Toolkit Transition Issues 5

This chapter describes toolkit terminology, common widgets, CDE exclusive widgets, and high-level widgets, and provides information about the different libraries.

OPEN LOOK Versus CDE Motif Toolkits	*31*
XView Libraries Versus CDE Motif Libraries	*35*
OLIT Libraries Versus CDE Motif Libraries	*39*

Note – Everything said in this chapter that refers to the CDE Motif toolkit also applies to the Solaris 2.4 Motif toolkit.

OPEN LOOK Versus CDE Motif Toolkits

When you compare an OPEN LOOK application with its CDE Motif counterpart, a few contrasting visual elements are immediately apparent. For example, the OPEN LOOK buttons are round whereas the CDE Motif buttons are square. The shading applied to buttons and other objects for a three-dimensional appearance are also different. Although such cosmetic elements do not affect a program's behavior, and can often be disregarded when porting, your application will not be CDE style-guide compliant if you deviate from the CDE look.

5

Several differences are often significant in a conversion effort. The most critical of these features and other GUI elements are summarized in three sections:

- Aspects of the OPEN LOOK user interface that are missing from CDE Motif
- Aspects of CDE Motif that do not appear in the OPEN LOOK user interface
- Features or other elements that appear in both specifications but are implemented differently

See the Preface for a list of style guides and other references that describe the OPEN LOOK and CDE Motif GUIs.

Features Only in the OPEN LOOK User Interface

The following features are found in the OPEN LOOK user interface and are implemented in XView or OLIT (or both) but do not appear in CDE Motif:

- Split window control
- Scrollbar anchors
- Defaults of the menu
  - Automatic default; shortcut method for selecting the default
  - Default menu item indicated by a *ring*
- Font chooser widget
- Pointer warping in notices
- Can drop onto minimized windows (icons)

Features Only in the CDE Motif User Interface

The following features are found in CDE Motif and not in the OPEN LOOK user interface:

- Front Panel
- Tools for producing and registering help volumes with hypertext capabilities
- Additional user interface objects (*widgets*) available for applications and specified in the CDE Style Guide

Notable Implementation Differences Between Toolkits

Many features are roughly equivalent in the OPEN LOOK user interface and CDE Motif but have significant implementation differences. The following are the most important differences:

- Tear off Menus
- Input focus indicators
- Widget classes (sliders and gauges versus scales, for example)
- Secondary text selections

Other significant differences include the following:

- Keyboard bindings
- The window manager controls associated with each window
- Internalization
- Mouse button behavior

The remainder of this section provides information on some of these features.

Tear-off Menus

In CDE Motif, tear-off menus replace the *pinned menus* of the OPEN LOOK user interface. Selecting the dashed line on the top of the menu "tears-off" the menu.

Secondary Text Selection

The OPEN LOOK secondary text selection is roughly analogous to the Motif quick transfer mechanism.

Window Controls

One of the most critical implementation differences involves window controls. When the Motif user presses the Window menu button in the upper left corner of the title bar, or the OPEN LOOK user presses the MENU button anywhere on the window background (including the header), a menu is displayed. The options offered under the two GUIs introduce a key contrast.

The Motif *Window menu* offers a choice of Restore, Move, Size, Minimize, Maximize, Lower, and Close. The OPEN LOOK *base Window menu* offers Close, Full Size, Properties, Back, Refresh, and Quit. The two lists are fundamentally the same, but have very different effects.

In the OPEN LOOK user interface, the Close option minimizes (iconifies) the window, and Quit terminates the application.

In Motif, the Minimize option minimizes the window, and the Close option terminates the application. Many users familiar with the OPEN LOOK user interface have found themselves exiting a Motif program when their intent was to close its window to an icon.

Mouse Button Behavior

The structure of the mouse buttons is very similar in both specifications; however, the difference is significant enough to cause some confusion.

Table 5-1 shows the default left-to-right mapping of the three OPEN LOOK mouse buttons.

