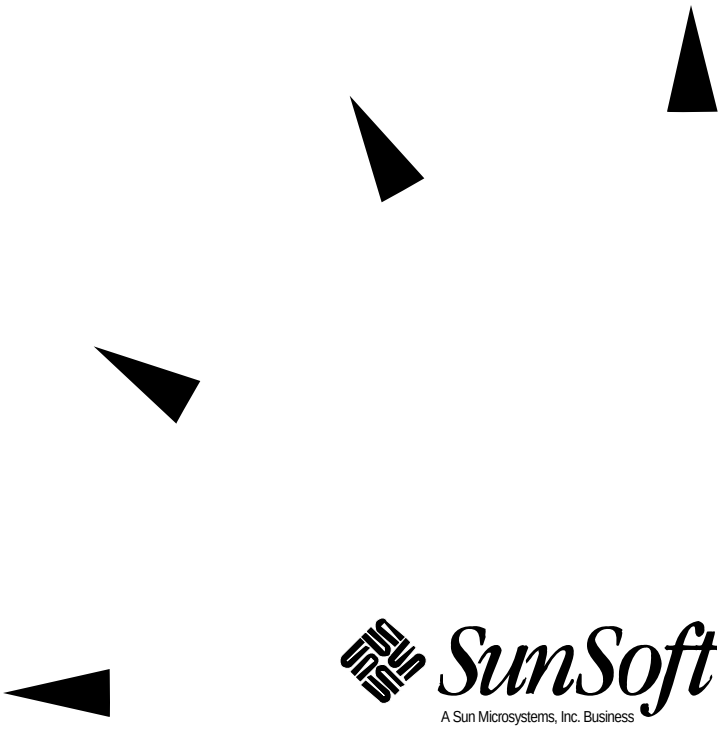
OPEN LOOK to Motif GUI Transition Guide



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

Contents

Preface v

1. The Motif Environment 1

Important Note to Developers 1

Using the Motif Window Manager with OpenWindows 2

Compiling and Linking 4

Install Motif from Solaris 2.4 4

Shared Library Policy 5

Future Compatibility 5

2. Porting Your Code to Motif 7

XView Libraries 8

Architecture 9

First Considerations When Porting 9

Potential Complications 9

Programming Model 10

Differences 12

X Resources and Command Line Options 12
OPEN LOOK versus Motif 1.2.3 13
Features Only in OPEN LOOK 13
Features Only in Motif 14
Notable Implementation Differences Between User Interfaces 14
OLIT versus Motif 1.2.3 Libraries 16
Only in OLIT 16
Only in Motif 17
Widgets 17
OLIT Exclusive Widgets 18
Interoperability Issues 19
Porting from IXI Motif 1.2.2 20
Tools to Ease the Transition 21
Application Architecture 22
A. Solaris 2.4 Packaging 25
B. References 27
Motif 1.2.3 Documentation 27
Related References 28
OPEN LOOK to Motif 1.2.3 Transition 28
Graphical User Interfaces 29
Motif Programming 29
OPEN LOOK Programming 30
Xt / XLib Programming 30

Preface

This version of Motif is based on OSF 1.2.3, and is compatible with Common Development Environment (CDE) Motif. SunSoft's previous Motif Release 1.2.2 was based on IXI Motif and has some features that were not OSF 1.2 Standard.

Solaris 2.4 provides a set of Motif libraries. This guide describes how to access these Motif 1.2.3 libraries, and compares the differences between the OPEN LOOK and Motif GUI.

Audience

As a developer, you may be asked to port existing applications that use the OPEN LOOK® graphical user interface (GUI) from XView™ and OLIT to Motif, or from IXI 1.2.2 to OSF 1.2.3. This overview gives a general description of the key architectural issues that you might encounter when planning this transition.

This overview assumes that you are proficient in OPEN LOOK application development using either XView or OLIT, X systems, and UNIX®, and assumes you are familiar with Motif, as well.

