Porting XUI Applications to Motif

Order Number: AA–PGZFA–TE

August 1991

This guide describes how to convert XUI applications to OSF/Motif applications.

Revision/Update Information:	This is a new manual.
Operating System and Version:	ULTRIX Version 4.2 VMS Version 5.4
Software Version:	ULTRIX Worksystem Software Version 4.2 VMS DECwindows Motif Version 1.0

Digital Equipment Corporation
Maynard, Massachusetts

August 1991

The postpaid Reader's Comments forms at the end of this document request your critical evaluation to assist in preparing future documentation.

ZK5640

This document was prepared using DECdocument, Version 3.3-1b.

Contents

About This Guide ix

1 Introduction to Porting

1.1 Name Changes 1–1
1.2 Toolkit 1–2
1.3 Motif Window Manager 1–2
1.4 User Interface Language 1–3
1.5 User Interface Style 1–3
1.6 Interoperability 1–4

2 Converting Your XUI Files

2.1 Before Converting Your Files 2–1
2.2 Running the Filters 2–1
2.3 Interpreting Output from the Filters 2–3

3 After Converting Your Files

3.1 Reviewing Conversion Output 3–1
3.1.1 Creating a Listing of Differences 3–1
3.1.2 Resolving Messages from Porting Filters 3–2
3.1.3 Reviewing the File for Readability 3–3
3.1.4 Making Additional Name Changes 3–3
3.1.5 Reviewing High-Level Conversions 3–6
3.1.6 Checking Return Values 3–7
3.1.7 Checking Compound String Usage 3–8
3.1.8 Checking UIL Files 3–9
3.2 Reviewing Motif Toolkit Components and Making Changes 3–9
3.2.1 Include Header Files 3–9

3.2.2 Motif Widgets 3–10
3.2.2.1 Bulletin Board Widget 3–12
3.2.2.2 Command Widget 3–13
3.2.2.3 Drawing Area Widget 3–13
3.2.2.4 File Selection Box Widget 3–13
3.2.2.5 List Widget 3–14
3.2.2.6 Main Window Widget 3–14
3.2.2.7 Message Box Widget 3–14
3.2.2.8 Push Button Widget 3–15
3.2.2.9 Row Column Widget 3–15
3.2.2.10 Scale Widget 3–16
3.2.2.11 Scroll Bar Widget 3–16
3.2.2.12 Scrolled Window Widget 3–16
3.2.2.13 Separator Widget 3–17
3.2.2.14 Text Widget 3–17
3.2.3 Digital Extended Motif Widgets 3–17
3.2.3.1 Color Mixing Widget 3–18
3.2.3.2 Compound String Widget 3–18
3.2.3.3 Help Widget 3–18
3.2.3.4 Print Widget 3–19
3.2.3.5 Structured Visual Navigation Widget 3–19
3.2.4 Custom Widgets (Widget Programmers Only) 3–20
3.2.4.1 Widget Hierarchy 3–20
3.2.4.2 XmPrimitive Class Record 3–20
3.2.4.3 Bit Gravity 3–21
3.2.4.4 Keyboard Focus 3–21
3.2.5 Callback Records 3–22
3.2.6 Callback Structures 3–23
3.2.6.1 Command Widget 3–24
3.2.6.2 Drawing Area Widget 3–24
3.2.6.3 File Selection Box Widget 3–25
3.2.6.4 List Widget 3–26
3.2.6.5 Row Column Widget 3–26
3.2.6.6 Selection Box Widget 3–27
3.2.6.7 Toggle Button Widget 3–27
3.2.7 Compound String Usage 3–28
3.2.8 Resolution Independence 3–29
3.2.9 Scroll Bars 3–29
3.2.10 Widget Layout 3–30
3.3 Reviewing Intrinsics Features and Making Changes 3–30
3.3.1 Specifying Application Context 3–31

3.3.2 Reviewing X11 Release 4 Changes 3–31
3.3.2.1 Routine Interfaces 3–31
3.3.2.2 Widget Semantics 3–32
3.3.3 Before Compiling Your Application 3–33
3.4 Compiling and Linking the Application 3–34
3.5 Testing the Application 3–36
3.6 Making Changes Required by the Motif Window Manager 3–37
3.6.1 Check MWM-Client Interaction 3–37
3.6.2 Checking Keyboard Focus 3–38
3.6.3 Checking Key and Mouse Bindings 3–38
3.7 Make Changes Required by the *OSF/Motif Style Guide* 3–38
3.7.1 Adding Menu Mnemonics 3–39
3.7.2 Adding Keyboard Accelerators 3–39
3.7.3 Adding Keyboard Traversal 3–40
3.7.4 Checking Context-Sensitive Pop-up Menus 3–41

4 Giving Information to Application Users

4.1 Updating the Documentation 4–1

A Summary of XUI and OSF/Motif Differences

A.1 Component Names A–1
A.1.1 Widget Classes A–2
A.1.2 Function Names A–3
A.1.3 Resource Names A–5
A.1.4 Enumeration Literal Names A–8
A.1.5 Callback Reason Names A–10
A.1.6 Compound Strings A–11
A.1.7 Fontlist Names A–12
A.1.8 Clipboard Names A–12
A.1.9 Resource Manager Names A–13
A.2 Terminology A–14
A.3 Windows and Window Managers A–15
A.4 Menus and Menu Items A–16
A.4.1 Menu Bar and Standard Menus A–17
A.4.2 File Menu Items A–18
A.4.3 Edit Menu Items A–20
A.4.4 Help Menu Items A–21
A.5 Standard Message Boxes A–22
A.6 Mouse Buttons Behavior A–22

B Porting Filter Summary

C Intrinsics Routine Summary

Index

Tables

3–1	Porting Filter Messages	3–2
3–2	Additional Name Changes Required for Conversion	3–4
3–3	Compound String Resources Not to Be Freed	3–28
3–4	VMS Shareable Images Required for Linking	3–34
3–5	ULTRIX Include Files (RISC Systems)	3–35
3–6	ULTRIX Libraries Required for Linking (RISC Systems)	3–35
3–7	ULTRIX Include Files (VAX Systems)	3–36
3–8	ULTRIX Libraries Required for Linking (VAX Systems)	3–36
A–1	Widget Class Name Changes	A–2
A–2	Function Name Changes	A–3
A–3	Resource Name Changes	A–5
A–4	Enumeration Literal Name Changes	A–8
A–5	Callback Reason Names	A–10
A–6	Compound String Names	A–11
A–7	Fontlist Names	A–12
A–8	Clipboard Names	A–12
A–9	Resource Manager Names	A–13
A–10	Terminology Differences Between XUI and Motif	A–14
A–11	Differences Between XUI and OSF/Motif Windows and Window Managers	A–15
A–12	Motif Window Menu Items and Functions	A–16
A–13	Differences Between the Motif and XUI Menus in the Menu Bar	A–17
A–14	Differences Between File Menu Items	A–18
A–15	Differences Between Edit Menu Items	A–20
A–16	Differences Between Help Menu Items	A–21
A–17	Differences in Mouse Buttons	A–22
B–1	Porting Filters: Names and Functions (VMS)	B–1
B–2	Porting Filters: Names and Functions (ULTRIX)	B–2

C–1 Default and Explicit Application Context Routines C–1

About This Guide

This guide is intended for application and widget programmers who want to port their XUI applications and custom widgets to OSF/Motif Version 1.1. Programmers should have experience writing applications for the X Window System; this manual is not a tutorial on writing Motif applications.

Related Documents

You will find the following documents useful during the porting process:

- *OSF/Motif Style Guide*
- *OSF/Motif Programmer's Guide*
- *OSF/Motif Programmer's Reference*
- *DECwindows Companion to the OSF/Motif Style Guide*
- *DECwindows Motif Guide to Application Programming*
- *DECwindows Extensions to Motif*
- *X Window System, Second Edition*
- *X Window System Toolkit: The Complete Programmer's Guide and Specification*

Conventions

This manual uses the following conventions:

italics	In syntax and function descriptions, italic type indicates terms that are variable.
Return	Unless otherwise specified, every command line is terminated by pressing the Return key.
. . .	In examples, a horizontal ellipsis indicates one of the following possibilities: • Additional optional arguments in a statement have been omitted. • The preceding item or items can be repeated one or more times. • Additional parameters, values, or other information can be entered.
. . .	A vertical ellipsis indicates the omission of items from a code example or command format; the items are omitted because they are not important to the topic being discussed.

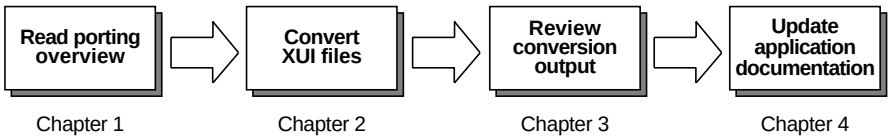
Abbreviations and Acronyms

The following abbreviations and acronyms are used throughout this manual:

MWM	Motif Window Manager
UIL	User Interface Language
UWS	ULTRIX Worksystem Software
XUI	X User Interface

1
Introduction to Porting

OSF/Motif is an industry-standard, cross-platform graphical user interface that features three-dimensional visuals and consistent operation and style. Porting your XUI application to Motif is a four-step process, as shown in the following figure:



Follow these steps, in order, to ensure a complete and efficient port of your application.

This chapter is the porting overview, and describes the following Motif features and how these features affect your application:

- Name changes
- Toolkit
- Motif window manager (MWM)
- User Interface Language (UIL)
- User interface style
- Interoperability

1.1 Name Changes

Motif provides new, consistent names for the following:

- Widget classes
- Create functions
- Resource manager (Mrm) functions
- Resources

- Enumeration literals
- Callback reasons
- String functions
- Fontlist functions
- Clipboard functions

You must change each instance in your application. Digital provides a set of porting filters to make this process easier, automatically changing the XUI names to Motif names. Chapter 2 describes how to convert your files using the porting filters. Appendix A contains a summary of the name changes and of other differences between XUI and Motif.

1.2 Toolkit

The Motif Toolkit features Motif widgets and Digital Extended Motif widgets, as well as changes to the following:

- Include files
- Custom widget creation
- Callbacks
- Compound string memory use
- Resolution independence
- Scroll bars
- Widget layout

Chapter 3 describes the changes you must make to your application in order to comply with the Motif Toolkit changes.

1.3 Motif Window Manager

Motif features a new window manager, with changes in the following areas:

- Window manager-client interaction
- Keyboard focus
- Key bindings and mouse bindings

Chapter 3 describes the changes you must make to your application in order to comply with the Motif window manager (MWM) changes. Section A.3 contains a summary of the differences between XUI and Motif window managers.

1.4 User Interface Language

The Motif User Interface Language (UIL) features changes for each object in the Motif Toolkit. These changes include new names, consistent with the Toolkit, for the following:

- Controls (widgets and gadgets)
- Arguments (widget attributes)
- Reasons (callbacks)

Digital provides a set of porting filters to make this process easier, automatically changing the XUI names to Motif names. Chapter 2 describes how to convert your files using the porting filters.