Table 5-1 OPEN LOOK Mouse Buttons

BUTTON	Description
SELECT	Specifies an object or manipulates objects and controls, drag
ADJUST	Extends or reduces the number of selected objects
MENU	Displays a menu associated with the pointer location or specified object

The three Motif mouse button assignments, described in Table 5-2, also start by default with the left mouse button.

Table 5-2 Motif Mouse Buttons

Button	Description
BSelect	Selects, activates, and sets the location cursor, drag
BTransfer	Moves and copies elements, drag and drop transfer. Can be customized to be the OPEN LOOK Adjust button.
BMenu	Displays menus

XView Libraries Versus CDE Motif Libraries

This section compares the XView and CDE Motif libraries.

Terminology

The XView toolkit and the Xt (OLIT and Motif) toolkits use the following terminology:

XView	Xt (OLIT and Motif)
Package	Widget
Attribute	Resources
Frame	Shell

XView is based directly on Xlib, whereas Motif is based on the Intrinsics (Xt) toolkit and the Intrinsics toolkit is based on Xlib.

XView	OLIT	Motif
XLib	Xt	Xt
	XLib	XLib

Because of this fundamental difference, the basic library functions to initialize the environment and create, modify and destroy graphical objects, are different, as shown in the following examples:

XView	Motif/Xt
`xv_init()`	`XtAppInitialize()`
`xv_create()`	`XtCreateWidget()`
`xv_set()`	`XtSetValues()`
`xv_destroy()`	`XtDestroyWidget()`

Functions that deal with event handling callbacks, and internationalization features, for example, get more complicated. For these features, simple one-to-one correspondence does not exist.

5

OPEN LOOK Versus Motif Toolkit Architecture

OLIT and Motif have very similar architectures whereas XView and Motif do not. When migrating from XView to Motif, you should note these toolkit differences.

The XView toolkit implements both the user interface objects, called *packages*, and the routines and processes that hold the interface together (creation routines, event processing, and so on), while Motif and OLIT implement basically just the user interface object, *widgets*, leaving the routines and processes to the Intrinsics library. For example, `xv_init()` is a function that is in the XView library. The corresponding function to use for OLIT or Motif programming, `XtAppInitialize()`, is in the Xt Intrinsics library.

The three toolkits represent two different GUIs. The appearance of the XView and OLIT toolkits are similar, whereas the appearance of CDE Motif is noticeably different. Although there is a rough one-to-one correspondence between the function calls in the libraries, the *behavior* of parallel programs is different. That is, even after an OLIT (or XView) program has been converted to use the CDE Motif library, it still to some degree has an OPEN LOOK appearance. A program in such a state does not adhere to either style completely.

Programming Model

Although the APIs are different, both XView and Motif are based on the same object-oriented methodology for programming a user interface:

- Initialize the toolkit
- Instantiate user interface objects
- Register callbacks on the user interface objects
- Enter event loop, waiting for user to generate events on the user interface objects

The overall structure of a program being ported from XView to Motif can remain intact even though all the function calls must be converted from one API to the other.

Common Types of User Interface Objects

Both XView and Motif are user interface toolkits that support some common types of user interface objects. However, XView implements many of these objects at a higher level that requires more than one Motif widget to produce the equivalent of a single XView object.

Table 5-3 lists the basic mapping of common objects for XView and its equivalent CDE Motif widget:

Table 5-3 Basic Mapping of Common Objects.

XView Package	Equivalent CDE Motif Widget
Base Frame	`XmTopLevelShell` + `XmMainWindow`
Command Frame	`XmDialogShell` + `XmBulletinBoard`
Notice	`XmDialogShell` + `XmMessageBox` (MessageDialog)
Canvas	`XmScrolledWindow` + `XmDrawingArea`
Panel	`XmBulletinBoard` or `XmForm`
Panel Button	`XmPushButton`
Menu Button	`DtMenuButton` (not accessible in OpenWindows environment)
Abbrev Menu Button	`XmRowColumn` (Option Menu)
Checkbox	`XmRowColumn` + `XmToggleButtons` (CheckBox)
Exclusive Choice	`XmRowColumn` + `XmToggleButtons` (Radio Box)
Scrolling List	`XmScrolledWindow` + `XmList`
Message	`XmLabel`
Slider	`XmScale`
Text Field	`XmTextField`
Numeric Text Field	`DtSpinBox` (not accessible in OpenWindows environment)
Text SubWindow	`XmScrolledWindow` + `XmText` (ScrolledText)
TTY SubWindow	`XmScrolledWindow` + `XmTermPane`; `DtTerm` (not accessible in OpenWindows environment)

