application has a menu bar, it is a horizontal bar at the top edge of the application, just below the title area of the window frame. A menu bar organizes the most common features of an application. It contains a list of menu topics in cascading buttons; each button is associated with a distinct pull-down menu containing commands that are grouped by common functionality. The use of a menu bar yields consistency across applications.
Required	❑	❑	❑	**6-5:**	The menu bar for your application contains only cascading buttons. When other buttons are included as topics in a menu bar, they inhibit menu browsing.
				6-6:	This item has been deleted. It is replaced by the following guideline.

	y	n/a	n		
Recommended	❑	❑	❑	**bn:**	There are several common menu operations that should be considered "standard". The standard menu bar entries are File, Edit, View, Options and Help. If your application provides that functionality to the user, it should be included in the menu bar under the appropriate name.The contents of these menu entries are discussed below in more detail.

Standard menu bar entries should be presented in the following order:

File Edit View Options Help

You should exclude from your menu bar any item shown in the preceding text if your application does not support the associated function. For example, if your application does not support the ability to display its data in different views, then you should not include a View menu.

You may add application-specific menus in between any of the standard menu items, with the following exceptions:

- The File menu, if present, is located in the first menu position on the left.

- The Help menu is located on the far right position.

- If File and Edit are present, they should be next to each other.

For example, your application may have:

File Edit <category1> <category2> View Options <category3> Help

Recommended	❑	❑	❑	**bo:**	Applications that are not file-oriented in nature (or that manage files transparently, not exposing this activity to the user) should replace the File menu with one or more application-specific menus.

Replacing the File menu:

Replacement1: <app-label> Selected

Replacement2: <app-label><obj-type>

Replacement3: <obj-type>

You may use Replacement1 if your application has more than one object type. Items on <app-label> would be used for global actions that are not specific to an object type. The items in Selected are actions that pertain to objects that are currently selected, and may change depending on what objects are selected. If nothing is selected, this menu should have a single item that says (none selected). If an item is selected, but there are no items that apply to that object, this menu should have a single item that says (none).

You may use Replacement2 if your application has a single object type. Actions that are global to the application are on <app-label>, and actions that are specific to the object type are on <obj-type>.

You may use Replacement3 if your application has a single object type, and does not require an <app-label> menu. For example, a Print Manager might contain a Printer menu.

All other menubar guidelines that apply to File-oriented applications also apply to non-File-oriented applications. Thus, the following menubar would be valid:

<app-label> Selected Edit <category1> View <category2> Help

Applications that are complex or are extremely domain-specific (for example, an application for medical imaging and diagnosis of cat scan data) may require other approaches to their menu bar design. For example,

<app-label><category1><category2> Selected Edit <object-type> Options Help

y n/a n

Recommended ❑ ❑ ❑ **bp:** Exit or Close should be located on the first (leftmost) menu of your menubar.

File Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

y n/a n

Required ❑ ❑ ❑ **bq:** If the user chooses Exit, or in any other manner indicates that the application should be terminated, but there are changes to the current file that have not been saved, your application displays a dialog box asking whether the changes should be saved before exiting.

The user must always be given the opportunity to explicitly state whether unsaved changes should be saved or discarded. A dialog box similar to the one described should also be displayed if the user chooses the Open from the File menu, but has not saved changes to the current file.

Required ❑ ❑ ❑ **6-7:** If your application uses a File menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application.

Required ❑ ❑ ❑ **New** Creates a new file. If the current client area will be used to display the new file, your application clears the existing data from the client area. If changes made to the current file will be lost, your application displays a dialog box, asking the user about saving changes. The mnemonic is N.

Required ❑ ❑ ❑ **Open...** Opens an existing file by prompting the user for a file name with a dialog box. If changes made to the current file will be lost, your application displays a dialog box asking the user about saving changes. The mnemonic is O.

Required ❑ ❑ ❑ **Save** Saves the currently opened file without removing the existing contents of the client area. If the file has no name, your application displays a dialog box, prompting the user to enter a file name. The mnemonic is S.

Required ❑ ❑ ❑ **Save As...** Saves the currently opened file under a new name by prompting the user for a file name with a dialog box. If the user tries to save the file using an existing name, your application displays a dialog box that warns the user about a possible loss of data. Does not remove the existing contents of the client area. The mnemonic is A.

	y	n/a	n		
Recommended	❑	❑	❑	**Print**	Schedules a file for printing. If your application needs specific information to print, it displays a dialog box, requesting the information from the user. In this case, the menu entry is followed by an ellipsis (Print...). The mnemonic is P.
Recommended	❑	❑	❑	**Close**	Closes the current primary window and its associated secondary windows. If your application uses only a single primary window or multiple dependent primary windows, this action is not supplied. The mnemonic is C.
Required	❑	❑	❑	**Exit**	Ends the current application and all windows associated with it. If changes made to the current file will be lost, your application displays a dialog box, asking the user about saving changes. The mnemonic is X.

The use of a File menu with these common file operations yields consistency across applications.

<Object-type> / Selected Menu Contents

Recommended ❑ ❑ ❑ **br:** If your application uses an <object-type> menu or a Selected menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application. Items should be presented to the user in the order listed below.

The <object-type> menu contains controls that allow the user to create instances of the object-type. Both the <object-type> and Selected menus allow the user to manipulate object instances. Additional items should be added to the <object-type> or Selected menus if they relate solely to the manipulation of objects managed by the application (as opposed to more generic services that the application might provide).

Recommended	❑	❑	❑	**New...**	Creates a new instance of the object-type. If appropriate, a dialog box is presented allowing the user to specify the values for settings associated with that object.

	y	n/a	n		
Optional	❑	❑	❑	**Move To**...	Allows the user to move the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
Optional	❑	❑	❑	**Copy To...**	Allows the user to copy the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
Optional	❑	❑	❑	**Put in Workspace**	Allows the user to put a link for the object onto the Common Desktop Environment in the current workspace.

Any of the preceding three menu choices should be provided only if the objects managed by your application are able to reside as separate entities outside of your application's main window. For example, a printer object created by a printer management application might be able to be placed in a Folder window and function as an application unto itself. Your application should also support drag and drop as a method for performing any of these actions.

	y	n/a	n		
Optional	❑	❑	❑	**Delete**	Removes the selected objects. A confirmation dialog box should be presented to the user before the object is actually deleted.
Recommended	❑	❑	❑	**Properties**	Displays a Properties window that shows the current values for settings associated with the selected object.
Recommended	❑	❑	❑	**<Default Action>**	This choice should enact the default action for the selected object. “Open” is a typical default.

Edit Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

---

	y	n/a	n			
Required	❑	❑	❑	**6-8:**	If your application uses an Edit menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application:	
Optional	❑	❑	❑		**Undo**	Reverses the most recently executed action. The mnemonic is U.
Optional	❑	❑	❑		**Cut**	Removes the selected portion of data from the client area and puts it on the clipboard. The mnemonic is T.
Optional	❑	❑	❑		**Copy**	Copies the selected portion of data from the client area and puts it on the clipboard. The mnemonic is C.
Optional	❑	❑	❑		**Copy Link**	Copies a link of the selected portion of data from the client area and puts it on the clipboard. The mnemonic is K.
Optional	❑	❑	❑		**Paste**	Pastes the contents of the clipboard into the client area. The mnemonic is P.
Optional	❑	❑	❑		**Paste Link**	Pastes a link of the data represented by the contents of the clipboard into the client area. The mnemonic is L.
Optional	❑	❑	❑		**Clear**	Removes a selected portion of data from the client area without copying it to the clipboard and does not compress the remaining data. The mnemonic is E.
Optional	❑	❑	❑		**Delete**	Removes a selected portion of data from the client area without copying it to the clipboard. The mnemonic is D.
Optional	❑	❑	❑		**Select All**	Sets the primary selection to be all the elements in a component of the client area.
Optional	❑	❑	❑		**Deselect All**	Removes from the primary selection all the elements in a component of the client area.
Optional	❑	❑	❑		**Select Pasted**	Sets the primary selection to the last element or elements pasted into a component of the client area.

	y	n/a	n			
Optional	❑	❑	❑		**Reselect**	Sets the primary selection to the last selected element or elements in a component of the client area. This action is available only in components that do not support persistent selections and only when the current selection is empty.
Optional	❑	❑	❑		**Promote**	Promotes to the primary selection the current selection of a component of the client area. This action is available only for components that support persistent selections.

The use of an Edit menu with these common editing operations yields consistency across applications.

Recommended ❑ ❑ ❑ **bs:** If your application does not provide an <object-type> or Selected menu, but allows the user to select data within the window and manage settings for the selected data, then it provides a Properties ... choice as the last item in the Edit menu.

Required ❑ ❑ ❑ **6-9:** This item has been deleted.

View Menu

Recommended ❑ ❑ ❑ **bt:** If your application provides a View menu, it only contains functions that affect the way the current data is presented. It does not contain any option that alters the data itself.

Options Menu

Recommended ❑ ❑ ❑ **bu:** If your application has global settings that control the way the application behaves, it provides an Options menu from which these can be set.

Help Menu Contents

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

---

	y	n/a	n			
Recommended	❑	❑	❑	**bv:**	If your application includes a Help menu, it contains the following set of choices, with the specified functionality, when the actions are actually supported by your application. The Help choices included here supercede those listed for Motif 1.2.	
Required	❑	❑	❑		**Overview**	Provides general information about the window from which help was accessed or about the application overall. The mnemonic is V. Place a separator after.
Optional	❑	❑	❑		**Index**	Provides an index listing topics for all help information available for your application. The mnemonic is I.
Recommended	❑	❑	❑		**Table of Contents**	Provides a table of contents listing topics for all help information available for your application. The mnemonic is C.
Recommended	❑	❑	❑		**Tasks**	Provides access to help information indicating how to perform different tasks using your application. The mnemonic is T.
Recommended	❑	❑	❑		**Reference**	Provides access to reference information. The mnemonic is R.
Optional	❑	❑	❑		**Tutorial**	Provides access to your application's tutorial. The mnemonic is L.
Optional	❑	❑	❑		**Keyboard**	Provides information about your application's use of function keys, mnemonics, and keyboard accelerators. Also provides information on general Common Desktop Environment use of such keys. The mnemonic is K.
Optional	❑	❑	❑		**Mouse**	Provides information about using a mouse with your application. The mnemonic is M.

	y	n/a	n		
Optional	❑	❑	❑	**Mouse and Keyboard**	Provides information about your application's use of function keys, mnemonics, keyboard accelerators, and using a mouse with your application. Also provides information on general Common Desktop Environment use of such keys. The mnemonic is M. Use rather than separate mouse and keyboard choices if this information is best presented together.
Recommended	❑	❑	❑	**On Item**	Initiates context-sensitive help by changing the shape of the pointer to the question mark pointer. When the user moves the pointer to a component and presses BSelect, any available context-sensitive help for the component is presented. The mnemonic is O. Set off with separators on both sides.
Required	❑	❑	❑	**Using Help**	Provides information on how to use the Common Desktop Environment Help Viewer. The mnemonic is U. Set off with separators on both sides.
Required	❑	❑	❑	**About** *applicationname*	Displays a dialog box indicating, minimally, the name and version of your application, and displays its icon or some other signature graphic for your application. The mnemonic is A.

6-10: This item has been deleted. It is replaced by item **bv:**.

Attachment Menu Contents

Recommended ❑ ❑ ❑ **bw:** If your application uses an attachment menu, it contains the following choices, with the specified functionality, when the actions are actually supported by your application.

Recommended	❑	❑	❑	**Add File...**	Selects files and other items to be attached. A file selection box is displayed allowing the user to select the desired files to attach. The default button in the file selection box is Attach.

	y	n/a	n		
Recommended	❑	❑	❑	**Save As...**	Saves the currently selected attachments. The user is prompted with a file selection dialog box for indicating where in the file system the attachments are to be saved. When multiple attachments are selected, the name field is inactive and the current names of the attachments are used as the name of the new file. This menu item is active only when one or more attachments are selected.
Recommended	❑	❑	❑	**Rename...**	Renames the attachment icon. The application should provide in-line renaming of attachment icons, such as File Manager uses. If the application cannot provide in-line renaming, then Rename allows the user to rename an attachment by displaying a dialog box, requesting the name from the user. This menu item is active only when a single attachment is selected. It is not active when multiple attachments are selected.
Recommended	❑	❑	❑	**Delete**	Deletes attachments from the attachment list. This menu item is active only when an attachment is selected.
Recommended	❑	❑	❑	**Select All**	Selects all the attachments in the attachment list.

Pop-up Menus

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

Recommended ❑ ❑ ❑ **bx:** If your application provides functions that apply to a data pane and not any specific element therein, then a pop-up menu is provided that contains the frequently used data pane functions and is accessible by pressing BMenu when the mouse pointer is over the background of the pane or a nonselectable element within the pane.

Recommended ❑ ❑ ❑ **by:** Your application should provide a pop-up menu for any element that is selectable within its data pane.

Pop-up menus provide access to frequently used functions and should be used pervasively throughout the Common Desktop Environment environment. A pop-up menu may contain a collection of options that appear in different menus available from the menu bar. For example, it may contain items from both the File and Edit menus.

	y	n/a	n		
Recommended	❑	❑	❑	**bz:**	When a pop-up menu is displayed over an unselected object, any action selected from the pop-up menu applies to that object only, and not to any other objects that might currently be selected.