1.5 User Interface Style

The *OSF/Motif Style Guide* specifies the user interface appearance and behavior of Motif applications. This includes the following:

- Client area design
- Menus
- Controls
- Dialog boxes
- Keyboard traversal

Chapter 3 describes the changes you must make to your application in order to comply with the *OSF/Motif Style Guide*.

See the *OSF/Motif Style Guide* and *DECwindows Companion to the OSF/Motif Style Guide* for guidelines on designing and implementing a Motif user interface.

1.6 Interoperability

Motif applications are intended to run with the Motif window manager. These same applications can also run with the older XUI window manager. However, Digital recommends that you run the applications with the MWM in order to take full advantage of the function and appearance of Motif.

If your application users run Motif applications with the XUI window manager, modal widgets will no longer be modal. In fact, application shell windows might be unmapped from the window manager, resulting in transparent windows.

2
Converting Your XUI Files

Digital provides a set of porting filters to assist you in converting your XUI files written in C or UIL to Motif files. The filters replace occurrences of XUI names in the input source file with the Motif names. The XUI names can be in the actual code, within quotation marks, or in comments; the filters make the replacement without checking context.

There are nine filters in all. Appendix B lists the individual filters and the conversion each one performs. This chapter tells you how to run the filters and what to expect.

2.1 Before Converting Your Files

Before you convert your XUI files, set up two directories: one for XUI and one for Motif. This will make it easier to keep track of the original files and aid in verifying the changes after the conversion. Also, if you are going to support both Motif and XUI on your system, separate directories reduce user and programmer confusion about which files to use.

Also decide how you will create the user interface. If your application uses toolkit calls to create the user interface and you are planning to convert to User Interface Language (UIL), convert to UIL before converting your source files. The main reason is the porting filters themselves: they convert UIL files more efficiently. Converting to UIL could reduce the time required to port your application, especially if your application is written in a language other than C.

2.2 Running the Filters

The porting filters are run either from a command procedure (VMS) or a script file (ULTRIX). You can specify a single file or a directory of files as input to the filters. Read the section that applies to your operating system.

VMS

VMS Systems

On VMS systems, the porting filter command procedure is named DECW$DXM_PORT.COM and is in the DECW$EXAMPLES directory. The command line to run the procedure has the following format:

@DECW$EXAMPLES:DECW$DXM_PORT *source-name output-name* [LOG]

source-name	A file name or a directory name. If you do not specify a *source-name*, the command procedure prompts you for a name. If you specify a directory name, all *.C, *.H, and *.UIL files in the directory are converted.
output-name	A file name or directory name. If you specified a file name for *source-name*, specify a file name. If you specified a directory name for *source-name*, specify a directory name. If you do not specify an *output-name*, the command procedure prompts you for a name. If you specify a directory name and the output directory does not exist, the command procedure automatically creates one.
LOG	An optional parameter. This displays messages on your display screen as the filters are running.

The command procedure places multiple versions of the output files in the output directory, one version for each filter. Save the source version and the latest output version; delete all other versions to free up disk space.

Instead of running the filters from the command procedure, you can also run a filter individually. However, to get the most benefit from the filters, run the filters from the command procedure.

Examples

The following example converts all *.C, *.H, and *.UIL files in the directory [JONES.XUI_FILES] and writes the converted files in the directory [JONES.MOTIF_FILES].

```
$ @DECW$EXAMPLES:DECW$DXM_PORT [JONES.XUI_FILES] - Return
_$ [JONES.MOTIF_FILES]
```

The following example converts the file MODULE.C, writes the converted file NEW_MODULE.C in the same directory, and displays messages on the screen as the filters are running.

```
$ @DECW$EXAMPLES:DECW$DXM_PORT MODULE.C NEW_MODULE.C LOG
```

ULTRIX Systems

ULTRIX

On ULTRIX systems, the porting filter script is named DXm_port and is in the /usr/lib/DXM/tools/filter directory (RISC systems) and /usr/lib/DXV/tools/filter directory (VAX systems). For both systems, the command line to run the script file has the following format:

DXm_port source-name

source-name A directory name. All *.c, *.h, and *.uil files will be converted.

The filters place the converted files in the source directory with the suffix _motif appended. For example, if a source file was named module.c, the converted file is named module.c_motif.

Example

The following example converts all *.c, *.h, and *.uil files in the directory /usr/users/jones/xui and writes the converted files in the same directory with the _motif suffix. The system is a RISC system.

```
system> /usr/lib/DXM/tools/filter/DXm_port /usr/users/jones/xui
```

2.3 Interpreting Output from the Filters

In most cases, the filters translate XUI names to Motif names. However, if the filters find an XUI name they cannot translate or an XUI name with multiple possible translations, the filters write a message in the output file identifying the problem. Review these messages and any suggestions in the output file and decide whether they are appropriate for your application.

For example, one filter translates high level XUI subroutine calls to Motif low-level creation calls, writing new low-level code directly into the input source file. In this case, you would look over the new code, check the description and syntax against the *OSF/Motif Programmer's Reference*, and decide whether it is appropriate for your application.

See Chapter 3 for information on reviewing output from the porting filters.

3
After Converting Your Files

After you have converted your files, do the following steps to complete the conversion:

1. Review conversion output and make additional changes.
2. Review Motif Toolkit components and make changes, as needed.
3. Review Intrinsics changes and make changes, as needed.
4. Compile and link your application.
5. Test the application.
6. Make changes required by the Motif Window Manager.
7. Make changes required by the *OSF/Motif Style Guide*.

3.1 Reviewing Conversion Output

To review the conversion output, first create a listing of differences between the source files and the output from the porting filters. Then, check specific areas in your program. The following sections describe how to generate a listing of differences and which areas of your program to check.

3.1.1 Creating a Listing of Differences

To create a listing of differences between a source file and the output from the filters, type one of the following commands:

VMS

VMS Systems

```
$ DIFFERENCE/OUTPUT=listing-file input-file output-file
```

ULTRIX

ULTRIX Systems

```
system> diff old-file new-file > listing-file
```

Print out the listing and review all differences to see if you agree with them. Refer to the listing as you complete this section.

3.1.2 Resolving Messages from Porting Filters

Edit your files and search for three consecutive right angle brackets (>>>); all filter messages begin with these characters. The messages indicate those sections of code that you need to evaluate. Table 3–1 lists the messages and their meaning.

Table 3–1 Porting Filter Messages

Message	Meaning
>>>Developer decision:	The filters changed the function for this section of code. Review the changes, read the Motif documentation, and determine whether the changes are appropriate. For example, you might have to choose a dialog box mode or specify additional parameters for a subroutine (the filters suggest values for new parameters).
>>>Mapping unknown:	The filters could not convert the XUI name to a Motif name; no Motif name exists for the XUI name. Review this section of your application code and the Motif documentation. Because the filter cannot complete the conversion, you must choose a Motif translation that meets your application's requirements.
>>>Mapping unsupported for this symbol:	The filters could not convert the XUI to a Motif name; the XUI name is unknown or not documented. Review this section of your application code and the Motif documentation. Choose a Motif translation that meets your application's requirements.
>>>Motif Transition:	The filters have made significant changes in the routine calls; the Motif routines should work the same as the XUI routines. Review this section of your application code and verify that it is appropriate for your application. In some cases, the message suggests alternative routines for you to choose from. Be sure to verify the routine's parameters with the Motif documentation for correct usage.

If a "Motif Transition" message appears in the output file, the original section of code is left intact, set off from the rest of the code by conditional statements. The filter message and suggested new code follow the original section of code. For example:

```
#ifdef ORIGINAL_CODE
  * DwtSetArg("TRUE",arglist,0,XtNallowShellResize)
#endif

>>Motif Transition : Routine conversion
{
   XtSetArg(arglist[0],XtNallowShellResize,"TRUE");
}
```

3.1.3 Reviewing the File for Readability

To improve the readability of your file, check the following parts of your program:

- **Comments** — The filters might have changed words embedded in the comments, particularly XUI widget names, argument names, and callback names in UIL files. Remove the changes and make sure the comments describe the code clearly.

  In addition, if a section of code was translated by the filters and the section included comments, the filters might have used the comment words as values for the resources in argument lists. Check the argument list resource values carefully.

- **Formatting** — The filters might have affected the way data declarations and other code are aligned. Adjust the code alignment.

3.1.4 Making Additional Name Changes

Review all name changes made by the porting filters. Although the filters make most of the necessary name changes, they do not make all necessary changes. Therefore, search for the characters or names in Table 3–2 and convert them to the appropriate Motif name. Section A.1 contains a summary of all name changes. Check the *OSF/Motif Programmer's Reference* to verify all Motif names.

Table 3–2 Additional Name Changes Required for Conversion

Search File for:	Replace with:	Comments:
DECwDwtApplProg.h	Xm.h	Replace with the appropriate include files (see Section 3.2.1).
DECwDwtWidgetProg.h	XmP.h	Replace with the appropriate include files (see Section 3.2.1).
decw	[1]	Replace with the appropriate Motif name.
decw$include	Xm, X11, Mrm, or DXm	Used within #include statements; the correct choice depends on the .h file. Replace to ensure application portability.
DRM	Mrm	For consistency with Motif naming conventions.
dwt	[1]	Replace with the appropriate Motif name.
DwtCallbackStructPtr	XtCallbackList	Replace when referencing callback lists (see Section 3.2.5).
DwtCallbackStruct	XtCallbackRec	Replace when referencing callback lists (see Section 3.2.5).
Dwt*Index	Xm*Index	Widget class indices used with XmPartOffset and XmField macros (* is a widget class name). For example, change DwtCompositeIndex to XmCompositeIndex.
DwtOffset	XmOffset	Replace the casting for the pointer to the class instance offsets.
DwtOffsetPtr	XmOffsetPtr	Replace the casting for the pointer to the class instance offsets.
external	externalref	Replace the label for external variables.
XmCreateLabel	[1]	If you do not want the label widget to resize itself each time to hold a new string, add the XmNrecomputeSize resource to the argument list and set it to FALSE.
XmStringCreateLtoR	XmStringCreateSimple	Replace if the original call was DwtLatin1String.

[1] String is not replaced. See comments for action.

(continued on next page)

Table 3–2 (Cont.) Additional Name Changes Required for Conversion

Search File for:	Replace with:	Comments:
XmNlabelString	XmNmessageString	Replace if part of argument list to XmErrorDialog, XmInformationDialog, XmMessageDialog, XmQuestionDialog, XmWarningDialog, or XmWorkingDialog.
XmNmaximum	XmNpaneMaximum	Replace if part of argument list to XmPanedWindow.
XmNminimum	XmNpaneMinimum	Replace if part of argument list to XmPanedWindow.
XmNtopPosition	XmNtopCharacter	Replace if part of argument list to XmText.
XmNvalue	XmNset	Replace if part of argument list to XmCreateToggleButton.
XtFree	XmStringFree	Replace if you are freeing XmString memory.
XtRemoveAllCallbacks	XtRemoveCallback or XtRemoveCallbacks	Replace to remove individual callbacks instead of all callbacks.
XtSetKeyboardFocus	XmProcessTraversal	For compatibility with Motif keyboard traversal operation.
.ecallback	[1]	Remove the string because you do not indirectly reference callback lists in Motif. See the following example.