How This Book Is Organized

Chapter 1, "The Motif Environment," describes how to access the Solaris 2.4 Motif libraries, and set up the Motif environment.

Chapter 2, "Porting Your Code to Motif," explains the major feature differences between the OPEN LOOK and Motif GUI, covers toolkit dependencies such as XView to Motif 1.2.3, OLIT, and tools to make the transition easier.

Appendix A, "Solaris 2.4 Packaging," lists the contents of the Solaris 2.4 new packaging environments.

Appendix B, "References," lists additional documents that may help you make the transition from the OPEN LOOK to Motif GUI.

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`	What you type, contrasted with on-screen computer output	`system% su` `Password:`
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.

Table P-1 Typographic Conventions *(Continued)*

Typeface or Symbol	Meaning	Example
Code samples are included in boxes and may display the following:		
%	UNIX C shell prompt	`system%`
$	UNIX Bourne and Korn shell prompt	`system$`
#	Superuser prompt, all shells	`system#`

The Motif Environment 1

This chapter describes accessing the set of Motif 1.2.3 libraries and setting up a Motif environment. This chapter contains the following sections:

Important Note to Developers	*page 1*
Compiling and Linking	*page 4*
Shared Library Policy	*page 5*
Future Compatibility	*page 5*

Important Note to Developers

As part of each Solaris release, SunSoft ships a separate Software Developer Kit (SDK) product which contains binaries, header files, and documentation used by software developers. Some SDK components only ship header files in the SDK and not the runtime. The SDK is used in tandem with the Solaris runtime product and it is released in that way.

Motif support in Solaris 2.4 is packaged differently than it was in 2.3. This new packaging provides runtime support in the runtime product, and developer support in the Developer's environment. Appendix A, "Solaris 2.4 Packaging," describes the new packaging scheme for this release of Solaris, and outlines the contents of all the Motif packages.

The Motif runtime support includes the following:

- Dynamic libraries
- Header files
- Key bindings

The Motif developer support includes the following:

- Debug libraries
- Man pages
- Demos and sample source
- Motif window manager (`mwm`)

The Motif runtime support is not part of the default installation in the Solaris 2.4 runtime product. This package must be explicitly chosen during the Solaris installation or added later using the `pkgadd` command. Install the runtime support before using the developer support.

Note – Motif applications built on the Solaris 2.4 (S494) version of the Motif runtime package will not work when used with the Motif dynamic libraries delivered in Solaris 2.3 (S1093).

The Motif Window Manager provided with the Solaris 2.4 Developer's Environment is not supported by SunSoft as a runtime user product and must not be distributed to users. `mwm` is *only* included in the Developers environment for development purposes. SunSoft recommends using the default `olwm` window manager.

Using the Motif Window Manager with OpenWindows

This section describes how to set up your environment to use the Motif Window Manager and to access it from OpenWindows.

▼ Setting Up the Motif Environment

To set up the environment to support Motif development, set the following variables in the initialization file for your shell (for example `.profile` for the Bourne or Korn shell or `.login` for the C shell).

1. **Set the `MOTIFHOME` environment to `/usr/dt`.**
2. **Set the `OPENWINHOME` environment variable to `/usr/openwin`.**

3. **Set the** LD_LIBRARY_PATH **environment variable to include the values** $MOTIFHOME/lib:$OPENWINHOME/lib**.**
4. **Include the value**$MOTIFHOME/bin **in your** PATH **environment variable.**
5. **If** SUNWfwm **is installed, add** /opt/SUNWmfwm/bin **to your** PATH**.**
6. **Set the** UIDPATH **environment variable to** /opt/SUNWmfdm**.**
7. **Include the value** $MOTIFHOME/lib/%T/%N%S **in your** XFILESEARCHPATH **environment variable.**
8. **Set the** XMBINDDIR **environment variable to** $MOTIFHOME/lib/bindings**.**
9. **Both developers and users need to add** /usr/dt/man **to their** MANPATH**. If developers have installed the optional** SUNWmfwm**, they need to add** /opt/SUNWmfwm/man **to their** MANPATH**.**
10. **Source the initialization file by typing** . .profile **for the Bourne or Korn shell, or** source .cshrc **for the C shell.**
    Alternatively, you can log out and log in again so that the changes are recognized in your complete environment.