The preceding helps to protect the user from inadvertently applying an action to objects that the user may not realize are currently selected. Pressing the menu button invokes a pop-up menu pertinent to the object under the mouse cursor whether it is selected to not; if the object under the mouse cursor and other objects are selected, the pop-up menu is pertinent to the selected set.

Recommended ❑ ❑ ❑ **ca:** Every pop-up menu in your application has a title that indicates the function the menu performs or the element on which it operates.

Recommended ❑ ❑ ❑ **cb:** The functions accessible from within your application's pop-up menus are also accessible from buttons displayed within the window or menus accessed through the menu bar.

Because pop-up menus are hidden, they should only provide redundant access to functions available from more visible controls within the application's windows.

Optional ❑ ❑ ❑ **6-11:** If your application uses any of the common pop-up menu actions, the actions function according to the following specifications. See item **cc:** for supplemental guidelines.

Optional	❑	❑	❑	**Properties**	Displays a Properties dialog box that the user can use to set the properties of the component.
Optional	❑	❑	❑	**Undo**	Reverses the most recently executed action.
Optional	❑	❑	❑	**Primary Move**	Moves the contents of the primary selection to the component. This action is available only in editable components.
Optional	❑	❑	❑	**Primary Copy**	Copies the contents of the primary selection to the component. This action is available only in editable components.

	y	n/a	n		
Optional	❑	❑	❑	**Primary Link**	Places a link to the primary selection in the component. This action is available only in editable components.
Optional	❑	❑	❑	**Cut**	Cuts elements to the clipboard. If the menu is popped up in a selection, cuts the entire selection to the clipboard.
Optional	❑	❑	❑	**Copy**	Copies elements to the clipboard. If the menu is popped up in a selection, this action copies the entire selection to the clipboard.
Optional	❑	❑	❑	**Copy Link**	Copies a link of elements to the clipboard. If the menu is popped up in a selection, copies a link to the entire selection to the clipboard.
Optional	❑	❑	❑	**Paste**	Pastes the contents of the clipboard to the component. This action is available only in editable components.
Optional	❑	❑	❑	**Paste Link**	Pastes a link of the contents of the clipboard to the component. This action is available only in editable components.
Optional	❑	❑	❑	**Clear**	Removes a selected portion of data from the client area without copying it to the clipboard. If the menu is popped up in a selection, deletes the selection.
Optional	❑	❑	❑	**Delete**	Removes a selected portion of data from the client area without copying it to the clipboard. If the menu is popped up in a selection, deletes the selection.
Optional	❑	❑	❑	**Select All**	Sets the primary selection to be all of the elements in the collection with the pop-up menu.
Optional	❑	❑	❑	**Deselect All**	Deselects the current selection in the collection with the pop-up menu.
Optional	❑	❑	❑	**Select Pasted**	Sets the primary selection to be the last element or elements pasted into the collection with the pop-up menu.

	y	n/a	n			
Optional	❑	❑	❑		**Reselect**	Sets the primary selection to be the last selected element or elements in the component with the pop-up menu. This action is available only in components that do not support persistent selections and only when the current selection is empty.
Optional	❑	❑	❑		**Promote**	Promotes the current selection to the primary selection. It is available only in components that support persistent selections.

The use of pop-up menus with these common actions yields consistency across applications.

Recommended	❑	❑	❑	**cc:**	Pop-up menus for selectable objects contain the following set of choices, with the specified functionality, when the actions are actually supported by your application. These guidelines supplement item **6-11:**.	
Optional	❑	❑	❑		**Move To ...**	Allows the user to move the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
Optional	❑	❑	❑		**Copy To ...**	Allows the user to copy the selected objects into a folder. A file selection dialog box is displayed allowing the user to select the desired folder.
Optional	❑	❑	❑		**Put in Workspace**	Allows the user to put a link for the selected objects onto the Common Desktop Environment in the current workspace.
Optional	❑	❑	❑		**Delete**	Deletes the selected object. A confirmation is displayed to the user before actually removing the object.
Recommended	❑	❑	❑		**Properties ...**	Displays a dialog box indicating the current settings for attributes associated with the selected object.
Recommended	❑	❑	❑		**Help ...**	Displays a help window pertaining to objects of the type selected.

y n/a n

Optional ❏ ❏ ❏ **cd:** Choices within your pop-up menus are organized in the following manner:

<choices that manage the object such as Open, Save, or Properties>

----------- separator ----------------

<standard edit menu choices such as Cut, Copy, and Paste>

----------- separator ----------------

<other choices>

Required ❏ ❏ ❏ **6-12:** When a pop-up menu is popped up in the context of a selection, any action that acts on elements acts on the entire selection.

In the context of a selection, pop-up menu actions affect the entire selection.

Dialog Boxes

Required ❏ ❏ ❏ **6-13:** Information dialog boxes do not interrupt the user's interaction with your application.

An information dialog box conveys information to the user that does not require immediate attention, so it does not need to be modal.

Menu Design

Recommended ❏ ❏ ❏ **ce:** If the selection of a menu item will result in the user being queried for more information, such as through the posting of a file selection dialog, the menu item should be followed by an ellipsis ("..."). This requirement does not apply to menu items that will result in a simple warning or confirmation dialog being displayed.

The use of an ellipsis helps set the user's expectation for the behavior of the interface. When they select an item without an ellipsis, they know that they can expect an immediate result.

Recommended ❏ ❏ ❏ **cf:** Menus accessed from within your application contain at least two menu items.

No menu should contain only one item. If your application provides a menu with only one item, you should look at moving that item into another menu or making it a button within the window. The longer the menu, the more effort is needed for the user to access choices near the bottom. If your menu has a lot of choices, break it up into two or more menus, or group some items into submenus.

	y	n/a	n		
Optional	❑	❑	❑	**cg:**	Submenus accessed from within your application contain at least three menu items.

Submenus may be used to group like items into a single secondary cascading menu where putting the items into the primary cascading menu would make it too long. However, if your submenu contains only two options, you should strongly look at removing the secondary cascading menu and putting the options into the primary cascading menu since it takes more effort for the user to access options located in a submenu.

Recommended ❑ ❑ ❑ **ch:** No menu in your application contains more than 15 choices.

The longer the menu the more effort is needed for the user to access choices near the bottom. If your menu has a lot of choices, you should look at breaking it up into two or more menus, or grouping some items into submenus.

Optional ❑ ❑ ❑ **ci:** If your application contains a menu that is expected to be accessed frequently, then a tear-off menu option is provided in that menu.

The user should be able to tear-off frequently accessed menus so that these can remain posted on the desktop as the user uses your application.

Optional ❑ ❑ ❑ **cj:** Provide keyboard accelerators where appropriate.

If specific menu items within a menu are expected to be used frequently, not the menu as a whole, then your application provides keyboard accelerators for these items and displays the keyboard accelerators in the associated menu to the right of the item to which they relate.

Recommended ❑ ❑ ❑ **ck:** The labels used for items in the menu bar do not appear as options within the menus themselves.

The names of items in the menu bar serve as titles for the options the menu contains. The name of the menu bar item should provide a term that accurately describes the concept of the category relating all of the menu items and should not be used as the name of any item within the menu itself.

	y	n/a	n		
Required	❑	❑	❑	**cl:**	Any menu choice that is not currently an appropriate selection is dimmed (insensitive).
					Dimmed controls cannot be activated by the user and should appear only when the inactive state is short-term (that is, there is something the user can do within the application or the desktop environment to make the control become active). When the control is persistently inactive (because of the current configuration of the application or system, or a particular set of companion software is not currently installed), the control should be removed rather than dimmed.
Recommended	❑	❑	❑	**cm:**	If a menu item is used to indicate a selection state, use a checkbox or radio button to indicate the state of the item. Use a checkbox if a single item is used to represent on or off states, and use radio buttons for multiple adjacent menu items in which only one of the items may be selected.
Required	❑	❑	❑	**cn:**	If radio buttons are used in a menu, use separators between each set of radio buttons and other menu items.
Recommended	❑	❑	❑	**co:**	If a checkbox or radio button is used on a menu item, it should always be shown as either selected or not selected, and should not dissappear when in the unselected state.
Required	❑	❑	❑	**6-14:**	If your application uses a tear-off button in a menu, the tear-off button is the first element in the menu.
					When a tear-off button is activated, the menu changes into a dialog box. The tear-off button needs to be the first item in the menu so that the entire contents of the menu are torn off.
Required	❑	❑	❑	**6-15:**	All menus are wide enough to accommodate their widest elements.
					The ability to see the full label of each menu element allows the user to browse through a menu.

Dialog Box Design

Note – These requirements apply only in a left-to-right language environment in an English-language locale. You must make the appropriate changes for other locales.

y n/a n

Recommended ❑ ❑ ❑ **cp:** The title of dialog boxes used within your application adheres to the conventions listed in Table 10-3.

Table 10-3 Dialog Box Title Conventions

Window Usage	Window Title Format
Message	<app or object name> : <action or situation>
Progress	<app or object name> : <action> in Progress
Action (Command)	<app name> : <action>
Object Properties	<app name> : <object-type> Properties
Application Options	<app name> : <type> Options

Required ❑ ❑ ❑ **cq:** Every dialog box in your application has at least one button that either performs the dialog box action and dismisses it or dismisses the dialog box without taking any action.

6-16: This item has been replaced by item **cr:**.

Recommended ❑ ❑ ❑ **cr:** If your application uses common dialog box actions, the actions have the following specified functionality and labels:

Optional ❑ ❑ ❑ **Yes** Indicates an affirmative response to a question posed in the dialog box.

Optional ❑ ❑ ❑ **No** Indicates a negative response to a question posed in the dialog box.

Optional ❑ ❑ ❑ **OK** Applies any changes made to components in the dialog box and dismisses the dialog box.

Optional ❑ ❑ ❑ **<command>** Applies any changes made to components in the dialog box, performs the action associated with the <command>, and dismisses the dialog box.

Should be used in lieu of OK, Yes, or No as a button label when it provides more meaning to the user as to the action that will be performed when that button is clicked.

	y	n/a	n		
Optional	❑	❑	❑	**Apply**	Applies any changes made to components in the dialog box and does not dismiss it.
Optional	❑	❑	❑	**Retry**	Causes the task in progress to be attempted again.
Optional	❑	❑	❑	**Stop**	Ends the task in progress at the next possible break point.
Optional	❑	❑	❑	**Pause**	Causes the task in progress to pause.
Optional	❑	❑	❑	**Resume**	Causes a task that has paused to resume.
Optional	❑	❑	❑	**Save As Defaults**	Saves the current settings as the default settings that will appear the next time the window is displayed. The settings are not applied to any selected object and the dialog box is not dismissed. A Save As Defaults button should be provided if it is expected that a user would want to use different default values for a set of controls within a dialog box than those that you provide as the factory settings. For example, a Save As Defaults button might be provided in a “New <object-type>” window, allowing the user to indicate that whenever a new instance of that object-type is created, the current values should be displayed as the default settings instead of the values given by the application.
Optional	❑	❑	❑	**Reset**	Cancels any changes that have not yet been applied by your application. The controls within the dialog box are reset to their state since the last time the dialog box action was applied. If no changes have been applied within the current invocation of the dialog box, the controls are reset to the state when the dialog box was first displayed.

	y	n/a	n			
Optional	❑	❑	❑		**Reset to Factory**	Cancels any changes that have not yet been applied. Components in the dialog box are reset to their default state and value as specified by the vendor that delivered the application (that is, the controls are restored to the original factory settings).
Optional	❑	❑	❑		**Cancel**	Dismisses the dialog box without performing any actions not yet applied.
Recommended	❑	❑	❑		**Help**	Provides help for the dialog box.

Recommended ❑ ❑ ❑ **cs:** Any visible control that is not currently active or whose setting is currently invalid is dimmed.

Dimmed controls cannot be activated by the user and should appear only when the inactive state is short-term (that is, there is something the user can do within the application or the desktop environment to make the control become active). When the control is persistently inactive (because of the current configuration of the application or system, or a particular set of companion software is not currently installed), the control should be removed rather than dimmed.

Optional ❑ ❑ ❑ **ct:** Keep the size of your dialog boxes to a minimum. Remember that on low-resolution displays, dialogs may take up most of the screen real estate, and may even run off the edge of the screen if not designed correctly.

Optional ❑ ❑ ❑ **cu:** Avoid complexity in your dialog boxes. If your dialog box must support many functions, consider using an expandable dialog box (see “Expandable Windows” on page 228), or use more than one dialog in a nested fashion.

Optional ❑ ❑ ❑ **cv:** Avoid the use of resize handles in your dialog box. However, you may use resize handles when resizing is useful in allowing users to see more information; for example, when your dialog contains a scrolling list that is likely to be quite long, and users will frequently need to search the list.

Optional ❑ ❑ ❑ **cw:** Every dialog box in your application has exactly one default button that is activated when the Return key is pressed.

The default button should be associated with the most likely response from the user and should not be potentially destructive or irreversible. Some applications may have dialog boxes that do not reveal a default button until a specific set of fields has been filled out or otherwise manipulated.

y n/a n

Optional ❑ ❑ ❑ **cx:** If a dialog box displayed by your application has controls that are considered to be advanced features, use an expandable dialog box, or use a multiple page dialog box that provides a <category> option menu that allows a user to navigate to each page.