[1] String is not replaced. See comments for action.

The following example shows the changes you must make to correctly reference a callback list.

Code output from filters

```
  if (ar_list.ecallback != NULL) ❶

where:

  XtCallbackList ar_list;        /*   callback list */
```

Code after additional changes

```
  if (ar_list != NULL)  ❶

where:

  XtCallbackList ar_list;        /*   callback list */
```

❶ Delete the .ecallback string.

3.1.5 Reviewing High-Level Conversions

One of the porting filters translates high-level XUI subroutine calls to Motif low-level creation calls and writes new code in the output file. The changes are indicated with the following message:

```
>>>Motif Transition : High-level widget routine conversion
```

When you see this message in your file, do the following:

- Check the *OSF/Motif Programmer's Reference* and make sure the arguments are valid.
- Check the defaults for the arguments and make sure they are still valid. In some cases, the defaults for parameters in Motif calls are different from the corresponding XUI calls. You might have to include additional arguments in your argument list if the defaults have changed in Motif. For example:

Code output from filters

```
        XtManageChild (
#ifdef ORIGINAL_CODE
 * DwtSeparator (menubar, "", 0, 0, 0) ❶
#endif

>>>Motif Transition : High-level widget routine conversion
    Source does not have correct number of parameters
);
```

Code after additional changes

```
ac = 0;
XtSetArg(al[ac], XmNorientation, XmHORIZONTAL); ac++;
XtManageChild (XmCreateSeparator (menubar, "", al, ac)); ❶
```

❶ Complete the conversion from the XUI name to the Motif name and supply the correct number of parameters.

- Check the ordering of the parameters on all calls. For example:

Code output from filters

```
#ifdef ORIGINAL_CODE
 * DwtMainSetAreas (main_widget, menubar, work_area, NULL, NULL, NULL)
#endif

>>>Motif Transition : High-level widget routine conversion
XmMainWindowSetAreas(,main_widget,menubar, NULL, work_area, NULL, NULL ❶
};
```

Code after additional changes

```
XmMainWindowSetAreas (main_widget, menubar, NULL, NULL, NULL, work_area); ❶
```

❶ Reorder and format parameters.

3.1.6 Checking Return Values

In converting from XUI to Motif, the Motif routines might no longer return the same number and types of return values as before. Check the return values of the Motif routine in the *OSF/Motif Programmer's Reference* and your application program's logic in processing them. For example:

Code input to filters

```
DwtCompString        cs;
DwtCompStringContext context;
int              status;

status = DwtInitGetSegment (&context, cs);

if (status == DwtFail)     /* String specified is not a compound string */
   {
   return k_badstring;
   }
else if (status == DwtEndCS) /* String passed is NULL */
   {
   *count = 0;
   return k_success;
   }
```

Code output from filters

```
XmString        cs;
XmStringContext context;
int             status;

status =
>>>Developer decision: XmStringInitContext
 (&context, cs);

if (status == FALSE)     /* String specified is not a compound string */ ❶
   {
   return k_badstring;
   }
else if (status == FALSE) /* String passed is NULL */ ❶
   {
   *count = 0;
   return k_success;
   }
```

❶ In this case, XmStringInitContext returns either TRUE or FALSE. You would change your program's logic and code to accommodate this.

3.1.7 Checking Compound String Usage

Although the porting filters change the XUI compound string routine names to Motif names, check the following for all XmString* routines:

- **Arguments** — The order of arguments might be changed or arguments might be missing.
- **Character set** — Check the *OSF/Motif Programmer's Reference* for a list of character sets.

- **Creation** — Use XmStringCreateSimple to create a compound string using the language environment of the widget.
- **Memory resources** — After you finish with a compound string, use XmStringFree to free memory resources.

3.1.8 Checking UIL Files

For UIL files, check the following:

- **Widget argument names** — Replace the argument name *x* with XmNx. See the *OSF/Motif Programmer's Guide* for additional information on UIL.
- **Comments** — Because the porting filters search and convert more common words (for example, help, map, title, width, y, and window), check the comments for readability.

3.2 Reviewing Motif Toolkit Components and Making Changes

The following sections describe the Motif Toolkit components that affect your application and the changes you must make to your application in order to be compatible with the Toolkit. See the *OSF/Motif Programmer's Guide* for a complete description of the Motif Toolkit.

3.2.1 Include Header Files

Motif requires a separate include file for each widget class used. You must add the appropriate include files to your program. Be sure to order your include files from the general to the more specific.

To ensure you have the required include files in your program, do the following:

- Search your program for all #include statements. Check all include files in your program and make sure they are Motif include files and not XUI include files. Change them as necessary. If you are writing an application, you need at least the following line in your program:

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

If you are writing a widget, you need at least the following line in your program:

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

- Search your program for all Xm calls, check against the *OSF/Motif Programmer's Reference* and determine whether you need to add any include files to your program. Add the appropriate include files.

  For example, if your program makes calls to XmCreateDrawingArea, find the entry in the *OSF/Motif Programmer's Reference* and look under the "Synopsis" heading. Add the following line to your program:

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

- Search for all Mrm calls. If you find any, add the following line to your program:

  ```
  #include <Mrm/MrmPublic.h>
  ```

- If you are going to use the Help widget, add the following line to your program:

  ```
  #include <DXm/DXmHelpB.h>
  ```

3.2.2 Motif Widgets

In addition to the name changes, Motif has also changed the resources and function for the following widgets:

- Bulletin board
- Command
- Drawing area
- File selection box
- List
- Main window
- Message box
- Push button
- Row column
- Scale
- Scroll bar

- Scrolled window

- Separator
- Text

3.2.2.1 Bulletin Board Widget

The bulletin board widget (XmBulletinBoard) is equivalent to the XUI dialog box widget (DwtDialogBox). If you are using the bulletin board widget, check your application for the use of the following resources. You might have to make some changes in order to obtain the desired appearance and operation of your widget.

- XmNautoUnmanage — Applies to both modal and modeless dialog boxes. The default value is True. See the *OSF/Motif Programmer's Reference* to determine whether this default is appropriate for your application.
- XmNbuttonFontList, XmNlabelFontList, and XmNtextFontList — Replace DwtNfont. See the *OSF/Motif Programmer's Reference* for a description of the resources.
- XmNdialogStyle — For modal dialog boxes, this resource can have one of the following values:

  XmDIALOG_FULL_APPLICATION_MODAL
  XmDIALOG_APPLICATION_MODAL
  XmDIALOG_SYSTEM_MODAL

- XmNunitType — This is a resource for XmPrimitive, XmManager, and XmGadget. The unit type can be set explicitly for the widget. If it is not set, it is inherited from the unit type of the parent widget. Be sure to check the default unit type of the parent. To change a default value, use XmSetFontUnit.

Focus callbacks will occur at different times.

The following XUI resources are not supported:

- DwtNautoUnrealize (use DXmNautoUnrealize; see *DECwindows Extensions to Motif*)
- DwtNfontX
- DwtNfontY
- DwtNgrabKeySyms

- DwtNgrabMergeTranslations
- DwtNtakeFocus

3.2.2.2 Command Widget

The command widget (XmCommand) is equivalent to the XUI command window widget (DwtCommandWindow). If you are using the command widget, check your application for the use of the following resources. You will have to make some changes in order to obtain the desired appearance and operation of your widget.

- XmNcommand — Resource value type is compound string.
- XmNhistoryItems — Resource value type is compound string.
- DwtNtTranslation — Not supported; use XmNtextTranslations instead.

3.2.2.3 Drawing Area Widget

The drawing area widget (XmDrawingArea) is equivalent to the XUI window widget (DwtWindow). If you are using the drawing area widget, check your application for the use of XmNexposeCallback and XmNresizeCallback resources. The bit gravity of the drawing area widget is NorthWestGravity. When the drawing area is resized, your application might not get an expose callback, particularly if there is no exposed area. Therefore, if your application needs to know when a resize occurs, use the XmNresizeCallback resource.

3.2.2.4 File Selection Box Widget

The file selection box widget (XmFileSelectionBox) is equivalent to the XUI file selection widget (DwtFileSelection). If you are using the file selection box widget, the following XUI resources are not supported:

- DwtNfileToExternProc
- DwtNfileToInternProc
- DwtNmaskToExternProc
- DwtNmaskToInternProc

Check your application for the use of these resources and make changes in order to obtain the desired appearance and operation of your widget.

3.2.2.5 List Widget

The list widget (XmList) is equivalent to the XUI list box widget (DwtListBox). If you are using the list widget, check your application for the use of the XmNselectionPolicy resource. Motif defines additional selection callbacks:

- XmNbrowseSelectionCallback (default)
- XmNextendedSelectionCallback
- XmNmultipleSelectionCallback
- XmNsingleSelectionCallback

Decide which Motif selection policy is appropriate for your list and make any changes to obtain the desired appearance and operation of your widget.

The XUI resource DwtNextendConfirmCallback is not supported.

3.2.2.6 Main Window Widget

The main window widget (XmMainWindow) is equivalent to the XUI main window widget (DwtMainWindow). If you are using the main window widget, check your application for the use of the XmMainWindowSetAreas function. The parameters are in a different order from those used with XUI. See the *OSF/Motif Programmer's Reference* for the correct order and make any changes.

The following XUI resources are not supported:

- DwtNacceptFocus
- DwtNfocusCallback

Check your application for the use of these resources and make changes in order to obtain the desired appearance and operation of your widget.

3.2.2.7 Message Box Widget

The message box widget (XmMessageBox) is equivalent to the XUI message box widget (DwtMessageBox). If you are using the message box widget, use the following convenience functions to create a message box:

- XmCreateErrorDialog — Informs a user of an invalid action.
- XmCreateInformationDialog — Provides status information to a user.

- XmCreateMessageDialog — Interactively queries and responds to a user.
- XmCreateQuestionDialog — Obtains information from a user.
- XmCreateWarningDialog — Informs a user of the results of an action and enables the user to act on the results.
- XmCreateWorkingDialog — Provides work-in-progress information to a user (for long operations).

The following XUI resources are not supported:

- DwtNnoCallback
- DwtNnoLabel
- DwtNsecondLabel
- DwtNsecondLabelAlignment

Check your application for the use of these resources and make changes in order to obtain the desired appearance and operation of your widget.