You can run Motif clients with the OPEN LOOK Window Manager (olwm) without making further changes.

▼ Setting Motif Window Manager as the Default

The openwin script indirectly starts the OPEN LOOK Window Manager (olwm). If you want the script to start the Motif Window Manager, you can edit the .xinitrc file in your home directory.

Note – mwm is only available if the Solaris 2.4 SUNWmfwm is installed.

---

1. **Edit the** .xinitrc **file in your home directory.**
   If you do not have an .xinitrc file in your home directory, copy the file $OPENWINHOME/lib/Xinitrc to your home directory and rename it .xinitrc.

2. **Replace** `olwm` **with** `mwm` **in the following line**:
   `olwm -syncpid $pid &`
3. **Save the changes and quit the** `.xinitrc` **file.**

▼ Enabling Quit on the Workspace Menu

The Motif Window Manager does not have an option to exit `mwm`. Use the following steps to enable Quit on the Workspace menu.

1. **Type** `cp /opt/SUNWmfwm/lib/system.mwmrc $HOME/.mwmrc` **and press Return.**
2. **Edit the** `$HOME/.mwmrc` **file and remove the leading exclamation mark (!) from the following line**:
   `!"Quit..." f.quit_mwm`

▼ Enabling Drag and Drop for the Motif Window Manager

To ensure the OpenWindows drag-and-drop option runs correctly with `mwm`, modify the `openwin-sys` script in `/usr/openwin/lib` (`$OPENWINHOME/lib`).

1. **Become superuser, and edit the** `$OPENWINHOME/lib/openwin-sys` **script. Remove the comment from the following line**:
   `#dsdm &`
2. **Save the changes and quit the editing file.**
3. **Exit the window system and restart it.**
   This change initiates the drop-site database manager and restarts the window system.

Compiling and Linking

Install Motif from Solaris 2.4

You install runtime package from Solaris 2.4 and Motif from the Software Developers Kit. The Motif header files required for application development are located under `/usr/dt/include`. The Motif libraries are located in `/usr/dt/lib`.

When you compile Motif programs, include the following compiler syntax to enable the compiler to find the Motif and X Window System header files:

```
-I$MOTIFHOME/include -I$OPENWINHOME/include
```

Use the following compiler syntax to direct the linker to the correct shared libraries as shown in the following:

```
-L$MOTIFHOME/lib -L$OPENWINHOME/lib
```

The following is an example of a compile and link-line for a Motif application that does not use uil:

```
cc -o myprog -I$MOTIFHOME/include -I$OPENWINHOME/include \
myprog.c -L$MOTIFHOME/lib -lXm -L$OPENWINHOME/lib  -lXt -lX11
```

By default, the SPARCworks C compiler dynamically links your application against the shared Motif, Intrinsics, and Xlib libraries. The shared library files must be accessible to the application at run time.

Shared Library Policy

SunSoft will increment the major version number of each shared Motif library whenever there are binary incompatible differences from the previous release. SunSoft will make available (either on the Motif distribution or through some other channel), all prior versions of each library. This will ensure that your applications linked with a particular release can continue to run, even after a new Motif release has been installed.

Future Compatibility

Developers who used Motif 1.2 to subclass widgets need to be aware of potential binary compatibility problems with future releases. Internal Motif widget data structures may be changed, breaking any subclass that relies on the position of fields in these data structures.

Motif provides a mechanism that a developer can use to avoid such problems. Refer to the *Motif Programmer's Reference* manual.