Controls that relate to advanced features should not be displayed with the set of options initially displayed to the user. The typical user should be presented with only those options that are necessary to use the basic functionality of the application. Users looking to access advanced functionality within the dialog box may use the <Category> option button (see Figure 7-1). If the number of advanced controls is few, or the settings for these controls are highly related to the settings of basic controls displayed in the dialog box (that is, the settings of the advanced controls change when the user changes settings for basic controls), you might choose to provide an expandable dialog box (see the section on Expandable Windows and Dialog Boxes).

Property Windows

Required ❑ ❑ ❑ **cy:** If your application provides settings that control the behavior of the application, these settings are displayed in an application properties window that is accessible from an Options menu.

Recommended ❑ ❑ ❑ **cz:** If your application manages objects and allows the user to see or modify settings for these objects, these settings are displayed in an object properties window that is accessible from a Properties ... choice in the Edit, <object-type>, or Selected menus, as well as from the pop-up menu associated with the object.

Recommended ❑ ❑ ❑ **da:** If your application provides access to a Properties or Options window, this window includes the following set of buttons in the order listed, with the specified functionality, when supported by your application.

	y	n/a	n			
Required	❑	❑	❑		**OK**	Applies any changes made to components in the dialog box and dismisses it. OK may be replaced by a more appropriate label; for example, Add. The alternate label should be a verb phrase.
Optional	❑	❑	❑		**Apply**	Applies any changes made to components in the dialog box and does not dismiss it.
Required	❑	❑	❑		**Reset**	Cancels any changes that have not yet been applied by your application. The controls within the dialog box are reset to their state since the last time the dialog box action was applied. If no changes have been applied within the current invocation of the dialog box, the controls are reset to the state when the dialog box was first displayed.
Optional	❑	❑	❑		**Reset to Factory**	Cancels any changes that have not yet been applied. Components in the dialog box are reset to their default state or value as specified by the vendor that delivered the application (that is, the controls are restored to the original factory settings).
Required	❑	❑	❑		**Cancel**	Dismisses the dialog box without performing any actions not yet applied.
Required	❑	❑	❑		**Help**	Provides help for the dialog box.
Recommended	❑	❑	❑	**db:**	If your application provides a Properties window that displays settings for a selected object, the Properties window tracks the current selection and modifies the state of any controls to accurately reflect the properties of the currently selected object.	

File Selection Dialog Box

	y	n/a	n		
Optional	❑	❑	❑	**dc:**	If your application allows the user to open or save files, then it uses the standard Common Desktop Environment file selection dialog box to allow the user to select specific files and directories.

All user interactions with the file system should be facilitated by providing a point-and-click style of choosing files and directories. The user should never be forced to memorize and type in file paths. The user must be able to explore the contents and structure of the file system using scrolling lists. The expert user, however, should be able to directly enter a complete file path, as well as be able to use relative paths and environment variables such as `$HOME`.

The labels and contents of the standard file selection dialog box may be modified as appropriate to make clear the particular context in which it is being used within your application.

	y	n/a	n		
Recommended	❑	❑	❑	**dd:**	If your application allows the objects it manages to exist as separate entities within folders or toolboxes within the desktop environment, a Copy To menu option or button is provided that displays a file selection dialog box that allows the user to select the desired folder in which an icon for the object should be placed.
Recommended	❑	❑	❑	**de:**	The file selection dialog box should not display hidden (dot) directories or files, unless your users depend on using these types of files. If your application does support displaying hidden files, you should supply a check box allowing users to toggle between showing and not showing hidden files, or else allow users to toggle between showing and hiding files at a global level in your application.
Recommended	❑	❑	❑	**df:**	The file selection dialog box should not show the full path names for files and directories, but should only show the relative names, except for the directory text field

The global Common Desktop Environment setting should be:

```
XmFileSelectionBox.fullPathMode: false
```

Unless your application overrides this behavior, your file selection dialog box should not show full path names in the list boxes.

	y	n/a	n		
Required	❑	❑	❑	**dg:**	In general, the file selection dialog box should recall the directory location that was previously set by the user.

For example, if the user brings up Save As and navigates to `/users/jay/letters` to save the file, the next time the user brings up Save As, the file selection box should be in the directory

`/users/jay/letters`. This information, however, should *not* be recalled once the user has closed the primary window, but should resort to the default directory.

About Dialog Box

y n/a n

Optional ❑ ❑ ❑ **dh:** The About dialog box should contain a minimum set of information about the application that is visible in a single text pane.

That minimum set should be:

- Application name
- Version number
- Release date
- Copyright

Required ❑ ❑ ❑ **di:** The About dialog box should contain a Close button. Other buttons are optional, such as Help and More.

Other information contained in the about box might be:

Recommended ❑ ❑ ❑ **dj:** Information about the operating system or other aspects required to run the application, for example, TriTeal Enterprise Desktop 4.0.

Optional ❑ ❑ ❑ **dk:** A More Information dialog box for additional information such as development team credits, licensing, client or xhost information.

Dialog Box Layout

Optional ❑ ❑ ❑ **dl:** Controls within your dialog box are placed in a left-right, top-down layout based on the order in which the user is expected to fill out or choose options within the dialog box.

Note – This assumes that your application is being designed for a left-to-right language environment. Alternative design approaches may be necessary for other locales.

---

	y	n/a	n		
Required	❑	❑	❑	**dm:**	Push buttons that affect the dialog box as a whole, either by modifying its contents or layout, invoking the action of the dialog box, or dismissing the dialog box, are located at the bottom of the dialog box. In general, there should only be one row of buttons at the bottom of a dialog box. If your application has dialog boxes that contain several global buttons, it may be necessary to create two or more rows of buttons at the bottom of the dialog box. The last row should contain the standard dialog box buttons (OK, Reset, Cancel, and Help). If a dialog box contains buttons that are not related to the dialog box as a whole, but relate to a specific control within the dialog box, the buttons should be located with the control to which they relate.
Required	❑	❑	❑	**dn:**	If your application provides an Apply button within a dialog box, it also provides an OK button or command button that performs the dialog box action then dismisses it.
Optional	❑	❑	❑	**do:**	Your application does not use cascading buttons within dialog boxes unless there is absolutely no other design alternative that can be used without a negative impact on the layout of your dialog box. In general, cascading buttons should only be used within menus and menu bars. You should avoid their use in all other locations unless absolutely necessary.
Recommended	❑	❑	❑	**dp:**	If your application needs to use cascading buttons outside of a menu pane, you should use the `DtMenuButton` widget.

Designing Drag and Drop

Recommended	❑	❑	❑	**dq:**	You should provide a drag-and-drop (DND) method for all objects represented as icons. Provide a DND method for all elements that the user can directly manipulate.
Recommended	❑	❑	❑	**dr:**	Any basic function that your application supports through drag and drop is also supported through menus, buttons, or dialog boxes. Drag and drop is considered an accelerator to functionality that is accessible through other user interface controls supported within your application. There should be no basic operation that is supported solely through drag and drop.

	y	n/a	n		
Recommended	❑	❑	❑	**ds:**	Use an icon graphic in a dialog box or window to indicate that objects within the dialog box or window can be dragged. Use the same icon graphic used to represent the draggable object in File Manager. Place the icon adjacent to any display of the contents of the object, if such display exists. If there is no such display, place the icon in the upper right corner of the dialog box or window, unless a more suitable placement is determined. The icon should be 32x32 in size and have a label under it. The label should indicate what kind of object the icon graphic represents. The icon graphic should also be used as the source indicator in the drag icon.
Required	❑	❑	❑	**dt:**	During a drag operation, your application changes the current pointer to a drag icon. A drag icon provides visual feedback that a drag operation is in progress.
Recommended	❑	❑	❑	**du:**	During a drag operation, your application changes the current drag cursor to include a source indicator. A source indicator gives a visual representation of the elements being dragged.
Recommended	❑	❑	❑	**dv:**	During a drag operation, your application changes the current drag cursor to indicate invalid drop zones. It uses the standard Common Desktop Environment cannot pointer. The user must receive feedback as to where an object can and cannot be dropped. Minimally, feedback should be provided as to what are invalid drop zones. Preferably, feedback for valid drop zones is enhanced by use of animation, recessing of the target drop zone, and other such drag-over effects.
Recommended	❑	❑	❑	**dw:**	During a drag operation, your application changes the drop zone feedback to indicate a valid drop zone. Preferably, feedback for valid drop zones is enhanced by use of animation, recessing of the target drop zone, and other such drag-over effects.

	y	n/a	n		
Required	❑	❑	❑	**dx:**	Pressing Cancel ends a drag-and-drop operation by canceling the drag in progress. Cancel provides a consistent way for the user to cancel a drag operation.
Required	❑	❑	❑	**dy:**	Releasing BTransfer (or BSelect) when not over a drop target ends a drag-and-drop operation. Releasing BTransfer (or BSelect) offers a consistent means of ending a drag operation.
Optional	❑	❑	❑	**dz:**	Any cursor change or drag-over effect your application uses occurs within .2 seconds of the mouse pointer reaching the target area and does not interfere, in any noticeable way, with the interactive performance of the drag operation.
Recommended	❑	❑	❑	**ea:**	In a collection that supports copy, move, or link operations that can be performed by dragging, the feedback presented to the user during the drag operation indicates whether a single object or multiple objects are being manipulated. Feedback provided during the drag operation should ensure that the user feels confident that the desired set of objects is being dragged. The drag icon used for multi-object drag operations should integrate the feedback used to indicate whether the operation is a move, copy, or link.
Required	❑	❑	❑	**eb:**	After a successful transfer, the data is placed in the drop zone, and any transfer icon used by your application is removed. A transfer icon can be used to represent the type of data being transferred during a drop operation. A successful drop operation results in the transfer of data.
Required	❑	❑	❑	**ec:**	If your application removes data upon the completion of a drag and drop, it does so only if the drag-and-drop transfer has completed successfully. If a drag-and-drop operation has been canceled or failed, the data or object that was the source of the drag must not be removed.

	y	n/a	n		
Required	❑	❑	❑	**ed:**	After a failed transfer, the data remains at the drag source and is not placed in the drop zone. Any transfer icon used by your application is removed. A failed drop operation does not result in the transfer of data.
Recommended	❑	❑	❑	**ee:**	If the user drops an object at an inappropriate drop zone within your application's window, your application participates in the display of a snap back effect and also posts an error dialog box indicating the reason the drop was disallowed. The error message should state the context (for example, running action A on object B), what happened (for example, could not connect to system X), and how to correct the problem (for example, press the Help button to obtain information on diagnosing remote execution problems).
Recommended	❑	❑	❑	**ef:**	Applications that accept only single items should reject all multiple-item drops. There is no consistent method to determine which of the selected items the user really wants to drop.
Recommended	❑	❑	❑	**eg:**	If your application supports drag and drop as a means of loading a file into the application, the application responds to this operation in a manner similar to when the file is loaded through more conventional means such as choosing Open from the File menu. As an accelerator, drag-and-drop loading of files should provide the same kind of feedback and behavior as choosing Open from the File menu. For example, if changes to a currently loaded file have not yet been saved, your application should display a message dialog box asking whether the changes should first be saved before loading the new file.
Required	❑	❑	❑	**6-17:**	If your application provides any drag-and-drop help dialog boxes, they contain a Cancel button for canceling the drag-and-drop operation in progress. The Cancel button in the help dialog box provides a convenient way for the user to cancel a drag-and-drop operation.

Attachments

	y	n/a	n		
Recommended	❑	❑	❑	**eh:**	Drag and drop should not be the only method for attaching objects.
Recommended	❑	❑	❑	**ei:**	Double-clicking is a shortcut for selecting the attachment and choosing the Open menu item for attachments and should never be the only way to access attachments.
Recommended	❑	❑	❑	**ej:**	When the user attempts to drop something into the attachment list that is not attachable, then the drop fails and the item is snapped back to its source.
Recommended	❑	❑	❑	**ek:**	When the user has one or more attachments open for editing and attempts to do any operation that would result in potentially losing the user's edits, the user should be clearly warned and given the opportunity to save changes.
Recommended	❑	❑	❑	**el:**	When the user chooses something to attach from the file selection dialog box that is not an attachable item, then the user receives an error message explaining why the chosen item cannot be attached. For example: `The folder "My.Stuff" cannot be attached because it is a folder. Only documents, applications, and scripts can be attached.`

Installation

Required	❑	❑	❑	**em:**	Applications should be installed to folders in the Application Manager not directly to the Front Panel or subpanels. For consistency, only Common Desktop Environment components will install to these locations. Users may choose to rearrange their Front Panel, but applications should not do this without user consent.

Interaction

Required	❑	❑	❑	**6-18:**	A warning dialog box allows the user to cancel the destructive action about which the dialog box is providing a warning. The user needs to have a way to cancel an operation that can cause destructive results.

y n/a n

Required ❑ ❑ ❑ **en:** When your application displays a dialog box, it places the input focus at the first text field into which the user is allowed to type an entry, or at the first control within the dialog box with which the user should interact.

Input focus should always be placed at a predictable and intuitive location. The user should not be forced to set focus at the control most likely to be used when the window is displayed.

Recommended ❑ ❑ ❑ **eo:** As the user presses the Tab key within dialog boxes of your application, the input focus moves to different controls within the window in a left-right, top-down order.

