Toolkit-Related Transition Issues 5

This chapter outlines the toolkit terminology, common widgets, CDE exclusive widgets, and high-level widgets, and provides information about the different libraries. This chapter contains these sections:

Note – In this chapter, *Solaris Motif* refers to the Solaris 2.4 or Solaris 2.5 Motif toolkits *only* (not the Solaris 2.3 (IXI) Motif toolkit). Follow the CDE Style Guide when developing an application with either of these Solaris Motif toolkits.

OPEN LOOK Versus Solaris and CDE Motif

When you compare an OPEN LOOK application with its Solaris or CDE Motif counterpart, a few contrasting visual elements are immediately apparent. The OPEN LOOK buttons are round, whereas the Solaris and CDE Motif buttons are square. The shading applied to buttons and other objects for a three-dimensional appearance are also different. Such cosmetic elements do not affect a program’s behavior, however, and can often be disregarded when porting. Your application would not be CDE style-guide compliant in the cases where you deviate from the CDE look, however.

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 OPEN LOOK that are missing from Solaris or CDE Motif
- Aspects of Solaris or CDE Motif that do not appear in OPEN LOOK
- 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 Solaris and CDE Motif GUIs.

Only in OPEN LOOK

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

- Split control versus sash Motif and vertical panes
- Secondary text selections
- Scrollbar anchors
- Defaults of the Menu
  - Automatic default, short cut method for selecting the default
  - Default menu item indicated by a *ring*
- Font chooser widgets
- Pointer warping
- Minimized windows no longer are valid drop areas

Only in CDE Motif

The following features are found in Solaris and CDE Motif and not in OPEN LOOK:

- Standard X clipboard cut-and-paste
- Front panel
- Tools for producing and registering help volumes with hypertext capabilities
- Additional user interface objects, *widgets*, available for applications and specified to the CDE Style Guide

Notable Implementation Differences Between Toolkits

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

- Tear off Menus
- Input focus indicators
- Context-sensitive help (applies only to CDE Motif, *not* Solaris Motif)
- Widget classes (sliders and gauges versus scales, for example)

Other significant differences include the following:

- Keyboard bindings
- The window manager controls associated with each window
- The approach to implementing internalization
- The interpretation of specific mouse actions

Tear-off Menus

In Solaris and CDE Motif, tear-off Menus replace the *pinned menus* of OPEN LOOK. Selecting the dashed line on the top of the menu, tears-off the menu.

Context-Sensitive Help

Context-sensitive help is present in each CDE Motif GUI, but the model of *context-sensitivity* differs in each environment. The help viewer in CDE displays context-sensitive help text in response to pressing the F1 key, and also when the On-item option is selected from an application's Help menu.

Note – Context-sensitive help is not available to applications developed for the Solaris platform.

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 title bar), 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 OPEN LOOK, selecting the Close option minimizes the window, and Quit terminates the application.

In Motif, selecting the Minimize option minimizes the window, and selecting the Close option terminates the application. Many users familiar with OPEN LOOK have found themselves exiting a Motif program when their intent was to close it 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 buttons.

Table 5-1 OPEN LOOK 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 buttons, described in Table 5-2, also start by default with the left mouse button.

Table 5-2 Motif Buttons

Button	Description
BSELECT	Selects, activates, and sets the location cursor, drag
BTRANSFER	Moves and copies elements, drag and drop transfer
BMENU	Pops up menus

XView Libraries Versus Solaris 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) and the Intrinsics are based on Xlib.

XView	OLIT	Motif
XLib	Xt	Xt
	XLib	XLib

Because of this fundamental difference, the basic library functions to initialize the environment, 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 correspondence does not exist.

Architecture

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

The XView toolkit implements both the UI objects, called *packages*, and the routines and processes that hold the interface together (creation routines, event-processing, etc.), while Motif and OLIT implement basically just the UI object, *widgets*, leaving the routines and processes to the 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 UI:

- Initialize the toolkit
- Instantiate UI Objects
- Register Callbacks on the UI Objects
- Enter event-loop, waiting for user to generate events on the UI 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.

Commonalities

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

The following table lists the basic mapping of common objects for XView and its equivalent Solaris or CDE Motif Widget:

Table 5-3 Basic Mapping of Common Objects.