A demonstration of this mechanism is provided in `/opt/SUNWmfdm/src/dogs`.

Porting Your Code to Motif 2

This chapter outlines the toolkit terminology, common widgets, high-level widgets, and compares the OPEN LOOK and Motif GUI.

This chapter contains these sections:

XView Libraries	*page 8*
First Considerations When Porting	*page 9*
Programming Model	*page 10*
OPEN LOOK versus Motif 1.2.3	*page 13*
OLIT versus Motif 1.2.3 Libraries	*page 16*
OLIT Exclusive Widgets	*page 18*
Interoperability Issues	*page 19*
Porting from IXI Motif 1.2.2	*page 20*
Tools to Ease the Transition	*page 21*
Application Architecture	*page 22*

2

XView Libraries

Terminology

The XView toolkit, and the OLIT and Motif toolkits use some of the following terminology:

XView	OLIT and Motif
Package	Widget
Attributes	Resources

XView is based directly on Xlib, whereas Motif and OLIT are Intrinsic based toolkits that are based on the Xt Intrinsics layer. The Intrinsics layer 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, create, modify and destroy graphical objects are different, as shown in the following examples:

XView	OLIT/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 are Xt based toolkits that have very similar architectures. XView is not an Xt based toolkit and its architecture differs from the previously mentioned toolkits. 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), 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 Motif 1.2.3 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 Motif 1.2.3 library, it still, to some degree, has an OPEN LOOK appearance. A program in such a state does not adhere to either style completely.

First Considerations When Porting

Porting an OLIT application to Motif 1.2.3 is easier than porting an XView application, because OLIT and Motif 1.2.3 have similar APIs. However, a conversion from OLIT is not necessarily easy and can require restructuring the software.

Potential Complications

Porting software from OPEN LOOK to Motif 1.2.3 involves more than changing the names and arguments of function calls or pointing the linker to a new library. A conversion of this sort is more than a mere translation; moving to Motif entails some subtle changes in how the software operates.

For programs in which important functions are insulated from the surrounding GUI, the impact of this difference can be negligible. However, if the code is tightly linked to the user's actions or relies on a specific OPEN LOOK feature, it may be difficult to produce a Motif equivalent. In an extreme case, you must choose between violating the style guide or redesigning part of the program.

A simple example illustrates this point. Suppose, in a text-editor application, the user is allowed to edit several sections of a text buffer simultaneously. It is easy to implement this feature with an OPEN LOOK *split view*. However, split views are not part of the Motif specification. To produce the split view effect, you have to maintain extra buffers—or link several *Text* widgets to the same buffer using `XmTextGetSource()` and `XmTextSetSource()` calls.

To determine the amount of time involved in taking full advantage of Motif significantly depends on how the application is laid out. You can eliminate porting options by the application's design. Applications that are well designed are easier to port and easy to properly break down for maintenance, and readability.

Programming Model

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

- 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 Motif widget(s):

Table 2-1 Basic Mapping of Common Objects.

XView Package	**Equivalent 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	CascadeButton
Abbrev Menu Button	OptionMenu
Checkbox	RowColumn + ToggleButtons (*CheckBox*)
Exclusive Choice	RadioBox
Scrolling List	List (Scrolled List)
Message	Label
Slider	Scale
Text Field	TextField
Numeric Text Field	None
Text SubWindow	ScrolledWindow Text + Text (*ScrolledText*)
TTY SubWindow	None
Scrollbar	ScrollBar
Popup Menu	MenuShell + RowColumn (*Popup Menu*)
Pulldown Menu	MenuShell + RowColumn (*Pulldown Menu*)
Pullright Menu	MenuShell + RowColumn (*Pulldown Menu*)
File Chooser	DialogShell + FileSelectionBox
Split Control	Motif Paned Window

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 the XView packages with no Motif equivalents follows:

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

Additionally, *no* Motif equivalent exists for the OPEN LOOK function to split windows into different views. It should be noted that subclassing a Motif Manager widget to implement this function is not easy. You can use a combination of the Motif paned window and the Motif text widget to give similar, more primitive functions.