Note – This assumes that your application is being designed for a left-to-right language environment. Alternative design approaches may be necessary for other locales.

Required ❑ ❑ ❑ **ep:** There is always exactly one control within any window of your application that has the input focus if the window in which it resides has the input focus.

If any window within your application has focus, some control within that window must have focus. The user should not have to explicitly set focus to a control within the window.

Optional ❑ ❑ ❑ **eq:** When a text field within your application does not have the input focus, the text cursor is not displayed within that field.

Although use of inactive text cursors is allowed within the Motif style, it is better to hide the text cursor on focus out rather than display the inactive text cursor. This makes it easier for the user to quickly scan the screen or a window and determine which text field currently has focus.

Optional ❑ ❑ ❑ **er:** Your application provides keyboard mnemonics for all buttons, menus, and menu items displayed within the application.

Once the user becomes adept at using your application, keyboard mnemonics provide the user a quick way to access functionality. Mnemonics also facilitate access to functionality from within keyboard-centric applications or windows. The user need not frequently switch between use of the mouse or use of the keyboard. Mnemonics should be provided pervasively throughout the user interface.

	y	n/a	n		
Optional	❑	❑	❑	**es:**	Your application provides keyboard accelerators for those functions that are expected to be used frequently by the user. Keyboard accelerators provide the user who has become expert at using your application a quick way to access application functionality without having to go through menus and dialog boxes.
Required	❑	❑	❑	**et:**	Dialog boxes displayed by your application never block input to other applications within the desktop (that is, they are not system modal) unless it is absolutely essential that the user perform no other action in the desktop until the user responds to the dialog box. Applications must allow the user the freedom to access information and tools within the user's desktop environment. Only in the most dire circumstances should an application ever block access to other applications and services within the environment.
Required	❑	❑	❑	**eu:**	Dialog boxes displayed by your application never block access to other functionality within the application (application modal) unless it is essential that the state of the application remains unchanged until the user responds to the dialog box.
Required	❑	❑	❑	**ev:**	If your application does not use the values of global environment settings, such as multiclick timeout intervals, drag thresholds, window color settings, mouse left- or right-handedness, and so on, but instead uses its own values for these settings, then your application provides one or more Options dialog boxes that allow the user to change the values for these settings. In general, you should not override the value of settings treated as global environment settings. These settings are controlled by the user through the Common Desktop Environment Style Manager. If you choose to ignore these settings and specify your own settings, then your application will behave inconsistently with other applications in the Common Desktop Environment. If you nevertheless choose to provide your own values, then you must provide the user a way to make your settings consistent with the rest of the desktop.

Visuals

	y	n/a	n		
Recommended	❏	❏	❏	**ew:**	Any icons or graphics displayed by your application are designed to be distinguishable on low- (640x480), medium- (800x600), and high- (mega-pixel) resolution displays. Alternatively, your application provides different sized visuals for low-, medium-, and high-resolution displays. Desktop system configurations are including more high-resolution monitors. The user must be able to discern any visuals used by your application on these type of monitors. The embedded base, however, still contains many standard VGA monitors. Your application's visuals must display well on these systems and should not appear overly large.
Recommended	❏	❏	❏	**ex:**	Any icons or graphics displayed by your application are designed to display well on black-and-white and gray-scale monitors. These visuals also display well on low-color (16) systems.
Recommended	❏	❏	❏	**ey:**	Icons should be used to represent only objects and applications. Icons provide a visual representation for objects and facilitate direct manipulation. If icons are used for other purposes (for example, as illustrations) where the user can't drag them, select them, and so on, it creates a confusing inconsistency.
Recommended	❏	❏	❏	**ez:**	Icons should use only the palette of 22 colors. The Common Desktop Environment icon palette was chosen to maximize attractiveness and readability without using an unnecessary number of colors. Use of additional colors may cause undesirable color shifting on the display.
Recommended	❏	❏	❏	**fa:**	Icons should be designed for international use. Don't use text, symbols, humor, animals, and other items that may be interpreted differently in other cultures.
Recommended	❏	❏	❏	**fb:**	16x16 and 32x32 icons are left-aligned; any empty bits are on the right side of the bounding box.
Recommended	❏	❏	❏	**fc:**	48x48 icons are centered in the bounding box.

Toolbars

	y	n/a	n		
Required	❑	❑	❑	**fd:**	If you use a tool bar, it should be used only in windows with a menu bar.
Required	❑	❑	❑	**fe:**	Tool bars should contain only operations that are already available to the user in your application menus. All items in a tool bar should be redundant.
Required	❑	❑	❑	**ff:**	When an action represented by a tool bar icon is unavailable to the user, that icon should be made insensitive, with the associated stippled appearance. Whenever a menu item is made insensitive, the corresponding tool bar item must be made insensitive as well.
Recommended	❑	❑	❑	**fg:**	Give users the option to hide the tool bar.
Required	❑	❑	❑	**fh:**	The tool bar container is placed directly under the menu bar and should be the same width as the window, as well as similar height to the menu bar.
Recommended	❑	❑	❑	**fi:**	If you use a tool bar in your application, then you should provide a status line in the same primary window as the tool bar. This status line should provide immediate feedback to the user as to the purpose of the button that the mouse is currently over or that has the keyboard focus. When the arrow is over a tool bar icon, the status line should display a brief definition of what the icon represents or what will happen when the user clicks the icon.
Recommended	❑	❑	❑	**fj:**	You may provide labels under tool bar icons. These labels should serve to explain the purpose of the icon.
Recommended	❑	❑	❑	**fk:**	Drawn buttons in the tool bar should be the same width and height. Similar or related items should be grouped, and groups should be evenly spaced across the tool bar.
Recommended	❑	❑	❑	**fl:**	All pixmaps in the tool bar should be the same size. This ensures that all the tool bar buttons are the same size.
Recommended	❑	❑	❑	**fm:**	The recommended size of the pixmap is 24x24. The default for the drawn button is to resize itself according to the size of its label type, which, in this case, would be a pixmap.

Expandable Windows

	y	n/a	n		
Recommended	❑	❑	❑	**fn:**	The primary pane of the dialog box or window should contain all of the controls needed to complete the task. This should include all critical and frequently used functionality.
Recommended	❑	❑	❑	**fo:**	It is assumed that infrequently used features are placed in the secondary pane. The core functionality of the application should not depend on any controls placed in secondary panes.
Required	❑	❑	❑	**fp:**	Command buttons are aligned along the bottom of the dialog box. When the window is expanded to show a secondary pane, then buttons are moved to the bottom of the secondary pane. See Chapter , “Application Design Principles” for information about layout of action buttons in dialog boxes.
Recommended	❑	❑	❑	**fq:**	If important controls *must* be placed in the secondary pane, the application can specify that the window in question should be displayed in its expanded state by default. Users should still be able to shrink the window by pressing the Contract button.
Recommended	❑	❑	❑	**fr:**	The secondary pane should expand in the direction most consistent with users’ expectations, the reading pattern of the language in which it will be displayed, and the content of the information displayed.
Recommended	❑	❑	❑	**fs:**	If possible, the panes should have the same default width.
Required	❑	❑	❑	**ft:**	A separator should be used to separate the primary pane from the secondary pane. The user needs to have clear visual feedback as to which elements are in the primary and which in the secondary panes of the expandable window.
Required	❑	❑	❑	**fu:**	If a window is resizable, any sizing changes should be allocated to the pane containing scrolling lists or text fields whose displayed length is less than their stored length. If both panes contain scrollable controls, size changes should be distributed evenly between the two panes. If neither pane contains scrollable controls, the window should not be resizable.

	y	n/a	n		
Required	❑	❑	❑	**fv:**	The expandable window should have one button that changes its label based on the state of the window.
Required	❑	❑	❑	**fw:**	The expand button should have two labels that reflect the two states of the expandable window accurately. The current label should indicate to the user what will happen if the user clicks the button. Examples of possible labels are Basic and Options, Expand and Contract, and More and Less.
Optional	❑	❑	❑	**fx:**	The expand button may contain a graphic in addition to the label. This graphic should indicate the direction in which the window will expand or contract.
Recommended	❑	❑	❑	**fy:**	The button should appear in the lower left-hand corner of the window or dialog box for expansion in the vertical direction and in the lower-right hand corner for expansion in the horizontal direction.
Required	❑	❑	❑	**fz:**	If the window or dialog box contains a scrolling list positioned to the far right side of the pane, do not align the drawn button with the scroll bar. For example, the button should be aligned with the list, not the scroll bar.
Required	❑	❑	❑	**ga:**	Applications must remember the state of each window or dialog box (expanded or not expanded) independently (not collectively). The state should be changed only by the user and should always be preserved until explicitly altered by the user.
Recommended	❑	❑	❑	**gb:**	Applications should remember the state of each expandable window or dialog box across sessions, so that users don't have to manually configure the expandable windows each time the application is run. If appropriate, applications can provide a mechanism, as an option, to allow users to set the state of an expandable window globally for the application. This would be part of the application's Options.

Messages

	y	n/a	n		
Recommended	❑	❑	❑	**gc:**	Messages displayed by your application do not assume that the user has any expert knowledge about computer systems in general, or the UNIX system in particular.

It is appropriate to assume that the user has knowledge about basic terms used within the desktop, such as files or programs. Such knowledge can be assumed to have been learned by the user through Tutorials, online help, and user documentation. However, terminology that is typically understood only by an expert or frequent computer user should be avoided unless the application is specifically targeted at computer professionals. Likewise, messages returned to your application by the underlying operating system should not be passed through to the user, but instead, should be "translated" into language that can be understood by the novice user.

	y	n/a	n		
Recommended	❏	❏	❏	**gd:**	Error messages displayed by your application indicate the possible cause of the error and indicate the possible actions the user can take in response.
Optional	❏	❏	❏	**ge:**	Your application uses audio feedback, in addition to any messages displayed, to signal error conditions and events.
Optional	❏	❏	❏	**gf:**	Don't rely on error messages from the kernel and library routines. Error messages from kernel and library routines are normally not seen by the user, and even when the user does see them, they are usually too low-level and cryptic to be understood by nonprogrammers. Applications should check for error conditions and use an error dialog box to present an appropriate error message in terms of the user's actions and intentions.
Recommended	❏	❏	❏	**gg:**	Your application displays a confirmation or warning message dialog box to the user when an action instigated by the user will be irreversible and potentially destructive with respect to the information stored within the system or the operation of the system or desktop environment.
Optional	❏	❏	❏	**gh:**	Urgent conditions that require immediate attention by the user, no matter which application or desktop service the user is currently accessing, are brought to the user's attention using audiovisual notification. The alarm is signaled in the current workspace regardless of the workspace in which the application resides.

Some applications, such as network monitors or stock watch programs, may need to grab the user's immediate attention to some event. Both visual and audio alarms should be used to signal the user. The user should be able to acknowledge the alarm and cause it to cease.

y n/a n

Recommended ❑ ❑ ❑ **gi:** Your application uses footer messages only to communicate status, progress, or information (help) messages. It does not use the footer to present error messages.

The footer is a good location for prompt messages that help the user to determine how to choose options within a window or fill out a particular field. It should not be used to present error messages to the user or informational messages that are important for the user to notice. These should be presented in the appropriate style message dialog box.

Recommended ❑ ❑ ❑ **gj:** Your application provides a Help button in all message dialog boxes, except those that contain self-explanatory messages.

Applications should be designed with both the expert and novice user in mind. The novice user must be able to access additional information clarifying the message, the circumstances under which it might have been displayed, and what the user should do in response to the message.

Recommended ❑ ❑ ❑ **gk:** Your application uses the appropriate style dialog box for the display of messages to the user.

Optional ❑ ❑ ❑ **gl:** An information dialog box is used to display status, completion of activity, or other informative types of messages to which the user need not necessarily respond other than to acknowledge having read the message.

Minimally, information dialog boxes should have an OK button so that the user can dismiss the dialog box. If there is additional information available about the situations under which the message is displayed or other references for the topic to which the message relates, then a Help button should be included.

Optional ❑ ❑ ❑ **gm:** An error dialog box is used to display error messages to the user. The error dialog box displayed states what the error is and specifies why it occurred. The error dialog box contains a Help button so that the user may get additional information, unless the message is self-explanatory. The error dialog box contains an OK button that dismisses the dialog box.

A Cancel button is not required for error dialog boxes unless the error resulted in the suspension of an activity that was in progress. In this case, the message should indicate whether the user has the option to continue the activity or stop it, and the buttons for the dialog box should be Continue, Cancel, and Help. In general, error dialog boxes should not be modal unless it is critical that the user not continue interacting with the application until the user has acknowledged having read the error message.

	y	n/a	n		
Optional	❑	❑	❑	**gn:**	A question dialog box is used to ask questions of the user. The question is clearly worded to indicate what a Yes response or a No response means. The buttons displayed are Yes, No, and Help. Help provides additional information as to what the application will do in response to a Yes or No choice.

Where possible, you should extend the label for the Yes and No buttons to make it clear what action will be performed as a result of choosing either option. For example, if the user has made changes to a document and has not saved these but has chosen the application's Exit option, you might display a question dialog box that asks “Changes have not been saved. Do you want to save these before exiting?” The buttons should be Save, Discard, Cancel, and Help. These labels allow the more experienced user to click the correct button without having to carefully read the question and relate it to the button labels.