XView Package	Equivalent Solaris/CDE Motif Widget(s)
Base Frame	TopLevelShell + MainWindow
Command Frame	DialogShell + BulletinBoard
Notice	DialogShell + MessageBox (*MessageDialog*)
Canvas	ScrolledWindow + DrawingArea
Panel	Bulletin Board
Panel Button	PushButton
Menu Button	MenuButton (only in CDE Motif)
Abbrev Menu Button	
Checkbox	RowColumn + ToggleButtons (*CheckBox*)
Exclusive Choice	Toggle Button
Scrolling List	List
Message	Label
Slider	Scale
Text Field	TextField
Numeric Text Field	DtSpinBox (only in CDE Motif)
Text SubWindow	ScrolledWindow + Text (*ScrolledText*)
TTY SubWindow	ScrolledWindow + TermPane; DtTerm (only in CDE)
Scrollbar	ScrollBar
Popup Menu	MenuShell + RowColumn (*Popup Menu*)
Pulldown Menu	MenuShell + RowColumn (*Pulldown Menu*)
Pullright Menu	MenuShell + RowColumn (*Pulldown Menu*)
File Chooser	DialogShell + FileSelectionBox

Differences

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. A list of these packages follows.

XView Packages with No Motif Equivalents

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

Additionally, *no* CDE Motif equivalent exists for OPEN LOOK to split windows into different views. To implement this function, 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. A list of the these functions follow:

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, but is easier to use.

X Resources and Command Line Options

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

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.

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 that of Motif. For example, Motif/Xt does not read in a message domain file (`.mo` file) that lives under $OPENWINHOME. The internationalization messaging scheme is completely different for Motif.

OLIT Libraries Versus Solaris and CDE Motif Libraries

In addition to widgets, CDE and OLIT support a number of features and convenience routines that are useful in manipulating widgets and the UI.

Only in OLIT

Drag and Drop Support

OLIT 3.0 provides an API, along with a `DropTarget` widget to make it easier to integrate Drag and Drop support for OpenWindows Version 3.0.

Dynamic Resources

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

Error-Handling Routines

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

Only in Solaris and CDE Motif

UIL Support

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

Clipboard Routines

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

Widget-Creation Routines

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

Compound String Support

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

Widgets

Both toolkits support a number of common widgets and gadgets with similar functionality and then each supports a number of more exclusive widgets. If a widget is implemented in one toolkit but not in the other, it is often possible to 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

High-Level Name	OLIT Class Name	Motif Class Name
BulletinBoard	BulletinBoard	XmBulletinBoard
Drawing Area	DrawArea	XmDrawingArea
Form Manager	Form	XmForm
Manager (Meta Class)	Manager	XmManager
Menu Button	MenuButton	XmCascadeButton
Message Popup	NoticeShell	XmMessageBox
Popup Dialog Shell	PopupWindowShell	XmDialogShell
Popup Menu Shell	PopupMenuShell	XmMenuShell
Primitive (Meta Class)	Primitive	XmPrimitive
PushButton	OblongButton	XmPushButton
RowColumnManager	ControlArea	XmRowColumn
ScrollBar	Scrollbar	XmScrollBar
Scrollable List	ScrollingList	XmList
Scrolled Window	ScrolledWindow	XmScrolledWindow
Sliding Scale	Slider	XmScale
StaticText/Label	StaticText	XmLabel
Text, Multi-line	TextEdit	XmText
TextField	TextField	XmTextField
Toggle Button	RectButton	XmToggleButton

Widgets Exclusive to Solaris and CDE Motif

The following is a list of the widgets that are exclusive to CDE Motif.

Note – These widgets are also in Solaris Motif, except for the Terminal Emulator, ComboBox, SpinBox, and MenuButton widgets.

The libDtWidget library, which contains the ComboBox, SpinBox, and MenuButton 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

LibDtTerm

This library provides a set of widgets to design or add a terminal emulator window to an application.

DtComboBox

The DtComboBox widget is a combination of a TextField and a List Widget that provides a list of valid choices for the TextField. Selecting an item from this list automatically fills the TextField with that list item.

DtSpinBox

The DtSpinBox widget is a convenient UI control that increments and decrements an arbitrary TextField.

DtMenuButton Widget

This is a command widget that complements the menu cascading functionality of an XmCascadeButton widget.

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.

Widgets Exclusive to OLIT

AbbrevMenuButton

This button is a primitive widget that displays exclusive and nonexclusive choice menus.

DropTarget

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

Flat Widgets

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 a significant memory savings is provided.

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 exposes its method hooks for an application to program custom widget behavior.