Some XView functions have Motif equivalent functions; however, the APIs are significantly different. A sample list of some of the functions follows:

XView API	Motif/Xt
Selection Service	Xt Selection API
Drop Target Package	Motif Drop and Drag API
Notifier	Xt Event Management API

X Resources and Command Line Options

The X resources that allow runtime customization of XView and Motif applications are different. XView objects may not have instance names attached to them. Motif/Xt do contain either class or instance names attached, for example, `mainframe.control_panel.button1.foreground`. To make Xt resources affect more than one object, use wildcards, for example, `mainframe*foreground`.

For more details on Motif/Xt resources, refer to some of the Xt books listed in Appendix B, “References.”

OPEN LOOK versus Motif 1.2.3

When you compare an OPEN LOOK application with its Motif counterpart, a few contrasting visual elements are immediately apparent. The OPEN LOOK buttons are round, whereas the 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.

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 Motif
- Aspects of Motif that do not appear in OPEN LOOK
- Features or other elements that appear in both specifications but are implemented differently

See Appendix B, "References," for a list of style guides and other references that describe the OPEN LOOK and Motif GUIs.

Features Only in OPEN LOOK

The following features are found in OPEN LOOK and are implemented in XView or OLIT (or both):

- Secondary text selections
- Scrollbar anchors
- Menu Defaults
  - Automatic default, short cut method for selecting a menu default
  - Default menu item indicated by a *ring*
- Font chooser widgets
- Pointer warping
- Minimized windows are valid drop areas
- tty subwindows
- Visual feedback on menu item selections
- Numeric text widgets

Features Only in Motif

The following features are found in only Motif:

- A fully-specified file selection mechanism
- Standard X clipboard cut and paste

Notable Implementation Differences Between User Interfaces

Many features are roughly equivalent in Motif and OPEN LOOK 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)

Other significant differences include the following:

- Keyboard bindings
- Window manager controls associated with each window
- Approach to implementing internationalization
- Interpretation of specific mouse actions

Tear-off Menus

In Motif, tear-off menus replace the *pinned menus* of OPEN LOOK. Select the dashed line on the top of the menu to tear off the menu.

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 title bar, a window menu is displayed. The options offered under the two GUIs introduce an important 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 1 shows the default left-to-right mapping of the three OPEN LOOK buttons.

Table 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 2, also start by default with the left mouse button.

Table 2 Motif Buttons

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

2

OLIT versus Motif 1.2.3 Libraries

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

Only in OLIT

The following features are available 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 V3.

Dynamic Resources

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

Error-Handling Routines

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

Input -Focus Routines

OLIT provides routines that you can use to manipulate and direct input focus around the UI.

Only in Motif

UIL Support

Motif supports using UIL definitions for the UI layout. This separation allows the UI to be modified without recompiling the executables of the program. 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. However, the changed UIL file must be recompiled using uil.

Clipboard Routines

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

Widget-Creation Routines

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

Compound String Support

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

Widgets

Both toolkits support a number of common widgets and gadgets with similar functionality. In addition, 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 2-1 matches the common widget name to the actual class name of the widget in each toolkit.

Table 2-1 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

OLIT Exclusive Widgets

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.

Interoperability Issues

The following section summarizes some of the interoperability issues that exist between OPEN LOOK and Motif applications. You can not do the following:

- Use primary copy, move, and link in XView and OLIT applications.
- Make secondary selections between Motif applications and XView or OLIT applications.
- Copy or cut from a Motif application into the Clipboard and then paste to an XView text subwindow.

- Copy or cut and paste Asian text between a Motif application and an XView or OLIT application.
- Drag and drop between Motif applications and XView or OLIT applications.