Optional ❑ ❑ ❑ **go:** A warning dialog box is used to communicate the consequences of an action requested by the user that may result in the loss of data, system or application accessibility, or some other undesirable event. The dialog box is presented before the action is performed and offers the user the opportunity to cancel the requested operation. The buttons displayed are Yes, No, and Help, or Continue, Cancel, and Help. Help provides additional information on the consequences of performing the action requested.

The use of Yes and No, or Continue and Cancel, depends on the wording of your message. The labels for Yes and No should be extended as suggested previously. Continue may be replaced with a label more specific to the action that will be performed.

Optional ❑ ❑ ❑ **gp:** A working dialog box is used to display in-progress information to the user when this information is not displayed in the footer of your application's window. The dialog box contains a Stop button that allows the user to terminate the activity. The operation is terminated at the next appropriate breakpoint, and a confirmation might be displayed asking whether the user really wants to stop the activity. The confirmation message might state the consequences of stopping the action.

	y	n/a	n		
Optional	❑	❑	❑	**gq:**	Your application writes error messages to the Common Desktop Environment error log when it is not appropriate to display these to the user in a message dialog box, but when the message may nevertheless be useful in diagnosing problems.

You might also write error messages that are displayed to the user in the error log if it would be valuable to the user or an administrator to refer to these messages at some later time. Messages written to the error log should provide additional information about the error and should state the context in which the error occurred.

Optional ❑ ❑ ❑ **gr:** Informational messages should be left aligned and displayed in a light font in keeping with their unobtrusive nature. Note that the margin where informational messages are displayed should *not* accept mouse focus.

Optional ❑ ❑ ❑ **gs:** Progress messages should normally be displayed only while the operation is in progress. Notices and other information that is no longer valid should be removed within a few seconds to avoid confusion about whether or not the information is current.

Work-in-Progress Feedback

Recommended ❑ ❑ ❑ **gt:** If any command chosen by the user is expected to take longer than 2 seconds to complete, but less than 10 seconds, your application displays the standard busy pointer as feedback that the command is executing.

The user must receive assurance that your application has “heard” the request and is working on it. If the results of the request cannot be displayed immediately, some feedback must be provided. The busy pointer should be displayed within 0.5 seconds of execution of the command.

y n/a n

Recommended ❑ ❑ ❑ **gu:** If any command chosen by the user is expected to take longer than 10 seconds to complete, your application displays a working dialog box or other feedback of similar character that indicates that the application is working on the request. The feedback should reveal progress toward completion of the activity.

If an activity is expected to take a significant amount of time (10 seconds or more), your application should display feedback stronger than the busy pointer. Displaying the busy pointer for long amounts of time may lead the user to conclude that the application has become "hung." A progress indicator should be displayed in these scenarios that indicates that the application is still functioning and is working on the user's request. The progress indicator should show how much of the activity has been completed and what amount remains.

Recommended ❑ ❑ ❑ **gv:** When your application displays work-in-progress feedback to the user, it does not block access to other applications and services within the desktop environment.

Multitasking should always be supported and, as such, your application should allow the user to access other services while it is busy performing some activity. Preferably, the user is also able to access other features within your application even though it is currently working on another request. When this is supported, your application should display an enhanced busy pointer that indicates that the application is busy but still willing to accept input.

Controls, Groups, and Models

CheckButton

Required ❑ ❑ ❑ **7-1:** The application uses check buttons to select settings that are not mutually exclusive. A check button graphically indicates its state with the presence or absence of a check mark.

A check button is used to select settings that are not mutually exclusive. The user needs to know whether the button is set or not.

y n/a n

Required ❑ ❑ ❑ **7-2:** When the user presses BSelect in a check button, the check button is armed. If the check button was previously unset, it is shown in the set state. If the check button was previously set, it is shown in the unset state.

BSelect Press arms a check button and shows the result of activating it by releasing BSelect.

Required ❑ ❑ ❑ **7-3:** When the user releases BSelect in the same check button in which the press occurred:

- If the check button was previously unset, it is set.
- If the check button was previously set, it is unset.

In all cases the check button is disarmed, and, if the check button is in a menu, the menu is unposted.

BSelect Release activates a check button.

Required ❑ ❑ ❑ **7-4:** When the user presses the Enter or Return key in a check button, if the check button is in a window with a default action, the default action is activated. If the check button is in a menu:

- If the check button was previously unset, it is set.
- If the check button was previously set, it is unset.
- In both cases, the check button is disarmed, and the menu is unposted.

The Enter and Return keys perform the default action of a window or activate a check button in a menu.

Required ❑ ❑ ❑ **7-5:** When the user presses the Select key or Spacebar in a check button, if the check button was previously unset, it is set. If the check button was previously set, it is unset. In both cases, the check button is disarmed, and, if the check button is in a menu, the menu is unposted.

The Select key and Spacebar activate a check button.

ComboBox

	y	n/a	n		
Required	❑	❑	❑	**gw:**	In a list that can be scrolled, such as a scrollable list box, do not allow the cursor to wrap.
Required	❑	❑	❑	**gx:**	Provide vertical scroll bars when some of the data is not visible in the combo box.
Recommended	❑	❑	❑	**gy:**	Provide horizontal scroll bars when elements are wider than the list box.
Recommended	❑	❑	❑	**gz:**	Display the elements in an order that is meaningful to the user.
Recommended	❑	❑	❑	**ha:**	Display an initial value from the list in the text-entry field. Display selected emphasis on the initial value so that typed text will replace the value.
Recommended	❑	❑	❑	**hb:**	Make the combination box large enough to display a minimum of six list items at a time.
Recommended	❑	❑	❑	**hc:**	When a user increases the size of the window in which the combo box is displayed, increase the number of items displayed in the combo box.
Recommended	❑	❑	❑	**hd:**	When a user decreases the size of the window in which the combo box is displayed, decrease the number of items displayed in the combo box. As a minimum, reduce the combo box to the text-entry field and a list box with one entry displayed. If the window is sized so that two list items cannot be displayed, clip the combo box.

CommandBox

Required	❑	❑	❑	**7-6:**	If your application uses a command box, it is composed of a text component with a command-line prompt for text input and a list component for a command history area. The list uses either the single selection or browse selection model. This specification ensures the consistent appearance and operation of a command box across applications.
Required	❑	❑	❑	**7-7:**	When an element of a command box list is selected, its contents are placed in the text area. This specification provides a convenient way of selecting a previously entered command.

y n/a n

Required ❑ ❑ ❑ **7-8:** The list navigation actions Up Arrow, Down Arrow, Control+Begin, and Control+End are available from the text component for moving the cursored element within the list and thus changing the contents of the text.

These actions provide a convenient way to choose a command from the list while focus remains in the text component.

Required ❑ ❑ ❑ **7-9:** The default action of the command box passes the command in the text area to the application for execution and adds the command to the end of the list.

Maintaining a history of commands provides a convenient means of entering often-used commands.

FileSelectionBox

Required ❑ ❑ ❑ **7-10:** If your application uses a file selection dialog box, it contains the following components:

- A directory text component showing the current directory path. The user can edit the directory text component and press Return or Enter to change the current directory.
- For applications that allow saving to different formats, an option button allowing users to specify the format when saving a file.
- A file name text component for displaying and editing a file name. This component is optional when the file selection box is used to choose an existing file or directory.
- A group of push buttons, including a command button, and Update, Cancel, and Help buttons. The command button is typically labeled Open or Save, but if there is another label that better describes the resulting action (such as Include), that label should be used. Activating the command button carries out the corresponding action and dismisses the file selection box.

Recommended ❑ ❑ ❑ **he:** When the file selection box is used to specify an existing file (for example, to open a document), the command button is normally labeled Open and it should be the default action.

	y	n/a	n		
Recommended	❑	❑	❑	**hf:**	If the Update button is activated while a directory is selected in the contents list, the directory is opened, its contents are displayed in the contents list, and the directory text is updated.
Required	❑	❑	❑	**hg:**	If the Open button is activated while the appropriate file is selected in the contents list, the file is utilized by the application and the file selection box is dismissed.
Recommended	❑	❑	❑	**hh:**	When the file selection dialog box is used to choose an existing directory (for example, to install a set of files into the chosen directory) or to specify a new directory, the command button should be given an appropriate label, such as Install, Choose, Create, or OK. If this button is activated while the appropriate directory is selected in the contents list, the directory is utilized by the application and the file selection box is dismissed.
Required	❑	❑	❑	**hi:**	When the file selection dialog box is used to choose an existing directory, there must also be an additional button, labeled Update, that is enabled whenever a directory is selected in the contents list, and opens the directory. This Update button is the default action.
Required	❑	❑	❑	**hj:**	When the file selection dialog box is used to specify a new file name (for example, a Save As dialog box), the command button is normally labeled Save and is the default action. This specification ensures the uniform appearance of a file selection box across applications.
Optional	❑	❑	❑	**hk:**	When the file selection dialog box is used to choose an existing file, files are shown in the contents list but they are all disabled. Double-clicking BSelect on a disabled file name has no effect.
Required	❑	❑	❑	**hl:**	The normal text navigation and editing functions are available in the text components for moving the cursor within each text component and changing the contents of the text. These actions provide a convenient way to choose a directory or file name from the corresponding list while focus remains in the text component.
				7-11:	This item has been deleted.

	y	n/a	n	
Required	❑	❑	❑	**7-12:** Double-clicking BSelect on an item in the contents list selects that item and activates the default action. In all cases, double-clicking BSelect on a directory in the contents list opens that directory and displays its contents in the contents list (the default action is Open).

- When the file selection box is used to choose an existing file, double-clicking BSelect on an appropriate file in the contents list chooses that file and dismisses the file selection box (the default action is Open).
- When the file selection box is used to choose an existing directory or to specify a new directory or file, the files list should not appear.

Required ❑ ❑ ❑ **7-13:** The normal text navigation and editing functions are available in the text components for moving the cursor within each text component and changing the contents of the text.

7-14: This item has been deleted.

Optional ❑ ❑ ❑ **7-15:** Your application allows the user to select a file by scrolling through the list of file names and selecting the desired file or by entering the file name directly into the file selection text component. Selecting a file from the list causes that file name to appear in the file selection text area.

This method of selecting a file needs to be consistent across applications.

Required ❑ ❑ ❑ **7-16:** Your application makes use of the selection when one of the following occurs:

- The user activates the command push button while an appropriate item is selected in the contents list.
- The user double-clicks BSelect on an appropriate file in the contents list.
- The user presses Return or Enter while the file name text component has the keyboard focus and contains an appropriate item.

	y	n/a	n		
Required	❑	❑	❑	**7-17:**	The file selection box displays the contents of a directory in the contents list when the file selection box is initialized, when the user presses Enter or Return in the directory text component, and when the user opens a directory in the contents list. The contents list is updated each time the contents of the directory changes. This specification ensures the consistent operation of a directory and file search in a file selection box.
Recommended	❑	❑	❑	**hm:**	If the user has opened the application without supplying a file name argument, the Open dialog box should use the user's home directory as the default directory. An exception to this rule might be made if a clearly more useful directory can be identified; for example, the icon editor might default to `HomeDirectory/.dt/icons`. For Applications that allow editing, never default to a directory in which the user does not have read and write permission, such as `/usr/dt/bin`.
Required	❑	❑	❑	**hn:**	If the user has opened the application with a file name argument, the Open dialog box should default to the directory in which that file resides.
Optional	❑	❑	❑	**ho:**	When using the file selection dialog box in Save As capacity, provide a default name of Untitled, place the location cursor in the file name field and highlight the file name text to create a "delete pending type-in" mode. If the current directory already has a file of that name, create a name Untitled2, and so forth.
Optional	❑	❑	❑	**hp:**	When using the file selection dialog box in a Save As capacity, add a file name extension if the application supports file typing by extension, and make this extension visible in the file name field. Do not highlight the extension to create a "delete pending type-in" mode, but allow users to modify the extension or delete it explicitly.
Optional	❑	❑	❑	**hq:**	The file selection dialog box should come up in a directory that makes sense for the task. For example, when saving a new file from an editor, the file selection box should come up in the user's home directory. If the user navigates to some other directory within the file selection box, the application should remember that directory the next time it is brought up.

y n/a n

Optional ❑ ❑ ❑ **hr:** Users should never be allowed to overwrite an existing file through the file selection box without a warning dialog box prompt.

Optional ❑ ❑ ❑ **hs:** Keyboard focus should be placed in the file name field each time users bring up a file selection dialog box.

Optional ❑ ❑ ❑ **ht:** Directory and file name lists should be presented alphabetically, case insensitive. The first item on the directory list should be the parent directory and it should be labeled "..".

Optional ❑ ❑ ❑ **hu:** Labels should be clear. In the English language, use the following labels for the file selection dialog box fields and lists:

Table 10-4 File Selection Dialog Box Labels

Component	Label
Directory text field	Enter Path or Folder Name:
Filter text Field	Filter:
Directory list	Folders:
Contents list	Files:
File text field	Enter File Name:*

Optional ❑ ❑ ❑ **hv:** Optionally, application developers can make this label more instructive and specific, such as Enter File to Open for Open dialog boxes.

These labels should be the default labels. If they are not set by default, you need to set them through resources in your application's app-defaults file.