Table 5-3 Basic Mapping of Common Objects. *(Continued)*

XView Package	Equivalent CDE Motif Widget
Scrollbar	`XmScrollBar`
Popup Menu	`XmMenuShell` + `XmRowColumn` (Popup Menu)
Pulldown Menu	`XmMenuShell` + `XmRowColumn` (Pulldown Menu)
Pullright Menu	`XmMenuShell` + `XmRowColumn` (Pulldown Menu)
File Chooser	`XmDialogShell` + `XmFileSelectionBox`

Differences Between XView and CDE Motif

XView abstracts a number of other X11 functions that Motif (Xt) does not. In order to get equivalent functionality in Motif, these must be re-coded with direct Xlib calls.

The XView packages with no Motif equivalents are:

- Icon
- Server Image
- Colormap Segment (CMS)
- Cursor
- Fullscreen
- Server
- Font

Additionally, *no* CDE Motif equivalent exists for the OPEN LOOK user interface functionality that splits windows into different views. To implement this function, you must subclass one of the Motif Manager widgets.

Some XView functions can be coded with the Motif/Xt API; however, these APIs are significantly different, and require complete redesign and recoding. These functions are:

XView API	Motif/Xt
Selection service	Xt selection API
Drop target package	Motif drag-and-drop API[1]
Notifier	Xt event management API

1. If you are porting to CDE, use the CDE drag-and-drop API. It is an extension of the Motif 1.2 drag-and-drop API and is easier to use.

X Resources

The X resources that control the behavior and appearance of XView and Motif applications are different. XView objects may not necessarily have instance names attached to them, unlike Motif/Xt objects. XView resources are independent of instance names. For example, `Window.Foreground.Color` affects all relevant XView objects.

Motif/Xt resources contain either class or instance names, for example, `mainframe.control_panel.button1.foreground`. To make Xt resources affect more than one object, use wildcards and class names.

External Files

XView reads in several files at start-up time ranging from message files to application specific default files containing X resources. The content and location of some of these files are different from Motif. (For example, Motif/Xt does not read in a message domain file (`.mo` file) under `$OPENWINHOME`.) The internationalization messaging scheme is completely different for Motif.

OLIT Libraries Versus CDE Motif Libraries

Solaris CDE Motif and OLIT support a number of convenience routines that are useful in manipulating objects in the user interface.

Routines Only in OLIT Library

These features are unique to the OLIT library.

- Error-Handling Routines

  OLIT provides a range of routines to enable the application to customize error handling.

- Dynamic Resources

  OLIT provides support for the user to dynamically change the value of certain resources (such as colors or fonts) after an application has been invoked.

5

Routines Only in CDE Motif Libraries

These features are unique to the CDE Motif libraries.

- UIL Support

  CDE Motif supports the engine for using UIL definitions for the user interface layout. This separation allows the user interface to be modified without recompiling the program's executables. OLIT's Devguide solution (`golit`) provides similar functionality by allowing the user interface to be defined in GIL file format; however, the application must be recompiled when the user interface changes.

- Clipboard Routines

  CDE Motif provides a library for managing the clipboard and its selections.

- Widget-Creation Routines

  CDE Motif provides a complete set of routines that create a particular type of widget or group of widgets.

- Compound String Support

  CDE Motif requires the user of compound strings for most text. To support these special string formats, CDE Motif includes a number of routines to create and manipulate compound strings.