The following section summarizes some of the interoperability issues between the Motif Window Manager (`mwm`) and OPEN LOOK Applications.

- The window manager items for some XView and OLIT windows are not correct, for example, some applications may display extra titles, or extra resize handles. These differences do not affect the functions of the application.
- The window manager function `f.kill` when executed on base frame windows of an XView application (typically done by pulling down the default window menu and selecting *Quit* or *Close*) may exit the application without user confirmation, or any further application-specific processing.
- The resource `keyboardFocusPolicy` if set to `pointer` in XView or OLIT applications does not darken the caret. To darken the caret, click inside the window.
- The default colors for XView applications are different from Motif.
- The drag-and-drop feature of dropping data onto an icon of a closed XView application will not work while running `mwm`. Running other Motif applications will work.

Porting from IXI Motif 1.2.2

- Incompatibilities between Motif 1.2.2 and OSF/Motif 1.2.2

  Some features present in the previous release were a legacy of IXI Motif 1.2.2. These features are not part of the OSF/Motif 1.2 specification, and are not present in this version of Motif released by SunSoft. Remove them if you have used them.

The following features are no longer supported by SunSoft:

- `XmList` convenience functions:

  `XmListRecolorItem`

  `XmListRecolorPos`

  `XmListSetClientDataPos`

  `XmListSetClientDatasPos`

- XPM format bitmap file support

  IXI 1.2.2 supported conversion of XPM format files for some resources with bitmap semantics. This feature is not available in this release.

- `XmForm` widget

  The `XmForm` widget implementation in the 1.2.2 was an IXI implementation, and not the OSF original though the API was identical. You can make your applications work with other Motif 1.2 implementations by recompiling them. However, internal data structures and algorithms were different. Binary compatibility with other Motif implementations for applications using `XmForm` widgets is not assured. This SunSoft Motif release uses the OSF `XmForm` widget implementation.

- Directory hierarchy

  The directory hierarchy of the SunSoft Motif 1.2.3 software is different from 1.2.2 and of other Motif products. The distribution has a similar structure to the CDE hierarchy for consistency and to ease migration for future CDE users. Some other Motif 1.2 implementations install into `/usr/lib`, `/usr/include` and `/usr/bin/X11`. You will need to ensure that your environment and your `make` configuration files reflect use the `MOTIFHOME` scheme outlined in these release notes. See Chapter 1, “The Motif Environment,” for more information.

Tools to Ease the Transition

Many third-party tools are offered to transition to Motif.

Integrated Computer Solutions (ICS) is offering two converters: one translates XView code to GIL code and one that translates GIL code to Motif code.

SunSoft's Devguide now offers Motif Conversion Utilities as a way of helping customers make the transition. Devguide's front end is still in OPEN LOOK, and it has the same OPEN LOOK palette, but it is possible to use the Motif Conversion Utilities to transform GIL files into UIL files or Motif C code. The Motif Conversion Utilities (`guil` and `gmf`) in Solaris 2.4 SDK allow conversion of GIL files to UIL and Motif C code. UIL files can be imported into Motif GUI builders to generate Motif-based GUIs.

A final strategy is to perform the conversion at the GUI-builder level. One third-party vendor is currently working on a program for Solaris 1.x to produce GIL files from XView source.

Application Architecture

The ease in migrating your application from OPEN LOOK to Motif is a function of how your software is laid out and the tools you used to develop it.

The following list of questions provides the most significant factors for determining the ease of your migration to Motif:

- Does your application cleanly separate UI capabilities from *internals*?

  If you can draw a line through your code modules and completely isolate those that constitute the UI from those that make up the remainder of your application, then you can focus your migration efforts on the process of replacing the UI modules with equivalent ones developed for Motif. Many application developers follow software development methods that require this kind of clean separation, and, in some cases, even formally specify the program boundaries between UI and application internals.