List

	y	n/a	n		
Required	❑	❑	❑	**7-18:**	Within a list component, your application uses the Up Arrow key to move the location cursor to the previous item in the list and the Down Arrow key to move the location cursor to the next item in the list. In a scrollable list, the Left Arrow key scrolls the list one character to the left, and the Right Arrow key scrolls the list one character to the right. The arrow keys provide a consistent means of moving the location cursor within a list component.
Required	❑	❑	❑	**7-19:**	Within a list component, your application uses Control+Begin to move the location cursor to the first item in the list and Control+End to move the location cursor to the last item in the list. In a scrollable list, the Begin key moves the horizontal scroll region so that the leftmost edge of the list is visible, and the End key moves the horizontal scroll region so that the rightmost edge of the list is visible. These keys offer a convenient mechanism for moving the location cursor quickly through a list.
Required	❑	❑	❑	**7-20:**	Within a scrollable list, the Page Up key moves the location cursor to the item one page up in the list, and the Page Down key moves the location cursor to the item one page down in the list. In a scrollable list, the Page Left key (or Control+Page Up) scrolls the list one page to the left, and the Page Right key (or Control+Page Down) scrolls the list one page to the right. These keys offer a convenient mechanism for paging through a list.
Required	❑	❑	❑	**7-21:**	Within a list component, your application uses BSelect Click 2 to select the item that was double-clicked and then initiate any default action for the window. Double-clicking using BSelect provides a consistent way of activating the default action for a list.

Option Button

	y	n/a	n		
Required	❑	❑	❑	**7-22:**	If your application uses option buttons, the label for the button is the last selection made from the option button. An option button is used to post an option menu which allows the user to select from a number of choices. The label of an option button needs to display the most recent selection from the associated option menu.
Required	❑	❑	❑	**7-23:**	When the user presses BSelect or BMenu in an option button, the associated option menu is posted. BSelect Press is a consistent way of activating an option button.
Required	❑	❑	❑	**7-24:**	When the user releases BSelect or BMenu within the same option button that the press occurred in, the associated option menu is posted if it was not posted at the time of the press. When the user releases BSelect or BMenu outside of the option button, the associated option menu is unposted. BSelect Release or BMenu Release posts or unposts an option menu, depending on whether the release occurs inside the option button and whether the option menu was posted at the time of the press.
Required	❑	❑	❑	**7-25:**	When the user presses the Select key or Spacebar in an option button, the associated option menu is posted. The Select key or Spacebar posts an option menu from the keyboard.

Paned Window

Required	❑	❑	❑	**7-26:**	If your application uses paned windows, they are composed of any number of groups of components, called panes, each separated by a sash and a separator. The panes, sashes, and separators are grouped linearly, either horizontally or vertically. A sash is the handle on a separator between two panes that is used to adjust the position of the separator. This specification ensures the consistent appearance of a paned window across applications.

Panel

	y	n/a	n		
Required	❑	❑	❑	**7-27:**	The Down Arrow, Left Arrow, Right Arrow, and Up Arrow directional keys navigate among components in a panel.

A panel group organizes a collection of basic controls in a horizontal, vertical, or two-dimensional layout. The directional keys are used to navigate among the controls.

Push Button

Required ❑ ❑ ❑ **7-28:** When the user presses BSelect in a push button, the push button is armed. When the user releases BSelect in the same push button that the press occurred in, the push button is disarmed and activated. When the user releases BSelect outside the push button, the push button is disarmed but not activated.

BSelect provides a consistent means of activating a push button.

Required ❑ ❑ ❑ **7-29:** When the user presses the Enter or Return key in a push button that is in a window with a default action, the push button is activated. When the user presses the Enter or Return key in a push button in a menu, the push button is activated and the menu is unposted.

The Enter and Return keys activate a dialog box or a push button in a menu.

Required ❑ ❑ ❑ **7-30:** When the user presses the Select key or Spacebar in a push button, the push button is activated. If the push button is in a menu, the menu is unposted.

The Select key and Spacebar activate a push button.

Radio Button

Required ❑ ❑ ❑ **7-31:** If your application uses radio buttons, each button graphically indicates its state.

Radio buttons are used to represent a panel of mutually exclusive selections. The user needs to know which button in the panel is set.

	y	n/a	n		
Required	❑	❑	❑	**7-32:**	When the user presses BSelect in a radio button, the radio button is armed. If the radio button was previously unset, it is shown in the set state. BSelect Press arms a radio button and shows the result of activating it by releasing BSelect.
Required	❑	❑	❑	**7-33:**	When the user releases BSelect in the same radio button that the press occurred in and the radio button was previously unset, it is set, and any other radio button in the same panel that was previously set is unset. The radio button is disarmed, and, if the radio button is in a menu, the menu is unposted. BSelect Release activates a radio button.
Required	❑	❑	❑	**7-34:**	When the user presses the Enter or Return key in a radio button, if the radio button is in a window with a default action, the default action is activated. If the radio button is in a menu: • If the radio button was previously unset, it is set, and any other radio button in the same panel that was previously set is unset. • The radio button is disarmed, and the menu is unposted. The Enter and Return keys perform the default action of a window or activate a radio button in a menu.
Required	❑	❑	❑	**7-35:**	When the user presses the Select key or Spacebar in a radio button, if the radio button was previously unset, it is set, and any other radio button in the same panel that was previously set is unset. The radio button is disarmed, and, if the radio button is in a menu, the menu is unposted. The Select key and Spacebar activate a radio button.

Sash

Required	❑	❑	❑	**7-36:**	Within a paned window, your application uses a sash to adjust the position of a separator, which adjusts the sizes of the panes next to it. As a sash is moved, the pane in the direction of the sash movement gets smaller and the opposite pane gets larger by an equal amount.

This specification results in the uniform operation of a paned window across applications.

	y	n/a	n		
Required	❑	❑	❑	**7-37:**	Within a sash, BSelect Motion or BTransfer Motion causes the sash to track the movement of the pointer. In a vertically oriented paned window, the sash tracks the vertical position of the pointer. In a horizontally oriented paned window, the pane tracks the horizontal position of the pointer.

BSelect, mouse button 1, and BTransfer, mouse button 2, provide a consistent means of moving a sash in a paned window using the mouse.

Required ❑ ❑ ❑ **7-38:** The Up Arrow and Down Arrow keys (for a sash that can move vertically) and the Left Arrow and Right Arrow keys (for a sash that can move horizontally) move the sash one increment in the specified direction.

The arrow keys offer a uniform means of moving a sash in a paned window.

Required ❑ ❑ ❑ **7-39:** Control+Up Arrow and Control+Down Arrow (for a sash that can move vertically) and Control+Left Arrow and Control+Right Arrow (for a sash that can move horizontally) move the sash one large increment in the specified direction.

These keys provide a convenient way of moving a sash quickly in a paned window.

Scale

Required ❑ ❑ ❑ **7-40:** If a scale has arrow buttons, your application uses BSelect Press in an arrow button to move the slider one increment in the direction of the side of the slider on which the button was pressed and autorepeats until the button is released.

BSelect Press provides a consistent means of adjusting a scale component using the mouse.

Required ❑ ❑ ❑ **7-41:** In a scale trough, if the scale has tick marks, BSelect Press moves the slider one major tick mark in the direction of the side of the slider on which the trough was pressed and autorepeats until the button is released. If the scale does not have tick marks, BSelect Press in the trough moves the slider one large increment in the direction of the side of the slider on which the trough was pressed and autorepeats until the button is released.

BSelect Press provides a consistent means of adjusting a scale component using the mouse.

	y	n/a	n		
Required	❑	❑	❑	**7-42:**	Within a scale slider, BSelect Motion causes the slider to track the position of the pointer. In a vertical scale, the slider tracks the vertical position of the pointer. In a horizontal scale, the slider tracks the horizontal position of the pointer.

BSelect Motion offers a convenient way to adjust a scale component precisely using the mouse.

Required	❑	❑	❑	**7-43:**	Within a scale slider or trough, BTransfer Motion positions the slider to the point of the button press and then causes the slider to track the position of the pointer. In a vertical scale, the slider tracks the vertical position of the pointer. In a horizontal scale, the slider tracks the horizontal position of the pointer.

BTransfer Motion provides another convenient way to adjust a scale component precisely using the mouse.

Required	❑	❑	❑	**7-44:**	If a mouse-based sliding action is in progress, the Cancel key cancels the sliding action and returns the slider to its position prior to the start of the sliding operation.

The Cancel key provides a consistent way for the user to cancel a mouse-based sliding action.

Required	❑	❑	❑	**7-45:**	In a vertical scale, the Up Arrow and Down Arrow keys move the slider one increment in the specified direction. In a horizontal scale, the Left Arrow and Right Arrow keys move the slider one increment in the specified direction.

The arrow keys provide a uniform way of adjusting the slider in a scale component using the keyboard.

Required	❑	❑	❑	**7-46:**	In a vertical scale, Control+Up Arrow and Control+Down Arrow move the slider one large increment in the specified direction. In a horizontal scale, Control+Left Arrow and Control+Right Arrow move the slider one large increment in the specified direction.

These keys provide a convenient way of adjusting the slider in a scale component quickly using the keyboard.

	y	n/a	n		
Required	❑	❑	❑	**7-47:**	Your application uses the Begin key or Control+Begin to move the slider to its minimum value. The End key or Control+End moves the slider to its maximum value.

These keys provide a convenient mechanism for setting a scale to its minimum or maximum value using the keyboard.

ScrollBar

Required ❑ ❑ ❑ **7-48:** Within a scroll bar, your application uses BSelect Press in an arrow button to move the slider one increment in the direction of the side of the slider on which the button was pressed and autorepeats until the button is released.

BSelect Press provides a consistent means of adjusting a scroll bar using the mouse.

Required ❑ ❑ ❑ **7-49:** In the trough of a scroll bar, BSelect Press moves the slider one page in the direction of the side of the slider on which the trough was pressed and autorepeats until the button is released.

BSelect Press provides a consistent means of adjusting a scroll bar using the mouse.

Required ❑ ❑ ❑ **7-50:** Within a scrollbar slider, BSelect Motion causes the slider to track the position of the pointer. In a vertical scroll bar, the slider tracks the vertical position of the pointer. In a horizontal scroll bar, the slider tracks the horizontal position of the pointer.

BSelect Motion offers a convenient way to adjust a scroll bar precisely using the mouse.

Required ❑ ❑ ❑ **7-51:** Within a scrollbar slider or trough, BTransfer Motion positions the slider to the point of the button press and then causes the slider to track the position of the pointer. In a vertical scroll bar, the slider tracks the vertical position of the pointer. In a horizontal scroll bar, the slider tracks the horizontal position of the pointer.

BTransfer Motion offers another convenient way to adjust a scroll bar precisely using the mouse.

	y	n/a	n		
Required	❑	❑	❑	**7-52:**	If a mouse-based scrolling action is in progress, pressing the Cancel key cancels the scrolling action and returns the slider to its position prior to the start of the scrolling operation. The Cancel key provides a consistent way for the user to cancel a mouse-based scrolling action.
Required	❑	❑	❑	**7-53:**	In a vertical scroll bar, the Up Arrow and Down Arrow keys move the slider one increment in the specified direction. In a horizontal scroll bar, the Left Arrow and Right Arrow keys move the slider one increment in the specified direction. The arrow keys provide a uniform means of adjusting a scroll bar using the keyboard.
Required	❑	❑	❑	**7-54:**	In a vertical scroll bar, Control+Up Arrow and Control+Down Arrow move the slider one large increment in the specified direction. Control+Left Arrow and Control+Right Arrow move the slider one large increment in the specified direction. These keys provide a convenient way of adjusting a scroll bar quickly using the keyboard.
Required	❑	❑	❑	**7-55:**	Your application uses the Page Up and Page Down keys to move the slider in a vertical scroll bar one page in the specified direction. The Page Left key (or Control+Page Up) and the Page Right key (or Control+Page Down) move the slider in a horizontal scroll bar one page in the specified direction. These keys allow for the convenient movement of the slider in a scroll bar using the keyboard.
Required	❑	❑	❑	**7-56:**	Your application uses the Begin key or Control+Begin to move the slider to the minimum value. The End key or Control+End moves the slider to the maximum value. These keys offer a convenient mechanism for setting a scroll bar to its minimum or maximum value using the keyboard.

SelectionBox

	y	n/a	n		
Required	❑	❑	❑	**7-57:**	If your application uses a selection box, it is composed of at least a text component for the selected alternative and a list component above the text component for presenting alternatives. The list uses either the single selection or browse selection model. Selecting an element from the list places the selected element in the text component. This specification ensures the consistent appearance and operation of a selection box across applications.
Required	❑	❑	❑	**7-58:**	The list navigation actions Up Arrow, Down Arrow, Control+Begin, and Control+End are available from the text component for moving the cursored element within the list and thus changing the contents of the text. These actions provide a convenient way to choose an element from the list while focus remains in the text component.

SpinBox

Required	❑	❑	❑	**hw:**	Present the items as a ring of items that wrap. For example, if a user is at the largest number and presses the up arrow, the smallest number is displayed and vice versa so that the user can spin through all the items by pressing the same arrow.
Required	❑	❑	❑	**hx:**	Move through the items in a spin box as shown in Table 10-5 on page 251.