3.2.2.8 Push Button Widget

The push button widget (XmPushButton) is equivalent to the XUI push button widget (DwtPushButton). The following XUI resources are not supported:

- DwtNbordHighlight
- DwtNfillHighlight

Check your application for the use of these resources and make changes in order to obtain the desired appearance and operation of your widget.

3.2.2.9 Row Column Widget

The row column widget (XmRowcolumn) is equivalent to the XUI menu widget (DwtMenu). If you are using the row column widget, Motif requires the children of a menu bar to be homogeneous.

The XUI resource DwtNchangeVisAtts is not supported.

Check your application for the use of this resource and make changes in order to obtain the desired appearance and operation of your widget.

3.2.2.10 Scale Widget

The scale widget (XmScale) is equivalent to the XUI scale widget (DwtScale). If you are using the scale widget, check your application for the use of the following resources. You might have to make some changes in order to obtain the desired appearance and operation of your widget.

- XmNorientation — The default is to display a vertical scale.
- XmNprocessingDirection — The default is to display the maximum value on top.
- XmNshowValue — The default is to not display a slider value.

The following XUI resources are not supported:

- DwtNshowValueAutomatic
- DwtNslider
- DwtNsliderPixmap

3.2.2.11 Scroll Bar Widget

The scroll bar widget (XmScrollBar) is equivalent to the XUI scroll bar widget (DwtScrollBar). If you are using the scroll bar widget, check your application for the use of the XmNorientation resource. This resource controls how the scroll bar is displayed. You might have to make some changes in order to obtain the desired appearance and operation of your widget. See Section 3.2.9 for more information on scroll bars.

The following XUI resources are not supported:

- DwtNtranslations1
- DwtNtranslations2

3.2.2.12 Scrolled Window Widget

The scrolled window widget (XmScrolledWindow) is equivalent to the XUI scroll window widget (DwtScrollWindow). If you are using the scrolled window widget, check your application for the use of the XmNscrollingPolicy resource. This resource controls the scrolling of the work area. You must choose a value for this resource. See the *OSF/Motif Programmer's Reference* for a description of this resource and possible values.

3.2.2.13 Separator Widget

The separator widget (XmSeparator) is equivalent to the XUI separator widget (DwtSeparator). If you are using the separator widget, check your application for the use of the XmNmargin resource. This resource specifies the left-right or top-bottom spacing, depending on the value of XmNorientation. See the *OSF/Motif Programmer's Reference* for a description of this resource and possible values.

3.2.2.14 Text Widget

The text widget (XmText) is equivalent to the XUI simple text widget (DwtSText). If you are using the text widget, check your application for the use of the XmNlosingFocusCallback resource. This resource is the list of callbacks called before the widget loses input focus. When the widget loses focus, the callback reason is XmCR_LOSING_FOCUS.

The XUI resource DwtNhalfBorder is not supported.

3.2.3 Digital Extended Motif Widgets

The include files and resources have changed for the following Digital Extended Motif widgets:

- Color mixing
- Compound string
- Help
- Print
- Structured Visual Navigation (SVN)

See the *VMS DECwindows Motif Guide to Application Programming* for a description of these widgets and their resources.

If you are using UIL, add the following line to your application after the call to MrmInitialize:

```
DXmInitialize();
```

This calls MrmRegisterClass for all Digital Extended Motif widgets. For application portability to operating systems that do not support shared libraries, call MrmRegisterClass for each widget used instead of calling MrmInitialize and DXmInitialize. This reduces the size of the application images.

3.2.3.1 Color Mixing Widget

If your application used the XUI color mix widget, do the following:

- Add the following header file to your program:

  ```
  #include <DXm/DXmColor.h>
  ```

- Check all resource names.

3.2.3.2 Compound String Widget

If your application used the XUI compound string text widget, do the following:

- Add the following header file to your program:

  ```
  #include <DXm/DXmCSText.h>
  ```

- Check all resource names.

3.2.3.3 Help Widget

If your application used the XUI help widget, you can use either of the following:

- HyperHelp
- Help widget

HyperHelp

HyperHelp uses Bookreader windows as its display window. Users can invoke HyperHelp from the Help menu, the help command, the Help push button, or the Help key.

HyperHelp includes the following features:

- Proportional fonts, which are more legible than fixed fonts
- Graphics
- Formatted tables
- Hotspots (The ability to move around in the Help text by clicking on parts of the text)
- The ability to create links between the online help and other pieces of information, such as a mail message and other Bookreader topics

Help Widget

If you are using the help widget, do the following:

- Add the following header file to your program:

  ```
  #include <DXm/DXmHelpB.h>
  ```

- Check all resource names, particularly XmNfontList resource.

  Note

  You cannot change the XmNtextFontList, XmNlabelFontList, and XmNbuttonFontList resources by using XtSetValues.

3.2.3.4 Print Widget

The print widget is a modeless widget that provides DECwindows applications with a fast, convenient method to print one or more files in multiple formats. If you want to use the print widget, do the following:

- Add the following header file to your program:

  ```
  #include <DXm/DXmPrint.h>
  ```

- Check all resource names.

3.2.3.5 Structured Visual Navigation Widget

The Structured Visual Navigation (SVN) widget presents the user with data in a hierarchical structure. The user can navigate in and select data from the structure. Your application is responsible for creating the hierarchy and supplying the data to the SVN widget; the actual data in the hierarchy is transparent to the SVN widget. If you want to use Structured Visual Navigation widget, do the following:

- Add the following header file to your program:

  ```
  #include <DXm/DXmSvn.h>
  ```

- Check all resource names.

3.2.4 Custom Widgets (Widget Programmers Only)

If you are a widget programmer and are porting custom widgets to Motif, check the following:

- Widget hierarchy
- XmPrimitive class record
- Bit gravity
- Keyboard focus

3.2.4.1 Widget Hierarchy

In the Motif Toolkit widget hierarchy, XmManager is the superclass for all composite widgets, XmPrimitive is the superclass for all primitive widgets, and XmGadget is the superclass for all gadgets.

Study the Motif widget hierarchy carefully. If your widget is a subclass of an XUI widget class, you might have to recode your program's structure declarations and resources in order to get your program to run.

See the *OSF/Motif Programmer's Guide* for a complete map of the OSF/Motif widget hierarchy.

3.2.4.2 XmPrimitive Class Record

The filters convert custom widgets that are a subclass of DwtCommon to a subclass of XmPrimitive. You must change the class record initialization to finish the conversion.

For example:

Code output from filters

```
{  /* dwt common class   */
   /* Pad0               */     _XtInherit,
   /* Pad1               */     _XtInherit,     ❶
   /* Pad2               */     _XtInherit,
   /* extension          */     NULL,
},
```

Code after additional changes

```
{  /* XmPrimitive class                          */
   /* XtWidgetProc          border_highlight   */   _XtInherit,      ❶
   /* XtWidgetProc          border_unhighlight */   _XtInherit,
   /* XtTranslations        translations       */   XtInheritTranslations,
   /* XtActionProc          arm_and_activate   */   NULL,
   /* XmSyntheticResource * syn_resources      */   NULL,
   /* int                   num_syn_resources  */   NULL,
   /* caddr_t               extension          */   NULL,
},
```

❶ Replace the five lines of DwtCommon widget class records with the eight lines of XmPrimitive widget class records.

3.2.4.3 Bit Gravity

If your widget is a subclass of a Motif widget and inherits the realize procedure from the superclass, the widget also inherits the bit gravity from the superclass. In Motif, this is typically NorthWestGravity.

3.2.4.4 Keyboard Focus

Remove any explicit calls to XSetInputFocus or XtSetKeyboardFocus from your widget. Use XmProcessTraversal instead.

See the *OSF/Motif Programmer's Guide* for a complete description of keyboard input focus models.

3.2.5 Callback Records

The callback records have changed, as shown in the following example:

XUI data structure

```
typedef struct {
  VoidProc    proc;  ❶
  int         tag;   ❷
} DwtCallback, *DwtCallbackPtr;

typedef struct {
  XtCallbackProc  callback;
  caddr_t         closure; ❷ ❸
} XtCallbackRec, *XtCallbackList;
```

Motif data structure

```
typedef struct {
  XtCallbackProc  callback; ❶
  XtPointer       closure; ❷ ❸
} XtCallbackRec, *XtCallbackList;
```

To reference fields in the callback record, make the following changes in your application:

❶ Change the casting *VoidProc* to *XtCallbackProc*.

❷ Change the casting *int* to *XtPointer* and the field name *tag* to *closure*.

❸ Change the casting *caddr_t* to *XtPointer*.

The following example shows how to reference fields in the callback record:

Code output from filters

```
static XtCallbackRec  vert_scroll_callback[2] =
{
  { (VoidProc) local_scroll_callback, k_vertical }, ❶
  { NULL }
};
static XtCallbackRec  horiz_scroll_callback[2] =
{
  { (VoidProc) local_scroll_callback, k_horizontal }, ❶
  { NULL }
};
```

Code after additional changes

```
static XtCallbackRec  vert_scroll_callback[2] =
{
  { (XtCallbackProc)) local_scroll_callback, k_vertical }, ❶
  { NULL }
};
static XtCallbackRec  horiz_scroll_callback[2] =
{
  { (XtCallbackProc) local_scroll_callback, k_horizontal }, ❶
  { NULL }
};
```

❶ *VoidProc* is changed to *XtCallbackProc*.

3.2.6 Callback Structures

The castings and field names in the callback data structure have changed for the following widgets:

- Command
- Drawing area
- File selection box
- List
- Row column
- Selection box
- Toggle button

The following sections show the difference between the XUI and Motif data structures and describe the changes.

3.2.6.1 Command Widget

The callback data structure for the command widget has changed from the XUI command window widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int     reason;
  XEvent  *event;
  int     length;
  char    *value; ❶
} DwtCommandWindowCallbackStruct;
```

Motif data structure

```
typedef struct {
  int        reason;
  XEvent     *event;
  XmString   value; ❶
  int        length;
} XmCommandCallbackStruct;
```

If you reference this structure, make the following change in your application, as needed:

❶ Change the casting for the field name **value* from *char* to *XmString*.

3.2.6.2 Drawing Area Widget

The callback data structure for the drawing area widget has changed from the XUI window widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int          reason;
  XExposeEvent *event; ❶
  Window       w; ❷
} DwtWindowCallbackStruct;
```

Motif data structure

```
typedef struct {
  int     reason;
  XEvent  *event; ❶
  Window  window; ❷
} XmDrawingAreaCallbackStruct;
```

If you reference this structure, make the following changes in your application, as needed:

❶ Change the casting for the **event* field from *XExposeEvent* to *XEvent*.

❷ Change the field name *w* to *window*.