  Alternatively, if your software is more monolithic and has application-specific abilities embedded within functions that also provide the UI, then you may have to spend extra time separating the two types of functions, thereby complicating your migration.

- Did you use any UI development tools (for example, *builders*) in the development of your application?

  Your use of an application builder gives you opportunities for saving effort in your migration to Motif. In most cases, the use of a builder implies that you have some degree of separation between UI functions and application internals, the advantages of which were previously discussed.

  Also, builders typically use generalized internal storage formats or are capable of generating interchange files, each of which may be post-processed to automate some of the conversion process. Contact your builder vendor to see what migration tools they are currently offering.

  Other less tangible *tools* that you might have used and that could ease your transition include development approaches that produced functional requirements documents or high-level designs. These representations may describe your application in terms less specific to OPEN LOOK and that are more amenable to being mapped to Motif than your source code.

- How do you follow the event-oriented functional model inherent in the Xt Intrinsics?

  The asynchronous style of programming typically used in applications using the Xt Intrinsics requires much of the application capabilities to be concentrated in the main program and *callback* functions registered to respond when particular events occur in the running application. If your application uses the standard facilities provided by Xt and follows its model closely, then you can save some time migrating by making a detailed study of your main program and callback functions. This will allow you to see how much of what these functions do to application internals.

  If you follow the OLIT model because OLIT is an Intrinsics-based toolkit, you will have an easier time migrating than applications built with XView.

- Do you share code with other applications or developers?

  Do not overlook the possibility that you may share some of the migration work with others. If you use libraries or tools from other vendors, you should contact them to see what plans they have for transitioning those software products to Motif.

2

Solaris 2.4 Packaging A

Solaris 2.4 contains Motif runtime and developer support as described in Table A-1. The first package is the runtime product and those that follow are the developers packages in the Solaris Developer's Kit.

Please refer to Chapter 1, "The Motif Environment," for information about the environments and installing these packages.

Table A-1 Solaris 2.4 Environments

Name	Default Location	Description
SUNWmfrun	/usr/dt	Contains the Motif dynamic libraries, their header files, and an executible, xmbind and its man page.
SUNWmfdev	/usr/dt	Contains the Motif dynamic debug libraries.
SUNWmfman	/usr/dt	Contains the Motif man pages.
SUNWmfwm	/opt/SUNWmfwm	Contains the Motif window manager and its man pages.
SUNWmfdm	/opt/SUNWmfdm	Contains the Motif demos.

A

References B

This appendix lists other documents that support Motif 1.2.3 in the following sections:

Motif 1.2.3 Documentation	*page 27*
Related References	*page 28*

Motif 1.2.3 Documentation

The following documents and books listed are not included on the CD. Check your local computer bookstores for their availability. In addition to this guide, the following documents support other Motif 1.2.3 documentation:

Note – The documentation describes Motif 1.2 interfaces. Motif 1.2.3 is a patch release of 1.2 and the same specifications apply.

- *OSF Application Environment Specification (AES) User Environment Volume, Revision C*, PTR Prentice Hall, 1993.

  Describes the Motif Application Environment Specification. Information on these specifications can also be found in the *Motif 1.2 Programmer's Reference Manual*.

- OSF / Motif User's Guide, Revision 1.2, PTR Prentice Hall, 1993.

  Describes how to interact with Motif-based applications, including how to use and customize the Motif Window Manager, how to use the Motif widgets, and how to customize your interaction with Motif applications.

- *OSF/Motif Programmer's Guide, Revision 1.2*, PTR Prentice Hall, 1993.

  Describes how to use the Motif API to create Motif applications. Also describes the architecture of the Motif widget set and the features of the toolkit. This guide is useful for new Motif programmers.

- *OSF/Motif Programmer's Reference, Revision 1.2*, PTR Prentice Hall, *1993.*

  Provides reference information for the Motif commands and functions in UNIX style man pages. The reference information covers the toolkit (library functions, widget documentation and resources), the window manager, user interface language commands, and the library functions.