Table 10-5 Navigation in a Spin Box

Movement	Keys	Example
Toward the beginning of the list	left arrow, down arrow	Chronological: If Tuesday is displayed, move to Monday when the user presses the left or down arrow. Magnitude: If 15 is displayed, move to 14 when the user presses the left or down arrow.
Toward the end of the list	right arrow, up arrow	Chronological: If Tuesday is displayed, move to Wednesday when the user presses the right or up arrow. Magnitude: If 15 is displayed, move to 16 when the user presses the right or up arrow.

y n/a n

Recommended ❏ ❏ ❏ **hy:** Values can be set using the arrow buttons or through keyboard input. Values should be evaluated immediately upon entry. If a value is entered that is already in the list, scroll to the position of that entry in the list.

Recommended ❏ ❏ ❏ **hz:** If entry of non-listed items is permitted, use the following behavior. When a new value is entered, scroll the list to the position appropriate for the new entry. If the user scrolls off the new entry, scroll to the next appropriate value in the list and the keyboard-entered value is lost.

Recommended ❏ ❏ ❏ **ia:** On entry of an invalid value, an auditory warning and error message should be provided.

Text

	y	n/a	n		
Required	❑	❑	❑	**7-59:**	In a multiline text component, the Up Arrow key moves the location cursor up one line, and the Down Arrow key moves the location cursor down one line. In a single-line text component, the Up Arrow key navigates upward to the previous component, and the Down Arrow key navigates downward to the next component, if the text component is designed to act like a basic control. The up and down arrow keys provide a uniform means of navigation within text components.
Required	❑	❑	❑	**7-60:**	The Left Arrow key moves the location cursor left one character, and the Right Arrow key moves the location cursor right one character. The Left Arrow and Right Arrow keys offer a consistent way of navigating within text components.
Required	❑	❑	❑	**7-61:**	In a text component used generally to hold multiple words, Control+Right Arrow moves the location cursor to the right by a word, and Control+Left Arrow moves the location cursor to the left by a word. Control+Right Arrow and Control+Left Arrow provide a uniform way of navigating by words in a text component. Moving right by a word means that the location cursor is placed before the first character that is not a space, tab, or newline character after the next space, tab, or newline. Moving left by a word means that the location cursor is placed after the first space, tab, or newline character preceding the first previous character that is not a space, tab, or newline.
Required	❑	❑	❑	**7-62:**	In a text component used generally to hold multiple words, the Begin key moves the location cursor to the beginning of the line, and the End key moves the location cursor to the end of the line. These keys allow the user to move quickly to the beginning or end of a line of text in a text component.
Required	❑	❑	❑	**7-63:**	In a multiline text component, Control+Begin moves the location cursor to the beginning of the file, and Control+End moves the location cursor to the end of the file.

These keys permit the user to move quickly to the beginning or end of a file in a text component.

	y	n/a	n		
Required	❑	❑	❑	**7-64:**	Your application uses Spacebar or Shift+Spacebar to insert a space in a text component. Modifying these with Control invokes the normal selection function.

This specification ensures that selection is available from the keyboard in a text component.

Required ❑ ❑ ❑ **7-65:** Return in a multiline text component inserts a carriage return. The Enter key or Control+Return invokes the default action.

This specification ensures that activation is available from the keyboard in a text component.

Required ❑ ❑ ❑ **7-66:** In a multiline text component, Tab is used for tabbing. In a single-line text component, Tab is used either for tabbing or to move to the next field.

Tab is used for tabbing in multiline text.

Required ❑ ❑ ❑ **7-67:** If a text component supports replace mode, insert toggles between insert mode and replace mode.

By default, the component starts in insert mode, where the location cursor is between two characters. In insert mode, typing a character inserts the character at the position of the location cursor.

In replace mode, the location cursor is on a character. Typing a character replaces the current character with that newly entered character and moves the location cursor to the next character, selecting it.

These rules ensure the uniform operation of a text component with a replace mode.

Required ❑ ❑ ❑ **7-68:** Your application uses BSelect Click 2 to select text a word at a time.

Double-clicking with mouse button 1 provides a convenient mechanism for selecting words in a text component.

Gauge

	y	n/a	n		
Required	❑	❑	❑	**ib:**	A gauge is similar to a scale except that a gauge is a display-only device with no user interactions. The appearance of a gauge is similar to a scale, but the gauge lacks a scale slider.
Optional	❑	❑	❑	**ic:**	Despite being a display-only device, a gauge should get keyboard focus so that the user can access Help or Settings for that control.

Accessibility

Recommended	❑	❑	❑	**id:**	All application functions are accessible from the keyboard.
Recommended	❑	❑	❑	**ie:**	Colors should not be hard coded.
Recommended	❑	❑	❑	**if:**	Graphic attributes, such as line, border, and shadow, should not be hard coded.
Recommended	❑	❑	❑	**ig:**	Font sizes and styles should not be hard coded.
Recommended	❑	❑	❑	**ih:**	All application code uses descriptive names for widgets. Such descriptive names for widgets using graphics instead of text (for example, palette items and icons) allow screen reading software to provide descriptive information to blind users.
Recommended	❑	❑	❑	**ii:**	Interactions do not depend upon the assumption that a user will hear an audible notification.

	y	n/a	n		
Recommended	❑	❑	❑	**ij:**	Where appropriate, users can choose to receive cues as audio or visual information.
Recommended	❑	❑	❑	**ik:**	The application does not overuse or rely exclusively on audible information.
Recommended	❑	❑	❑	**il:**	Users can choose to configure the frequency and volume of audible cues.
Recommended	❑	❑	❑	**im:**	Tear-off menus and user configurable menus for key application features may be provided for users with language and cognitive disabilities.

y n/a n

Recommended ❑ ❑ ❑ **in:** Application keymappings do not conflict with existing system-level key mappings reserved for access features in the X Windows server as shown in Table 10-6.

Table 10-6 Keyboard Mappings for Server-Level Access Features

Keyboard Mapping	Reserved For
Five consecutive clicks of Shift key	On/Off for StickyKeys
Shift key held down 8 seconds	On/Off for SlowKeys and RepeatKeys
Six consecutive clicks of Control key	On/Off for screen reader numeric keypad functions.
Six consecutive clicks of Alt key	Reserved for future access use

Keyboard Functions A

Table A-1 Keyboard Functions and Key Engravings

Function	Key Engraving (US Keyboard)	Description
Activation	Select, spacebar, or Control+space bar	Starts the action associated with a component (for example, an action choice).
Backspace	Backspace	Deletes one character to the left of the cursor or the current selection.
Backtab	Shift+Tab or Backtab	Moves the cursor to the previous field. The cursor is positioned either on the previous choice in that field or on the currently set choice in that field. The cursor moves from right to left and bottom to top. At the top-leftmost field, the cursor moves to the bottom-rightmost field. Within a text-entry field, backtab moves the cursor to the character position defined by the previous tab stop.
Beginning of data	Control+Home	Moves the cursor to the top-leftmost position in the current field.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Beginning of line	Home	Moves the cursor to the leftmost choice in a group of choices, or to the beginning of the current line in a text-entry field.
Cancel	Esc or Cancel	Removes the window without applying any changes that were not previously applied in that window.
Cancel direct manipulation	Esc or Cancel	Cancels the direct manipulation operation.
Clear	Delete or no assignment[1]	Removes selected element or group of elements from window without compressing the space previously occupied by the element or group of elements.
Close	Alt+F4	Closes active window.
Context-sensitive help	F1, Shift+F1, Help	Displays context-sensitive help for the element relative to the current context, such as the cursor position or the process currently in progress.
Copy	Control+C, Control+Insert	Produces a duplicate of the selected element or group of elements and places it on the clipboard.
Cut	Control+X, Shift+Delete	Removes the selected element or group of elements to the clipboard.
Default action	Enter, Return	If the cursor is on a component that can be activated, performs that action. If the cursor is on a component that does not support activation performs the default action for that window.
Delete next character	Delete[1]	Deletes the next character in text.
Delete previous character	Backspace	Deletes the previous character in text.
Delete selection	Delete[1]	Removes a selected element or group of elements and, when appropriate for the window, compresses the space it occupied.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Deselect all	Control+ Backslash	Removes selected-state emphasis from all elements in the active window.
Display drop-down list or drop-down combination box	Alt+down arrow	Displays or removes the list for the drop-down list or the drop-down combination box.
End of data	Control+End	Moves the cursor to the bottom-rightmost position in the current field.
End of line	End	Moves the cursor to the rightmost choice in a group of choices, or to the end of the current line in a text-entry field.
Extend selection to beginning of data	Control+Shift +Home	Extends the selection to the beginning of the data.
Extend selection to beginning of the line	Shift+Home	Extends the selection to the beginning of a line.
Extend selection down	Shift+down arrow	Extends the selection down a line.
Extend selection to end of data	Control+Shift+ End	Extends the selection to the end of the data.
Extend selection to end of line	Shift+End	Extends the selection to the end of a line.
Extend selection to the left	Shift+left arrow	Extends the selection to the left.
Extend selection down one page	Shift+Page Down	Extends the selection down a page.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Extend selection to the left one page	Control+Shift+ Page Up	Extends the selection a page to the left.
Extend selection to the right one page	Control+Shift+ Page Down	Extends the selection a page to the right.
Extend selection up one page	Shift+Page Up	Extends the selection up one page.
Extend selection down one paragraph	Control+Shift+ down arrow	Extends the selection down one paragraph.
Extend selection up one paragraph	Control+Shift+ up arrow	Extends the selection up one paragraph.
Extend selection to the right	Shift+right arrow	Extends the selection to the right.
Extend selection	Shift+spacebar, Control+Shift+ spacebar	Extends the selection to the cursor position (not valid in text).
Extend selection up	Shift+up arrow	Extends the selection up a line.
Extend selection word left	Control+Shift+ left arrow	Extends the selection one word to the left.
Extend selection word right	Control+Shift+ right arrow	Extends the selection one word to the right.
Help (Overview help)	F2 (in a Help window)	Displays a brief overview of each action, task, or both, that a user can perform within the window.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Help index	F11 (in a Help window)	Displays an alphabetic listing of help topics for an object or a product.
Help	F1	Displays context-sensitive help for the item that contains the cursor
Hide	Alt+F9	Removes the window and all associated windows from the screen.
Keyboard help	F9 (in a Help window)	Displays a listing of all the key assignments for an object or a product.
Maximize	Alt+F10	Enlarges the window to its largest possible size.
Minimize	Alt+F9	Reduces the window to its smallest possible size and removes all of the windows associated with that window from the screen.
Move	Alt+F7	Allows a user to move a window to a different location.
Move cursor	Arrow keys or Control+arrow keys	Moves the cursor left, right, up, or down. At the last choice, the cursor wraps. For example, at the bottom-most choice, the cursor wraps to the top-most choice to the right.
Move cursor to and from menu bar	F10	Moves the cursor from within a window to its menu bar or from the menu bar to within the window.
Move cursor to the next field	Tab or Control+Tab	Moves the cursor to the next field.
Move cursor to the previous field	Shift+Tab or Control+Shift+Tab	Moves the cursor to the previous field.
Move cursor to the next associated window (within a window family)	Alt+F6	Moves the input focus to the next window within a window family.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Move cursor to the previous associated window (within a window family)	Alt+Shift+F6	Moves the input focus to the previous window within a window family.
Move cursor forward between unassociated windows	Alt+Esc	Moves the input focus forward between the groups of associated windows displayed from different objects (if more than one object is displayed by an object).
Move cursor between window families	Alt+Tab	Moves the input focus between open, but not hidden, window families.
Move cursor backwards between window families	Alt+Shift+Tab	Moves the input focus backwards between open, but not hidden, window families.
Move cursor to another window pane	F6	Moves the cursor in a clockwise direction from one window pane to the next.
New line	Enter or Return	Moves the cursor to the next line of text in replace mode or adds a new line in text in insert mode.
Page down	Page Down	A scrolling action that displays information below the currently visible window area.
Page left	Control+Page Up	A scrolling action that displays information to the left of the currently visible window area.
Page right	Control+Page Down	A scrolling action that displays information to the right of the currently visible window area.
Page up	Page Up	A scrolling action that displays information above the currently visible window area.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Paragraph down	Control+down arrow	A scrolling action that displays one paragraph below the currently visible window area.
Paragraph up	Control+up arrow	A scrolling action that displays one paragraph above the currently visible window area.
Paste	Control+V or Shift+Insert	Copies the contents of the clipboard into the window at the specified location.
Pop-up menu	Shift+F10	Displays a pop-up menu for the indicated element or group of selected elements.
Redo	Shift+Alt+Backspace	Reverses the effect of the last applied undo action.
Refresh now	F5, Control+R	Updates the window to reflect the underlying data.
Restore	Alt+F5	Returns the window to the size it was and the position it was in before the user minimized or maximized the window.
Select all	Control+Slash	Selects all elements in active window.
Select element on which cursor is positioned	Spacebar (if it is not assigned to any other function) or Control+ spacebar	Selects element on which cursor is positioned.
Size	Alt+F8	Allows a user to change the size of the window.
Space	spacebar or Shift+spacebar	Inserts a space in text.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	Key Engraving (US Keyboard)	Description
Tab	Tab	Moves the cursor to the next field. The cursor is positioned either on the first choice in that field or on the currently set choice in that field. The cursor moves from left to right and top to bottom. At the bottom-rightmost field, the cursor moves to the top-leftmost field. Within a text-entry field, Tab moves the cursor to the character position defined by the next tab stop or resets the tab.
Toggle between insert and replace modes in text entry	Insert	Toggles between insert and replace modes.
Toggle in or out of keyboard add mode when in extended selection mode	Shift+F8	Toggles in or out of keyboard add mode when extended selection is provided for a view. The initial deselection is bypassed and the new selected objects are added or removed from the current group of selected objects.
Tutorial	Shift+F2 (in a Help window)	Displays online educational information.
Undo	Alt+Backspace, Control+Z	Reverses the action of the most recently performed user action.
Using help	Shift+F10 (in a pop-up window)	Displays help information that describes how to use the help facility.
Window list	Control+Esc	Displays the window list window from the window menu.