3.2.6.3 File Selection Box Widget

The callback data structure for the file selection box widget has changed from the XUI file selection widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int            reason;
  XEvent         *event;
  DwtCompString  value;
  int            value_len;❶
  DwtCompString  dirmask;❷
  int            dirmask_len;❸
} DwtFileSelectionCallbackStruct;
```

Motif data structure

```
typedef struct {
  int           reason;
  XEvent        *event;
  XmString      value;
  int           length;❶
  XmString      mask;❷
  int           mask_length;❸
  XmString      dir;
  int           dir_length;❹
  XmString      pattern;
  int           pattern_length;
} XmFileSelectionBoxCallbackStruct;
```

If you reference this structure, make the following changes in your application, as needed:

❶ Change the field name *value_len* to *length*.

❷ Change the field name *dirmask* to *mask*.

❸ Change the field name *dirmask_len* to *mask_length*.

❹ You can now reference these additional fields in the callback structure.

Note

The porting filters change the casting of the *value* and *mask* fields from *DwtCompString* to *XmString*.

3.2.6.4 List Widget

The callback data structure for the list widget has changed from the XUI list box widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int            reason;
  XEvent         *event;
  DwtCompString  item;
  int            item_length;
  int            item_number; ❶
} DwtListBoxCallbackStruct;
```

Motif data structure

```
typedef struct {
  int       reason;
  XEvent    *event;
  XmString  item;
  int       item_length;
  int       item_position; ❶
  XmString  *selected_items;
  int       selected_item_count; ❷
  int       *selected_item_positions;
  char      selection_type;
} XmListCallbackStruct;
```

If you reference this structure, make the following changes to your application, as needed:

❶ Change the field name *item_number* to *item_position*.

❷ You can now reference these additional fields in the callback structure.

Note

The porting filters change the casting of the *item* field from *DwtCompString* to *XmString*.

3.2.6.5 Row Column Widget

The callback data structure for the row column widget has changed from the XUI radio box widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int     reason;
  XEvent  *event;
  Widget  s_widget; ❶
  char    *s_tag;      ❷
  char    *s_callbackstruct; ❸
} DwtRadioBoxCallbackStruct;
```

Motif data structure

```
typedef struct {
  int     reason;
  XEvent  *event;
  Widget  widget; ❶
  char    *data;    ❷
  char    *callbackstruct; ❸
} XmRowColumnCallbackStruct;
```

If you reference this structure, make the following changes to your application, as needed:

❶ Change the field name *s_widget* to *widget*.

❷ Change the field name **s_tag* to **data*.

❸ Change the field name **s_callbackstruct* to **callbackstruct*.

3.2.6.6 Selection Box Widget

The callback data structure for the selection box widget has changed from the XUI file selection widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int            reason;
  XEvent         *event;
  DwtCompString  value;
  int            value_len; ❶
} DwtSelectionCallbackStruct;
```

Motif data structure

```
typedef struct {
  int          reason;
  XEvent       *event;
  XmString     value;
  int          length; ❶
} XmSelectionBoxCallbackStruct;
```

If you reference this structure, make the following change, as needed:

❶ Change the field name *value_len* to *length*.

Note

The porting filters change the casting of the *value* field from *DwtCompString* to *XmString*.

3.2.6.7 Toggle Button Widget

The callback data structure for the toggle button widget has changed from the XUI toggle button widget structure, as shown in the following example:

XUI data structure

```
typedef struct {
  int     reason;
  XEvent  *event;
  int     value; ❶
} DwtTogglebuttonCallbackStruct;
```

Motif data structure

```
typedef struct {
  int     reason;
  XEvent  *event;
  int      set; ❶
} XmToggleButtonCallbackStruct;
```

If you reference this structure, make the following change, as needed:

❶ Change the field name *value* to *set*.

3.2.7 Compound String Usage

If your application uses a compound string as the value of a resource in an argument list, you must free the compound string memory, using XmStringFree. To do this, you would call XmStringFree after the argument list is passed to the widget, whether at widget creation or using XtSetValues. For example:

```
Arg      al[2];      /* Argument list */
int      ac = 0;     /* Argument count */
XmString label_cs;

label_cs = XmStringCreateSimple ("Cancel");
XtSetArg(al[ac], XmNlabelString, label_cs); ac++;
XtSetArg(al[ac], XmNactivateCallback, callback); ac++;
p = XmCreatePushButton (parent, "", al, ac);
XtManageChild (p);
XmStringFree (label_cs); ❶
```

❶ The string *label_cs* is freed after widget creation.

If your application uses XtGetValues to retrieve a compound string, you must free the returned copy of the compound strings after using it by calling XmStringFree. However, some compound string values (XmString and XmStringTable) are not copied. Table 3–3 lists the resources whose compound string values are not copied. Do not free these resources.

Table 3–3 Compound String Resources Not to Be Freed

When using this widget:	Do not free these resources:	Resource type:
XmBulletinBoard	XmNdialogTitle	XmString
XmFileSelectionBox	XmNdirectory XmNnoMatchString	XmString
XmList	XmNitems XmNselectedItems	XmStringTable
XmRowColumn	XmNlabelString	XmString
XmScale	XmNtitleString	XmString

3.2.8 Resolution Independence

Motif extends the resolution independence mechanism to all widgets and gadgets. Now, applications can create and display images that are the same physical size on any display. Use the XmNunitType resource to specify the following unit types for widgets and gadgets:

- XmPIXELS (default)
- Xm100TH_MILLIMETERS
- Xm1000TH_INCHES
- Xm100TH_POINTS
- Xm100TH_FONT_UNITS

For example, if the XmNunitType resource is specified as Xm1000TH_INCHES, all values passed to the widgets are treated as 1/1000 of an inch.

The Motif font units are not equivalent to the XUI font units. If you used XUI font positioning, you will have to make some adjustments to the x and y values of dialog box children.

See the *OSF/Motif Programmer's Guide* for more information on the resolution independence mechanism.

3.2.9 Scroll Bars

In Motif, when you place an XmText or XmList widget inside of an XmScrolledWindow widget, the XmScrolledWindow widget creates scroll bars. Scroll bars can be placed vertically, on either the left or right, and horizontally, on either the top or bottom.

Motif provides two convenience functions to create scroll bars:

- **XmCreateScrolledList** — Creates an XmList widget inside an XmScrolledWindow, and attaches XmList-specific callbacks to the scroll bars. The argument list provided with XmCreateScrolledList is passed to the XmList widget, which interprets the following arguments to control the scroll bars:
  - XmNlistSizePolicy — Specifies whether to change the size of the list area if a list item is larger than the list area
  - XmNscrollBarDisplayPolicy — Together with XmNlistSizePolicy, specifies when to display horizontal and vertical scroll bars
- **XmCreateScrolledText** — Creates an XmText widget inside an XmScrolledWindow widget, and attaches XmText-specific callbacks to the scroll bars. The argument list provided with XmCreateScrolledText is passed to the XmText widget, which interprets the following arguments to control the scroll bars:
  - XmNscrollHorizontal and XmNscrollVertical — Specify whether or not to provide horizontal or vertical scroll bars
  - XmNscrollLeftSide and XmNscrollTopSide — Specify scroll bar placement

3.2.10 Widget Layout

Motif widgets are usually larger than XUI widgets because of the highlighting required to support keyboard traversal. Therefore, be sure to check the layout of your widgets on the display screen and make any adjustments.

3.3 Reviewing Intrinsics Features and Making Changes

Motif uses MIT X Window System Release 4 Intrinsics. The change from Release 3 to Release 4 might result in binary and compile-time incompatibilities. To minimize the effect of these incompatibilities, do the following:

- Specify an application context
- Review X11 Release 4 changes
- Make changes to your application before you compile it.

3.3.1 Specifying Application Context

Every program needs an application context. An application can explicitly name an application context or use the default application context. Applications that explicitly name an application context have the following characteristics:

- Are more portable than applications that use the default application context.
- Can create windows on more than one display device. The Intrinsics gather events from all displays specified.
- Use different routines. Release 4 provides routines to use if you explicitly name an application context and compatibility routines for applications that continue to use the default application context. Appendix C lists the routines for each case.

Note

Do not mix explicit application contexts and default application contexts in the same program.

3.3.2 Reviewing X11 Release 4 Changes

The X11 Release 4 Intrinsics contain changes to the following:

- Routine interfaces
- Widget semantics

3.3.2.1 Routine Interfaces

Review the following changes to the routine interfaces and modify your code accordingly:

- **Application shell names** — Release 3 gets the shell name from the XtInitialize *name* parameter; Release 4 does not (it uses NULL). To name the shell, do one of the following:
  - Instead of using XtInitialize, call the three Intrinsics functions that make up XtInitialize (XtToolkitInitialize, XtOpenDisplay, and XtAppCreateShell) individually.

– Set the string that appears in the application's title bar by setting the XtNtitle resource in the argument list and calling XtSetValues on the shell widget returned by XtInitialize. For example:

```
toplevel  = XtInitialize (buffer, "testclass", NULL, 0, &argc, argv);
ac = 0;                                                                    ❶
XtSetArg (al[ac], XtNtitle, buffer); ac++; /* buffer contains the string */
XtSetArg (al[ac], XtNiconName, buffer); ac++;  ❷
XtSetValues (toplevel, al, ac);  ❸
```

❶ Defines the title resource.

❷ Defines the icon name.

❸ Alters the main window's resources.

- **XtCancelSelectionCallbackProc parameter** — Release 4 deleted this parameter from calls to XtGetSelectionValueIncremental and XtGetSelectionValuesIncremental. A canceled selection calls XtSelectionCallbackProc with the type parameter value XT_CONVERT_FAIL.
- **XtDisplayInitialize** — Release 3 Intrinsics accept a NULL application context and use the default application context; Release 4 does not.
- **XtNameToWidget** — The name parameter differs in meaning. Release 3 expects the first name in the qualified name to be the root widget; Release 4 expects the first name in the qualified name to be a child of the root widget.

3.3.2.2 Widget Semantics

Review the following changes to the widget semantics and modify your application accordingly:

Note

Widgets created using DWT low- and high-level routines will retain the Release 3 behavior.

- **Accelerator actions** — Release 3 invokes accelerator actions on insensitive widgets; Release 4 does not.

- **Callbacks** — The use of Release 3 widget code to specify callbacks using XtSetValues might not work with the Release 4 Intrinsics; Release 4 correctly handles callbacks passed to XtSetValues. If your widget's set_values routine compares the old and new values of a callback field and attempts to replace the callbacks, remove that code from your application; the Intrinsics now do this for you.
- **Class extension record** — Release 4 requires you to set a flag in order to allow gadget children for a widget class (see the *MIT R4 Intrinsics Specification*).
- **Event handlers, widget instance initialization procedures, and widget set_values procedures** — Release 4 passes additional parameters. These parameters can be ignored and the Release 3 behavior can be retained (see the *MIT R4 Intrinsics Specification*).
- **Incremental selection callbacks** — Parameters use a different passing mechanism. In XtConvertSelectionIncrProc, XtSelectionDoneIncrProc, and XtCancelConvertSelectionProc, Release 3 passes *receiver-id* by value; Release 4 passes it by reference.