Widgets

Both the OLIT and CDE Motif toolkits support a number of common widgets and gadgets with similar functionality, and each supports a number of more exclusive widgets. If a widget is implemented in one toolkit but not in the other, you can often build an equivalent object using multiple widgets in the other toolkit.

Table 5-4 matches the common widget name to the actual class name of the widget in each toolkit.

Table 5-4 Common Widget Mapping

OLIT Class Name	CDE Motif Class Name
BulletinBoard	XmBulletinBoard
DrawArea	XmDrawingArea
ExclusiveChoice + RectButtons	XmRowColumn + XmToggleButtons (Radio Box)
Form	XmForm
Manager	XmManager
MenuButton	DtMenuButton (not accessible in OpenWindows environment)
NoticeShell	XmDialogShell + XmMessageBox
AbbrevMenuButton	XmRowColumn (Option Menu)
PopupWindowShell	XmDialogShell
NonExclusiveRectButton	XmRowColumn + XmToggleButtons (Check Box)
PopupMenuShell	XmMenuShell
Primitive	XmPrimitive
OblongButton	XmPushButton
ControlArea	XmRowColumn
Scrollbar	XmScrollBar
ScrollingList	XmList + XmScrolledWindow
ScrolledWindow	XmScrolledWindow
Slider	XmScale
StaticText	XmLabel
TextEdit	XmText
TextField	XmTextField
RectButton	XmToggleButton

5

Widgets Exclusive to CDE Motif

This section briefly describes the widgets that are exclusive to CDE Motif.

Note – These widgets are also available for Motif development in the OpenWindows environment, except for the `DtTerm`, `DtEditor`, `DtComboBox`, `DtSpinBox`, and `DtMenuButton` widgets.

- `DtTerm`

  This widget provides the functionality required to emulate an ANSI X3.64-1979-style terminal emulator (specifically a DEC VT220-like terminal with extensions).

- `DtEditor`

  This widget provides a programmatic interface for editing services such as cut and paste.

- `DtComboBox`

  This widget is a combination of a text field and a list widget that provides a list of valid choices for the text field. Selecting an item from this list automatically fills the text field with that list item.

- `DtSpinBox`

  This widget is a convenient user interface control that increments and decrements an arbitrary TextField.

- `DtMenuButton`

  This widget is a command widget that complements the menu cascading functionality of an `XmCascadeButton` widget. (OLIT has its own menu button widget, with equivalent functionality to DtMenuButton.)

- `XmArrowButton`

  This button is a primitive push button widget that displays an arrow label.

- `XmCommand`

  This is a manager widget that builds a command box and manages the user-selected commands and command history.

- `XmDrawnButton`

  This button is a primitive push button whose label can be drawn by the program.

- `XmFrame`

  This manager widget is used to parent a single child and enclose that child with a frame or border.

- `XmLabelGadget`

  This gadget is a low-overhead object for read-only text.

- `XmMainWindow`

  This manager widget supports a menu bar, command area, and work area.

- `XmPanedWindow`

  This manager widget implements resizeable panes within a window.

- `XmSelectionBox`

  This widget box allows you to select one item from a list in a general dialog box.

- `FileSelectionBox`

  This widget provides a standard way to select a file (typically for the application to read or write).

The `libDtWidget` library, which contains the `DtComboBox`, `DtSpinBox`, `DtMenuButton` and `DtEditor` widgets, depends directly on the following libraries:

- Xm library for the Motif superclass support
- Xt library for creation and manipulation of widgets
- X library for the base X Window System

Widgets Exclusive to OLIT

This section briefly describes the widgets that are exclusive to OLIT.

- `DropTarget`

  This primitive widget implements both the source and destination ends of drag-and-drop operations.

- `FlatWidget`

  These special widgets manage any number of subobjects within the context of a single widget. When implementing menus or choice objects that contain many subitems, they provide a significant memory savings.

- `FooterPanel`

  This manager widget automatically supports a window with a floating footer area.

- `RubberTile`

  This manager widget allows relative-sizing constraints to be placed on its children.

- `Stub`

  This primitive widget enables you to customize its behavior without subclassing.