Table A-1 Keyboard Functions and Key Engravings *(Continued)*

Function	**Key Engraving (US Keyboard)**	**Description**
Window menu	Alt + Spacebar or Shift+Esc	Displays window menu.
Word left	Control+left arrow	Moves the cursor to the beginning of the word to the left of the cursor.
Word right	Control+right arrow	Moves the cursor to the beginning of the next word to the right of the cursor.

1. Assign the Delete key to either the Delete or Clear function, if only one is provided. If both Delete and Clear functions are provided, assign the Delete key to the Delete function

Table A-2 Mnemonic Assignments for menu Choices

Choice	**Mnemonic**	**Location**
About	A	Help menu
Clear	E	Edit menu
Close	C	File menu and Window menu
Copy	C	Edit menu
Copy Link	K	Edit menu
Copy To	C	Selected menu
Cut	T	Edit menu
Delete	D	Object-type menu and Edit menu
Deselect All	L	Edit menu
Exit	X	File menu
Find	F	Edit menu
Icon	C	View menu
Include	I	View menu

Table A-2 Mnemonic Assignments for menu Choices *(Continued)*

Choice	Mnemonic	Location
Index	I	Help menu
Keyboard	K	Help menu
Lower	L	Window menu
Maximize	X	Window menu
Minimize	N	Window menu
Mouse	M	Help menu
Mouse and Keyboard	M	Help menu
Move	M	Window menu
Move To	M	Selected menu
New	N	File menu
Occupy All Workspaces	A	Window menu
Occupy Workspace	O	Window menu
On Item	O	Help menu
Open	O	File menu
Overview	V	Help menu
Paste	P	Edit menu
Paste Link	L	Edit menu
Print	P	File menu
Promote	M	Edit menu
Properties	P	Selected menu
Put in Workspace	T	Selected menu
Redo	R	Edit menu
Reference	R	Help menu
Refresh	E	View menu
Reselect	R	Edit menu
Restore	R	Window menu

Table A-2 Mnemonic Assignments for menu Choices *(Continued)*

Choice	**Mnemonic**	**Location**
Save	S	File menu
Save as	A	File menu
Select All	S	Edit menu
Select Pasted	A	Edit menu
Size	S	Window menu
Sort	S	View menu
Table of Contents	C	Help menu
Tasks	T	Help menu
Tutorial	L	Help menu
Undo	U	Edit menu
Unoccupy Workspace	U	Window menu
Using Help	U	Help menu

Table A-3 Keyboard Mappings for Space Bar and Enter Key

Element	**Space bar**	**Enter key**
menu	Activates current item	Activates current item
pushbutton	Activates button	Performs default action if default action exists for the window containing the element
text field	Adds a space	Performs default action if default action exists for the window containing the element
list item	Selects current item	Performs default action if default action exists for the window containing the element
icon	Selects	Performs default action if default action exists for the window containing the element

Mouse Functions B

On a one-, two-, or three- button mouse, the mouse buttons are assigned to various functions which are defined below.

Some two-button mice use chording as a way to simulate a third mouse button. If so, treat as a three-button mouse, where mouse button 3 is mouse button 1+mouse button 2 (chorded).

Motif supports two different mouse models:

- Separate Selection and Transfer:
  - Mouse button 1 is used only for selection and activation.
  - Mouse button 2 is used only for data transfer and direct manipulation.
- Integrated Selection and Transfer:
  - Mouse button 1 is used both for selection and activation and for data transfer and direct manipulation.

The way the mouse buttons are assigned depends upon the number of mouse buttons available, as well as whether selection and transfer are integrated or are separate.

Regardless of the way that the buttons are assigned, a number of "virtual" mouse buttons are defined:

SELECT	Used for selection and activation. SELECT is always mouse button 1 (the leftmost button,for a right-handed person).

ADJUST	Used for adjusting and selection. ADJUST is always Shift+mouse button 1. In addition, on a three-button mouse, with integrated selection and transfer, ADJUST may optionally be assigned to mouse button 2.
TRANSFER	Used for data transfer and manipulation operations. With separate selection and transfer, TRANSFER is always assigned to mouse button 2. With integrated selection and transfer, TRANSFER is mouse button 1 (integrated with SELECT), and on a three-button mouse, may also optionally be assigned to mouse button 2.
MENU	Used to obtain pop-up menus. On a three-button mouse, MENU is always assigned to mouse button 3. On a two-button mouse with integrated selection and transfer, MENU is assigned to mouse button 2. Otherwise, MENU is assigned to Alt+mouse button 1.

That is, on a two- or three-button Mouse, with Separated SELECTION and TRANSFER, the virtual mouse buttons are assigned as follows:

SELECT	mouse button 1
ADJUST	Shift+mouse button 1
TRANSFER	mouse button 2
MENU	mouse button 3 on a three-button mouse, or Alt+mouse button 1 on a two-button mouse

On a one-, two-, or three-button mouse, with Integrated SELECTION and TRANSFER, the virtual mouse buttons are assigned as followed:

SELECT	mouse button 1 (integrated with TRANSFER)
ADJUST	Shift+mouse button 1. Optionally mouse button 2 on a three-button mouse
TRANSFER	mouse button 1 (integrated with SELECT) Optionally mouse button 2 on a three-button mouse.
MENU	mouse button 3 on a three-button mouse, or mouse button 2 on a two-button mouse, or Alt+mouse button 1 on a one-button mouse.

Note – On a three-button mouse, with integrated selection and transfer, if neither ADJUST nor TRANSFER are assigned to mouse button 2, mouse button 2 may be used for application-defined purposes.

Mouse Operations and Functions

Table B-1 Mouse Operations and Functions

Operation Name	Function
Activate	Activates a control that doesn't have selections.
Default Activate Open	Selects and performs default action on item Open.
Open	Opens view corresponding to icon Open.
Manipulate	Manipulates nonselectable aspects of the interface (for example: scroll).
Move Cursor	Moves cursor to component or element.
Spring-Loaded Pull-down Menu	Displays persistent pull-down menu from cascade button.
Persistent Pull-down Menu	Displays persistent pull-down menu from cascade button.
Point Select	Selects an item if over one, deselecting other items (browse and extended selection).
Browse Select	Shows which items can be selected, selecting one over which pointer is released.
Group Click Select	Selects a range or area of elements.
Group Swipe Select	Selects a range or area of elements.
Point Toggle	[select mode] Toggles selection state of an item (extended selection).
Group Click Toggle	[select mode] Toggles elements in range or area (extended selection).
Group Swipe Toggle	[select mode] Toggles elements in range or area (extended selection).

Table B-1 Mouse Operations and Functions *(Continued)*

Operation Name	Function
Point Toggle	[toggle mode] Toggles selection state of an item (single and multiple selection).
Group Click Toggle	[toggle mode] Toggles elements in range or area (multiple selection).
Adjust Click	Adjusts current selection region.
Adjust Swipe	Adjusts current selection region.
Select Word	Selects a word in text.
Range Click Select Word	Selects a range of words
Range Swipe Select Word	Selects a range of words.
Toggle Word	Toggles selection of a word.
Range Click Toggle Word	Toggles a range of words.
Range Swipe Toggle Word	Toggles a range of words.
Adjust Click Word	Adjusts selection to word boundary.
Adjust Swipe Word	Adjusts selection in word increments.
Primary Copy	Copies primary selection to pointer.
Primary Move	Moves primary selection to pointer.
Primary Link	Links primary selection to pointer.
Quick Copy	Makes and copies secondary selection to destination.
Quick Move	Makes and move secondary selection to destination.
Quick Link	Makes and link secondary selection to destination.
Drag Transfer	Transfers dragged items to pointer (usually move).
Drag Copy	Copies dragged items to pointer.
Drag Move	Moves dragged items to pointer.
Drag Link	Links dragged items to pointer.
Spring-Loaded Pop-up Menu	Displays spring-loaded pop-up menu.
Persistent Pop-up Menu	Displays persistent pop-up menu.

Select and Adjust Binding

SELECT — This is the virtual mouse button used for selection and activation. SELECT is always mouse button 1 (the leftmost button, for a right-handed person).

ADJUST — This is the virtual mouse button used for adjusting a selection. ADJUST is always Shift+mouse button 1. In addition, on a three-button mouse, with integrated selection and transfer, ADJUST may optionally be assigned to mouse button 2.

Table B-2 Select and Adjust Key Bindings

Operation Name	Key Bindings
Activate	SELECT Click
Default Activate	SELECT Double-Click
Open	SELECT Double-Click
Manipulate	SELECT Press, Move, Release
Move Cursor	Control+SELECT Click
Spring-Loaded pull-down Menu	SELECT Press
Persistent pull-down Menu	SELECT Click
Point Select	SELECT Click
Browse Select	SELECT Press, Move, Release
Group Click Select	SELECT Click, Move, ADJUST Click
Group Swipe Select	SELECT Press, Move, Release
Point Toggle	[select mode] Control+SELECT Click
Group Click Toggle	[select mode] Control+SELECT Click, Move, ADJUST Click
Group Swipe Toggle	[select mode] Control+SELECT Press, Move, Release
Point Toggle	[toggle mode] SELECT Click
Group Click Toggle	[toggle mode] SELECT Click, Move, ADJUST Click
Group Swipe Toggle	[toggle mode] SELECT Press, Move, Release

Table B-2 Select and Adjust Key Bindings *(Continued)*

Operation Name	**Key Bindings**
Adjust Clock	ADJUST Click
Adjust Swipe	ADJUST Press, Move, Release
Adjust Click	Control+ADJUST Click
Adjust Swipe	Control+ADJUST Press, Move, Release
Select Word	SELECT Double-Click
Range Click Select Word	SELECT Double-Click, Move, Adjust Click
Range Swipe Select Word	SELECT Double Press, Move, Release
Toggle Word	Control+SELECT Double-Click
Range Click Toggle Word	Control+SELECT Double-Click, Move, ADJUST Click
Range Swipe Toggle Word	Control+SELECT Double-Click, Move, Release
Adjust Click Word	ADJUST Double-Click
Adjust Swipe Word	ADJUST Double Press, Move, Release
Adjust Click Word	Control+ADJUST Double-Click
Adjust Swipe Word	Control+ADJUST Double Press, Move, Release

Transfer Bindings

TRANSFER

This is the virtual mouse button which may be used for data transfer and manipulation operations. On a two- or three-button Mouse, with separate Selection and Transfer, TRANSFER is always assigned to mouse button 2. On a one-, two-, or three-button Mouse, with Integrated Selection and Transfer, TRANSFER is always assigned to mouse button 1 (integrated with SELECT). In addition, on a three-button Mouse, with Integrated Selection and Transfer, TRANSFER may optionally be assigned to mouse button 2.

Table B-3 Key Bindings When TRANSFER is Assigned Mouse Button 2

Operation Name	Key Bindings
Manipulate	TRANSFER Press, Move, Release
Primary Copy	TRANSFER Click
Primary Copy	Control+TRANSFER Click
Primary Move	Shift+TRANSFER Click
Primary Link	Control+Shift+TRANSFER Click
Quick Copy	Alt+TRANSFER Press, Move, Control+Release
Quick Move	Alt+TRANSFER Press, Move, Shift+Release
Quick Link	Alt+TRANSFER Press, Move, Control+Shift+Release

The set of bindings in Table B-4 are always defined. When selection and transfer are integrated, Style Guide rules indicate when these bindings are used for transfer vs. selection.

Table B-4 Key Bindings

Operation Name	Key Bindings
Drag Transfer	TRANSFER Press, Move, Release
Drag Copy	TRANSFER Press, Move, Control+Release
Drag Move	TRANSFER Press, Move, Shift+Release
Drag Link	TRANSFER Press, Move, Control+Shift+Release

Menu Bindings

MENU is the virtual mouse button used to obtain popup menus. On a two-or three-button mouse, with separate SELECTION and TRANSFER:

MENU	mouse button 3 on a three- button mouse, or Alt+mouse button 1 on atwo-button mouse.

On a one-, two-, or three-button mouse, with integrated selection and transfer:

MENU	mouse button 3 on a three-button mouse, or mouse button 2 on a two-button mouse, or Alt+mouse button 1 on a one-button mouse.

Table B-5 Menu Bindings

Operation Name	Key Bindings
Spring-Loaded Popup Menu	MENU Press
Persistent Popup Menu	MENU Click
Spring-Loaded pull-down Menu	MENU Press
Persistent pull-down Menu	MENU Click

Index

D

L

M

N

O

P

Q

R

S