  In XtConvertSelectionIncrProc, Release 3 passes *max_length* by value; Release 4 passes it by reference.
- **Instance fields** — Release 4 adds new fields to the end of WMShell and VendorShell instance parts. If you used DwtResolvePartOffsets in your application, this will not affect you. Otherwise, you will have to check the offsets used to access fields in the top-level and application shells.

3.3.3 Before Compiling Your Application

Before compiling your application, make changes to the following files:

- **Header files**

  The Intrinsic.h file contains information that existed in four XUI public .h files (Convert.h, Translate.h, Selection.h, Event.h). Although these files are retained for upward compatibility, if your application includes these files, include Intrinsic.h instead.

The XUI public .h files (CompObj.h, CompObjP.h, WindowObj.h, and WindowObjP.h) no longer exist. Although these files are retained for upward compatibility, do not use them.

- **Xdefaults files (ULTRIX systems)**

  On ULTRIX systems, the leading period character (.) is no longer required as part of the Xdefaults file name for applications. For example, note the difference between the Xdefaults file for the clock application under XUI and Motif:

  XUI
  `/usr/users/fred/.DXclock`

  Motif
  `/usr/users/fred/DXclock`

3.4 Compiling and Linking the Application

To compile and link your application, read the section that applies to your operating system.

VMS Systems

VMS

If your application uses a UIL specification file to specify the user interface, type the following to run the UIL compiler:

```
$ UIL/MOTIF file_name.UIL
```

Table 3–4 lists the shareable images required to link your application. These images are in the SYS$SHARE directory.

Table 3–4 VMS Shareable Images Required for Linking

Shareable Image	What It Contains
DECW$DWTLIBSHR.EXE	Xt library (intrinsics) and XUI library
DECW$XMLIBSHR.EXE	Xm and Mrm library
DECW$XLIBSHR.EXE	X library
DECW$DXMLIBSHR.EXE	DXm library (Digital extensions to Motif)

ULTRIX

ULTRIX Systems (RISC)

To compile applications on RISC systems, include the files in Table 3–5 on the cc command line

Table 3–5 ULTRIX Include Files (RISC Systems)

Include File Library Name	What It Contains
/usr/lib/DXM/lib/Xt/X11	Intrinsics include files
/usr/lib/DXM/lib/Xm	Motif widget set
/usr/lib/DXM/lib/Mrm	Motif Resource Manager
/usr/lib/DXM/lib/DXm	Digital extensions to Motif

For example:

```
system> cc file_name.c -I/usr/lib/DXM/lib/Xt/X11\
  -I/usr/lib/DXM/lib/Xm -I/usr/lib/DXM/lib/Mrm\
  -I/usr/lib/DXM/lib/DXm
```

To run the UIL compiler on RISC systems, type the following:

```
system> /usr/lib/DXM/clients/UIL/uil file_name.uil
```

Table 3–6 lists the libraries required to link your application on RISC systems running the ULTRIX operating system.

Table 3–6 ULTRIX Libraries Required for Linking (RISC Systems)

Object Library Name	What It Contains
/usr/lib/libX11.a	X library
/usr/lib/DXM/lib/Xt/libXt.a	Xt library
/usr/lib/DXM/lib/Xm/libXm.a	Xm library
/usr/lib/DXM/lib/Mrm/libMrm.a	Mrm library
/usr/lib/DXM/lib/DXm/libDXm.a	DXm library

ULTRIX Systems (VAX)

To compile applications on VAX systems, include the files in Table 3–7 on the cc command line.

Table 3–7 ULTRIX Include Files (VAX Systems)

Include File Library Name	What It Contains
/usr/lib/DXV/lib/Xt/X11	Intrinsics include files
/usr/lib/DXV/lib/Xm	Motif widget set
/usr/lib/DXV/lib/Mrm	Motif Resource Manager
/usr/lib/DXV/lib/DXm	Digital extensions to Motif

For example:

```
system> cc file_name.c -I/usr/lib/DXM/lib/Xt/X11\
  -I/usr/lib/DXM/lib/Xm -I/usr/lib/DXM/lib/Mrm\
  -I/usr/lib/DXM/lib/DXm
```

To run the UIL compiler on VAX systems, type the following:

```
system> /usr/lib/DXV/clients/UIL/uil file_name.uil
```

Table 3–8 lists the libraries required to link your application on VAX systems running the ULTRIX operating system.

Table 3–8 ULTRIX Libraries Required for Linking (VAX Systems)

Object Library Name	What It Contains
/usr/lib/libX11.a	X library
/usr/lib/DXV/lib/Xt/libXt.a	Xt library
/usr/lib/DXV/lib/Xm/libXm.a	Xm library
/usr/lib/DXV/lib/Mrm/libMrm.a	Mrm library
/usr/lib/DXV/lib/DXm/libDXm.a	DXm library

3.5 Testing the Application

To test your application after you have compiled and linked it, do the following:

- Start the application.
- Look at all screen text. Is any text jumbled or missing?

- Look at dialog boxes. Do they look right?

  Do any fields or buttons overlay each other? Remember that Motif font units are not equivalent to the XUI font units.

  Are the defaults correct? You might have to explicitly set defaults.

  Test dialog box operation to see if everything worked as before the conversion.
- Check icons. Do they look all right? Do they work correctly?
- Test different menu operations. Does everything work as it did before the conversion?
- Check scrolling, keyboard traversal, and mouse button and mnemonics operations.
- Check context-sensitive help, if your application supports it.
- Resize the application windows.
- Test print operations, if your application supports it.
- Test different error conditions.
- Stop the application.

Make any necessary changes before going on to the next sections.

3.6 Making Changes Required by the Motif Window Manager

To make the changes required by the Motif Window Manager (MWM), do the following:

- Check MWM-client interaction
- Check keyboard focus
- Check key and mouse bindings

3.6.1 Check MWM-Client Interaction

The rules governing the relationship between the MWM and the client applications are defined in the *Inter-Client Communications Conventions Manual*. Your application must follow these rules so its operation is consistent with all Motif applications. See the *X Window System* for more information.

3.6.2 Checking Keyboard Focus

Keyboard focus is determined by the MWM based on the focus model chosen. Motif has two focus policies: explicit and pointer; XUI has only one: explicit. The focus model is set by the keyboardFocusPolicy resource. The default is explicit, but the user can override this in the Xdefaults file.

See the *OSF/Motif Programmer's Guide* for a complete description of keyboard input focus models.

3.6.3 Checking Key and Mouse Bindings

The user can perform window management functions using either a mouse or a keyboard. Both methods must be available to the application user. MWM has a default button and keyboard bindings for this purpose. You must change your application's translation tables to reflect these defaults.

A user can close a window by selecting the Close selection from the Window menu or by double-clicking the Select mouse button on the Window menu button. If the user closes the window, make sure your application performs those tasks that normally occur when a user ends the application. See the *OSF/Motif Programmer's Guide* for a description of MWM protocols and functions.

3.7 Make Changes Required by the *OSF/Motif Style Guide*

To make the changes required by the *OSF/Motif Style Guide*, do the following:

- Add menu mnemonics
- Add keyboard accelerators, if you want them
- Add keyboard traversal
- Check context-sensitive pop-up menu operation

For additional information on Motif style, see the *OSF/Motif Style Guide* and the *DECwindows Companion to the OSF/Motif Style Guide*.

3.7.1 Adding Menu Mnemonics

In Motif, pull-down menu selections use mnemonics, a single, underlined character that enables experienced users to make a selection from the keyboard. For consistency with other Motif applications, your pull-down menus must have mnemonics. To add a mnemonic to your pull-down menus, use the XmNmnemonic resource. For example:

```
/* C code */
XtSetArg(args[0], XmNmnemonic, 'character');


/* UIL code */
arguments {XmNlabelString = "Help"; XmNmnemonic = keysym('H');};
```

See the *OSF/Motif Programmer's Guide* for more information on mnemonics.

Note

The mnemonic character must match the label character in the label string exactly in order for the mnemonic to work.

3.7.2 Adding Keyboard Accelerators

Pull-down menu selections can also have accelerators. These enable a user to make a menu selection by using a single key or key sequence, without having the menu displayed.

You can use any key not already reserved for keyboard traversal as a keyboard accelerator. If you want to add a keyboard accelerator to your application pull-down menu selection, use the XmNaccelerator and XmNacceleratorText resources.

The following example defines the Control and O keys as the keyboard accelerator for an associated menu selection:

```
/*C code */
cs = XmStringCreateSimple("Ctrl+O");
XtSetArg(args[0], XmNaccelerator, "Ctrl<key>o:");
XtSetArg(args[1], XmNacceleratorText, cs);


/* UIL code */
arguments{
          XmNaccelerator     = "Ctrl<key>o:";
          XmNacceleratorText = compound_string('Ctrl+O');
         };
```

See the *OSF/Motif Programmer's Guide* for more information on keyboard accelerators.

Note

You can place keyboard accelerators only on direct children of a RowColumn class widget; the RowColumn parent must also be a pop-up or pull-down type.

3.7.3 Adding Keyboard Traversal

A user can use the keyboard to navigate among primitive widgets and their children. To enable this action, you must arrange the primitive widgets into tab groups. After the widgets have been arranged into tab groups, an application user can change the focus to different tab groups and to different widgets within a tab group.

The toolkit automatically calculates tab groups and sets the XmNtraversalOn resource default to TRUE for all widgets and gadgets except for the following:

- XmLabel
- XmLabelGadget
- XmScrollBar
- XmSeparator
- XmSeparatorGadget

To enable traversal for these widgets and gadgets, add the following line to the argument list for the widget:

```
XtSetArg(al[ac], XmNtraversalOn, True); ac++;
```

See the *OSF/Motif Programmer's Guide* for more information on keyboard traversal.

3.7.4 Checking Context-Sensitive Pop-up Menus

Context-sensitive pop-up menus are activated by MB3 instead of MB2. You must change your application to comply with this style.

4

Giving Information to Application Users

As the last step in the porting process, you need to give your application users information about the application and OSF/Motif. This involves updating the application and system documentation.

4.1 Updating the Documentation

After you are finished testing your application and making any changes, update the following:

- **Application build procedures** — Change the include files and other file names, if you have not already done it.
- **Application user manual** — Remove any reference to XUI or XUI operations; change the description of how the application works, if necessary; change routine names; and change any screen pictures to reflect the Motif look.
- **Installation procedures and manual** — You might want to change the file names, so you do not overwrite the XUI files for the application; the location of the installed files, to keep them separate from the XUI files; and any screen pictures, to reflect the Motif look.
- **Release notes** — If you use release notes, list any behavior changes, routine name changes, and new restrictions.