- *OSF/Motif Style Guide*, PTR Prentice Hall, *1993.*

  A description of the framework of behavior specifications (the Motif feel) to guide developers in the design and implementation of products with Motif style user interface. The guide is split into sections applicable to four audiences (application designers, widget designers, user interface system designers, and window manager designers) The appendixes describe how the Motif widgets correspond to the components in the guide, and provide a checklist of requirements for OSF / Motif Application-level certification.

Related References

OPEN LOOK to Motif 1.2.3 Transition

Starks, Cindy. A Developer's Look at the COSE Desktop. *UNIX Review.* March 1994; 12(3) 41-45.

Bryant, David, J. Fowler, R. Levine. From OpenWindows to CDE. *UNIX Review.* March, 1994; 12(3) 47-51.

Moore, Michael. The CDE Libraries. *UNIX Review.* March 1994; 12(3) 53-57.

Keefe, Sarah. Migrating Your Sun Workstation To Motif. *The X Journal*, July-August 1993.

Graphical User Interfaces

OPEN LOOK to Motif GUI Transition Guide 801,6567-10, SunSoft, October, 1993.

OPEN LOOK Graphical User Interface: Programmer's Guide, UNIX System Laboratories, 1992.

OPEN LOOK Graphical User Interface: User's Guide, UNIX System Laboratories, 1992.

OPEN LOOK Graphical User Interface Application Style Guidelines, Sun Microsystems, Inc., Addison-Wesley Publishing Company, Inc., 1990.

OPEN LOOK Graphical User Interface Functional Specification, Sun Microsystems, Inc., Addison-Wesley Publishing Company, Inc., 1990.

OPEN LOOK Graphical User Interface: Programmer's Reference Manual, Prentice Hall, 1992.

OPEN LOOK Intrinsics Toolkit Widget Set Programmer's Guide, AT&T, 1990.

OPEN LOOK Intrinsics Toolkit Widget Set Reference Manual, AT&T, 1990.

OLIT & Motif: A Technical Comparison, Amy Moore, M. Goyal, SunSoft, September, 1992.

Motif Programming

Motif Programming in the X Window System Environment, William A. Parrette, McGraw-Hill, 1993.

Motif Programming: The Essentials—and More, Marshall Brain, Digital Press, 1992.

Motif Programming Manual, Dan Heller, O'Reilly & Associates, 1992.

Motif Reference Manual, Paula Ferguson, O'Reilly & Associates, 1992.

The X Window System Programming and Applications with Xt, OSF/Motif Edition, Douglas Young, Prentice Hall, 1990.

B

OPEN LOOK Programming

An OPEN LOOK at Unix: A Developer's Guide to X. John David Miller, M&T Books, 1990.

The X Window System Programming and Applications with Xt, OPEN LOOK Edition, Douglas Young and John A. Pew, Prentice Hall, 1992.

XView Programming Manual, Dan Heller, O'Reilly & Associates, Inc., 1991.

XView Reference Manual, Thomas Van Raalte (ed.), O'Reilly & Associates, Inc., 1991.

Xt/XLib Programming

Programmer's Supplement for Release 5 of the X Window System, Version 11, David Flanagan, O'Reilly & Associates, Inc., 1991.

X Toolkit Intrinsics Programming Manual, Adrian Nye and Tim O'Reilly, O'Reilly & Associates, Inc., 1990.

X Toolkit Intrinsics Reference Manual, O'Reilly & Associates, Inc., 1991.

Xlib: C Language X Interface, James Gettys, Robert W. Scheifler, Ron Newman, Silicon Press, 1989.

Xlib Programming Manual, Adrian Nye, O'Reilly & Associates, Inc., 1990.

Xlib Reference Manual, Adrian Nye (ed.), O'Reilly & Associates, Inc., 1990.