A

Summary of XUI and OSF/Motif Differences

This appendix summarizes the differences between XUI and OSF/Motif for the following in the following areas:

- Component names
- Terminology
- Windows and window managers
- Menus and menu items
- Standard message boxes
- Mouse button bindings

A.1 Component Names

This section summarizes name changes for the following OSF/Motif components:

- Widget classes
- Functions
- Resources
- Enumeration literals
- Callback reasons
- Compound strings
- Fontlists
- Clipboards
- Resource manager functions

For complete descriptions of the widget classes, see the *OSF/Motif Programmer's Reference*.

A.1.1 Widget Classes

Table A–1 summarizes the differences between the XUI widget hierarchy and the OSF/Motif widget hierarchy.

Table A–1 Widget Class Name Changes

XUI	Motif
DwtAttachedDB	XmForm
DwtCommandWindow	XmCommand
DwtCommon	[1]
DwtDialogBox	XmBulletinBoard
DwtFileSelection	XmFileSelectionBox
DwtHelp	DXmhelp
DwtLabel	XmLabel
DwtListBox	XmList
DwtMainWindow	XmMainWindow
DwtMenu	XmRowColumn
DwtMessageBox	XmMessageBox
DwtPullDownMenuEntry	XmCascadeButton
DwtPushButton	XmPushButton
DwtScale	XmScale
DwtScrollBar	XmScrollBar
DwtScrollWindow	XmScrolledWindow
DwtSelection	XmSelectionBox
DwtSeparator	XmSeparator
DwtSText	XmText
DwtToggleButton	XmToggleButton
DwtWindow	XmDrawingArea

[1] No equivalent in Motif. The resources are in XmPrimitive, XmManager, and XmGadget.

A.1.2 Function Names

Table A–2 summarizes the differences between the XUI function names and the OSF/Motif function names. If you change a create function name, you might have to change the widget class names.

Table A–2 Function Name Changes

XUI	Motif
Dwt*Create	XmCreate*[1]
DwtAttachedDBCreate	XmCreateForm
DwtAttachedDBPopupCreate	XmCreateFormDialog
DwtCautionBoxCreate	XmCreateWarningDialog, XmCreateMessageDialog, XmCreateErrorDialog, or XmCreateQuestionDialog
DwtCommandAppend	XmCommandAppendValue
DwtCommandErrorMessage	XmCommandError
DwtCommandSet	XmCommandSetValue
DwtCommandWindowCreate	XmCreateCommand
DwtDialogBoxCreate	XmCreateBulletinBoard
DwtDialogBoxPopupCreate	XmCreateBulletinBoardDialog
DwtFileSelectionCreate	XmCreateFileSelectionDialog
DwtLabelCreate	XmCreateLabel
DwtLabelGadgetCreate	XmCreateLabelGadget
DwtListBoxCreate	XmCreateList
DwtMainWindowCreate	XmCreateMainWindow
DwtMenuBarCreate	XmCreateMenuBar
DwtMenuCreate	XmCreateRowColumn
DwtMenuPopupCreate	XmCreatePopupMenu
DwtMenuPulldownCreate	XmCreatePulldownMenu

[1]Most of the name changes follow this form. The table lists those function name changes that do not follow this form.

(continued on next page)

Table A–2 (Cont.) Function Name Changes

XUI	Motif
DwtMessageBoxCreate	XmCreateInformationDialog[2]
DwtOptionMenuCreate	XmCreateOptionMenu
DwtPullDownMenuEntryCreate	XmCreateCascadeButton
DwtPullDownMenuEntryHilite	XmCascadeButtonHighlight
DwtPullEntryGadgetCreate	XmCreateCascadeButtonGadget
DwtPushButtonCreate	XmCreatePushButton
DwtPushButtonGadgetCreate	XmCreatePushButtonGadget
DwtRadioBoxCreate	XmCreateRadioBox
DwtScaleCreate	XmCreateScale
DwtScaleGetSlider	XmScaleGetValue
DwtScaleSetSlider	XmScaleSetValue
DwtScrollBarCreate	XmCreateScrollBar
DwtScrollBarGetSlider	XmScrollBarGetValues
DwtScrollBarSetSlider	XmScrollBarSetValues
DwtScrollWindowCreate	XmCreateScrolledWindow
DwtSelectionCreate	XmCreateSelectionBox
DwtSeparatorCreate	XmCreateSeparator
DwtSeparatorGadgetCreate	XmCreateSeparatorGadget
DwtSTextCreate	XmCreateText
DwtToggleButtonCreate	XmCreateToggleButton
DwtToggleButtonGadgetCreate	XmCreateToggleButtonGadget
DwtWindowCreate	XmCreateDrawingArea
DwtWorkBoxCreate	XmCreateWorkingDialog[2]

[2]Instantiates an XmMessageBox widget inside an XmDialogShell widget. To instantiate only the XmMessageBox widget, use XmCreateMessageBox.

A.1.3 Resource Names

Table A–3 summarizes the differences between the XUI resource names and the OSF/Motif resource names. Some XUI resource names have multiple Motif resource names. To help you determine which Motif resource name applies to your widget, the widget class is listed in parentheses after the Motif name.

Table A–3 Resource Name Changes

XUI	Motif
DwtN*	XmN*[1]
DwtNactivateCallback	XmNokCallback (XmSelectionBox)
DwtNadb*	XmN*[1]
DwtNapplyLabel	XmNapplyLabelString
DwtNautoShowInsertPoint	XmNautoShowCursorPosition
DwtNbuttonAccelerator	XmNaccelerator
DwtNcancelLabel	XmNcancelLabelString
DwtNchildOverlap	XmNallowOverlap
DwtNcols	XmNcolumns
DwtNconformToText	XmNrecomputeSize
DwtNdefaultHorizontalOffset	XmNhorizontalSpacing
DwtNdefaultPushbutton	XmNdefaultButtonType
DwtNdefaultVerticalOffset	XmNverticalSpacing
DwtNdirectionRtoL	XmNprocessingDirection (XmScale, XmScrollBar)
DwtNdirectionRtoL	XmNstringDirection (XmLabel, XmBulletinBoard, XmList)
DwtNextendCallback	XmNextendedSelectionCallback
DwtNfilterLabel	XmNfilterLabelString
DwtNfont	XmNfontList (XmLabel, XmList, XmScale, XmText)
DwtNfont	XmN*fontList (XmBulletinBoard)
DwtNhistory	XmNhistoryItems

[1]Most of the name changes follow this form. The table lists those resource name changes that do not follow this form.

(continued on next page)

Table A–3 (Cont.) Resource Name Changes

XUI	Motif
DwtNhorizontal	XmNscrollBarDisplayPolicy
DwtNhotSpotPixmap	XmNcascadePixmap
DwtNiconPixmap	XmNsymbolPixmap
DwtNinc	XmNincrement
DwtNindicator	XmNindicatorOn
DwtNinsensitivePixmap	XmNlabelInsensitivePixmap
DwtNinsensitivePixmapOff	XmNlabelInsensitivePixmap
DwtNinsensitivePixmapOn	XmNselectInsensitivePixmap
DwtNinsertionPointVisible	XmNcursorPositionVisible
DwtNinsertionPosition	XmNcursorPosition
DwtNitems	XmNlistItems
DwtNitemsCount	XmNitemCount (XmList)
DwtNitemsCount	XmNlistItemCount (XmSelectionBox)
DwtNlabel	XmNlabelString (XmLabel, XmRowColumn)
DwtNlabel	XmNlistLabelString (XmSelectionBox)
DwtNlabel	XmNmessageString (XmMessageBox)
DwtNlabelAlignment	XmNmessageAlignment
DwtNlines	XmNhistoryItemCount
DwtNlostFocusCallback	XmNlosingFocusCallback
DwtNmaxValue	XmNmaximum
DwtNmenuAlignment	XmNisAligned
DwtNmenuEntryClass	XmNentryClass
DwtNmenuExtendLastRow	XmNadjustLast
DwtNmenuIsHomogeneous	XmNisHomogeneous
DwtNmenuNumColumns	XmNnumColumns
DwtNmenuPacking	XmNpacking
DwtNmenuRadio	XmNradioBehavior
DwtNmenuType	XmNrowColumnType

(continued on next page)

Table A–3 (Cont.) Resource Name Changes

XUI	Motif
DwtNmergeTextTranslations	XmNtextTranslations
DwtNminValue	XmNminimum
DwtNokLabel	XmNokLabelString
DwtNpageDecCallback	XmNpageDecrementCallback
DwtNpageInc	XmNpageIncrement
DwtNpageIncCallback	XmNpageIncrementCallback
DwtNpixmap	XmNlabelPixmap
DwtNpixmapOff	XmNlabelPixmap
DwtNpixmapOn	XmNselectPixmap
DwtNprompt	XmNpromptString
DwtNpullingCallback	XmNcascadingCallback
DwtNresize	XmNlistSizePolicy (XmList)
DwtNresize	XmNresizePolicy (XmBulletinBoard)
DwtNselectedItemsCount	XmNselectedItemCount
DwtNselectionLabel	XmNselectionLabelString
DwtNshadow	XmNshadowThickness
DwtNshape	XmNindicatorType
DwtNshown	XmNsliderSize
DwtNsingleCallback	XmNsingleSelectionCallback
DwtNsingleConfirmCallback	XmNdefaultActionCallback
DwtNsingleSelection	XmNsingleSelectionPolicy
DwtNspacing	XmNlistSpacing
DwtNstyle	XmNdialogStyle
DwtNtextCols	XmNtextColumns
DwtNtitle	XmNdialogTitle (XmBulletinBoard)
DwtNtitle	XmNtitleString (XmScale)
DwtNunitDecCallback	XmNdecrementCallback
DwtNunitIncCallback	XmNincrementCallback
DwtNvalue	XmNcommand (XmCommand)

(continued on next page)

Table A–3 (Cont.) Resource Name Changes

XUI	Motif
DwtNvalue	XmNset (XmToggleButton)
DwtNvalue	XmNtextString (XmSelectionBox)
DwtNvalueChangedCallback	XmNcommandChangedCallback (XmCommand)
DwtNvisibleItemsCount	XmNvisibleItemCount (XmList)
DwtNvisibleItemsCount	XmNlistVisibleItemCount (XmSelectionBox)
DwtNyesCallback	XmNokCallback
DwtNyesLabel	XmNokLabelString

A.1.4 Enumeration Literal Names

Table A–4 summarizes the differences between the XUI enumeration literal names and the OSF/Motif enumeration literal names. Some XUI enumeration literal names have multiple Motif enumeration literal names. To help you determine which Motif enumeration literal name applies to your widget, the widget class is listed in parentheses after the Motif name.

Table A–4 Enumeration Literal Name Changes

XUI	Motif
Dwt*AaaaAaaa*	Xm*AAAA_AAAA*[1]
DwtAttachAdb	XmATTACH_FORM
DwtAttachOppAdb	XmATTACH_OPPOSITE_FORM
DwtAttachOppWidget	XmATTACH_OPPOSITE_WIDGET
DwtCancelButton	XmDIALOG_CANCEL_BUTTON
DwtCString	XmSTRING
DwtMenuPackingColumn	XmPACK_COLUMN
DwtMenuPackingNone	XmPACK_NONE
DwtMenuPackingTight	XmPACK_TIGHT

[1]Most of the name changes follow this form. The table lists those enumeration literal name changes that do not follow this form.

(continued on next page)

Table A–4 (Cont.) Enumeration Literal Name Changes

XUI	Motif
DwtMenuWorkArea	XmWORK_AREA
DwtModal	XmDIALOG_APPLICATION_MODAL
DwtModal	XmDIALOG_FULL_APPLICATION_ MODAL
DwtModal	XmDIALOG_SYSTEM_MODAL
DwtModeless	XmDIALOG_MODELESS
DwtOrientationHorizontal	XmHORIZONTAL
DwtOrientationVertical	XmVERTICAL
DwtOval	XmONE_OF_MANY
DwtRectangular	XmN_OR_MANY
DwtResizeFixed	XmRESIZE_NONE (XmBulletinBoard)
DwtResizeFixed	XmCONSTANT (XmList)
DwtResizeGrowOnly	XmRESIZE_GROW (XmBulletinBoard)
DwtResizeGrowOnly	XmVARIABLE (XmList)
DwtResizeShrinkWrap	XmRESIZE_ANY (XmBulletinBoard)
DwtResizeShrinkWrap	XmVARIABLE (XmList)
DwtWorkArea	XmDIALOG_WORK_AREA
DwtYesButton	XmDIALOG_OK_BUTTON

A.1.5 Callback Reason Names

Table A–5 summarizes the differences between the XUI callback reason names and the OSF/Motif callback reason names. Some XUI callback reason names have multiple Motif callback reason names. To help you determine which Motif callback reason name applies to your widget, the widget class is listed in parentheses after the Motif name.

Table A–5 Callback Reason Names

XUI	Motif
DwtCRA*aaaAaaa*	XmCR_*AAAA_AAAA*[1]
DwtCRActivate	XmCR_OK (XmSelectionBox)
DwtCRActivate	XmCR_CASCADING (XmCascadeButton)
DwtCRExtend	XmCR_EXTENDED_SELECTION
DwtCRHelpRequested	XmCR_HELP
DwtCRLostFocus	XmCR_LOSING_FOCUS
DwtCRPageDec	XmCR_PAGE_DECREMENT
DwtCRPageInc	XmCR_PAGE_INCREMENT
DwtCRSingle	XmCR_SINGLE_SELECT
DwtCRSingleConfirm	XmCR_DEFAULT_ACTION
DwtCRUnitDec	XmCR_DECREMENT
DwtCRUnitInc	XmCR_INCREMENT
DwtCRValueChanged	XmCR_COMMAND_CHANGED (XmCommand)
DwtCRYes	XmCR_OK

[1]Most of the name changes follow this form. The table lists those callback reason name changes that do not follow this form.

A.1.6 Compound Strings

Table A–6 summarizes the differences between the XUI compound string names and the OSF/Motif compound string names.

Although the compound string names are changed, some functions change the order and number of arguments. See the *OSF/Motif Programmer's Reference* to verify the arguments.

Table A–6 Compound String Names

XUI	Motif
DwtCompString	XmString
DwtCSbytecmp	XmStringByteCompare
DwtCSempty	XmStringEmpty
DwtCSString	XmStringSegmentCreate[1]
DwtCStrcat	XmStringConcat
DwtCStrcpy	XmStringCopy
DwtCStrlen	XmStringLength
DwtCStrncat	XmStringNConcat
DwtCStrncpy	XmStringNCopy
DwtDisplayCSMessage	[2]
DwtDisplayVMSMessage	[2]
DwtGetNextSegment	XmStringGetNextSegment
DwtInitGetSegment	XmStringInitContext
DwtLatin1String	XmStringCreateSimple[1]
DwtString	XmStringSegmentCreate[1]

[1] Suggested replacement only.

[2] No equivalent in Motif.

A.1.7 Fontlist Names

Table A–7 summarizes the differences between the XUI fontlist names and the OSF/Motif fontlist names.

Table A–7 Fontlist Names

XUI	Motif
DwtAddFontList	XmFontListAdd
DwtCreateFontList	XmFontListCreate

A.1.8 Clipboard Names

Table A–8 summarizes the differences between the XUI clipboard names and the OSF/Motif clipboard names.

Table A–8 Clipboard Names

XUI	Motif
DwtBeginCopyToClipboard	XmClipboardStartCopy
DwtCancelCopyFormat	XmClipboardWithdrawFormat
DwtCancelCopyToClipboard	XmClipboardCancelCopy
DwtCopyFromClipboard	XmClipboardRetrieve
DwtCopyToClipboard	XmClipboardCopy
DwtEndCopyFromClipboard	XmClipboardEndRetrieve
DwtEndCopyToClipboard	XmClipboardEndCopy
DwtInquireNextPasteCount	XmClipboardInquireCount
DwtInquireNextPasteFormat	XmClipboardInquireFormat
DwtInquireNextPasteLength	XmClipboardInquireLength
DwtListPendingItems	XmClipboardInquirePendingItems
DwtReCopyToClipboard	XmClipboardCopyByName
DwtStartCopyFromClipboard	XmClipboardStartRetrieve
DwtStartCopyToClipboard	XmClipboardStartCopy
DwtUndoCopyToClipboard	XmClipboardUndoCopy

A.1.9 Resource Manager Names

Table A–9 summarizes the differences between the XUI resource manager names and the OSF/Motif resource manager names.

Table A–9 Resource Manager Names

XUI Resource Manager Name	Motif Resource Manager Name
DwtCloseHierarchy	MrmCloseHierarchy
DwtDrmFreeResourceContext	[1]
DwtDrmGetResourceContext	[1]
DwtDrmHGetIndexedLiteral	[1]
DwtDrmRCBuffer	[1]
DwtDrmRCSetType	[1]
DwtDrmRCSize	[1]
DwtDrmRCType	[1]
DwtFetchColorLiteral	MrmFetchColorLiteral
DwtFetchIconLiteral	MrmFetchIconLiteral
DwtFetchInterfaceModule	MrmFetchInterfaceModule
DwtFetchLiteral	MrmFetchLiteral
DwtFetchSetValues	MrmFetchSetValues
DwtFetchWidget	MrmFetchWidget
DwtFetchWidgetOverride	MrmFetchWidgetOverride
DwtInitializeDRM	MrmInitialize
DwtOpenHierarchy	MrmOpenHierarchy
DwtRegisterClass	MrmRegisterClass
DwtRegisterDRMNames	MrmRegisterNames

[1] No equivalent in Motif. Use MrmFetchLiteral, MrmFetchIconLiteral, or MrmFetchColorLiteral.

A.2 Terminology

Table A–10 lists terminology differences between XUI and OSF/Motif.

Table A–10 Terminology Differences Between XUI and Motif

XUI	Motif
Dialog box, modal	Dialog box, primary application modal, application modal or system modal
Direct manipulation Interface	Graphical User Interface
End box (in a dialog box)	Command line (in a dialog box)
Exit (menu item)	Exit (menu item). Unlike XUI Exit, you are prompted for whether you want to save the file.
Ghost	No equivalent
Hierarchical dialog boxes	Secondary windows. XUI does not talk about secondary windows, and OSF does not talk about hierarchical dialog boxes
Icons	Icons or minimized windows
Maximum sliders, minimum sliders	Sliders (no distinction)
No equivalent	Maximize
No equivalent	Stepper buttons
No equivalent	Sash (window sash)
Option box	Option menu
Pointer speed	Gain
Push to back	Lower
Quit (menu item)	Exit (menu item). You are prompted for whether you want to save the file.
Radio icons	No equivalent
Scales, scroll bars, sliders	Valuators (Includes scales, scroll bars, and sliders)
Shrink to icon	Minimize

(continued on next page)

Table A–10 (Cont.) Terminology Differences Between XUI and Motif

XUI	Motif
Stepping arrows	Stepper arrows
Submenu	Cascading menu
Terminal Screen	Work space
Text Entry Field	Entry box
Text insertion character	Insertion cursor
Toggle button	Check button
Work area	Client area

A.3 Windows and Window Managers

Table A–11 lists differences between XUI and OSF/Motif windows and window managers.

Table A–11 Differences Between XUI and OSF/Motif Windows and Window Managers

XUI	Motif
Does not have a root menu	Has a root menu (a menu that pops up in the root window when you press the select button in blank area of the root window).
Shrink-to-icon button is in upper left	Shrink-to-icon (minimize) button is the left-hand button of the two buttons in the upper right
Does not have a window menu	Has a window menu (a menu that pops up in the window when you press the menu button).
Has a resize button in the far right	Has a resize border and resize handles
The default window manager has an icon box	Default window manager does not have an icon box.
The text label in the title bar is left-justified	The text label in the title bar is centered.

(continued on next page)

Table A–11 (Cont.) Differences Between XUI and OSF/Motif Windows and Window Managers

XUI	Motif
Only has explicit focus policy	Has both explicit (pointer) focus and implicit focus
Has a push-to-back button	Has a Lower menu item

A.4 Menus and Menu Items

Table A–12 lists the differences between the XUI and OSF/Motif window menu items.

Table A–12 Motif Window Menu Items and Functions

Menu Item	Motif Function	XUI Equivalent
Restore	Returns a window to original size after it has been iconified or enlarged	Resize button
Move	Changes the location of a window	Press and drag on the title bar
Size	Changes the size of a window	Resize button
Minimize	Shrinks the window to an icon	Shrink-to-icon button
Maximize	Enlarges the area to cover the whole screen	Resize button
Lower	Sends a window to the back or bottom of the window stack	Push-to-back button
Close	Closes a window and removes it from the workspace	

A.4.1 Menu Bar and Standard Menus

Table A–13 lists the standard menus in each menu bar, and describes the differences between XUI and OSF/Motif. XUI uses dotted lines as separators; Motif uses solid lines.

Table A–13 Differences Between the Motif and XUI Menus in the Menu Bar

XUI Menu	Motif Menu	Explanation
File	File	Mainly the same menu items
Edit	Edit	Mainly the same menu items
	View	Some XUI applications have a View menu
Customize	Options	OSF/Motif provides no specific menu items; the menu items are application-specific.
Font		No equivalent in Motif
Help	Help	Menu items have different names, some similar functions

A.4.2 File Menu Items

Table A–14 lists the File menu items and describes the differences between XUI and OSF/Motif.

Table A–14 Differences Between File Menu Items

Menu Item	XUI	Motif
New	Creates an empty copy of window, does not affect the previous window.	Clears the existing window, does not provide a new window. To continue to have the XUI "new" capability, include a check button in your File Selection box labeled "Open in New Window."
Open. . .	Generates a dialog box that allows users to open an existing file	Same in Motif
Include	Generates a dialog box that allows users to add the contents of a specified file	Not standard in Motif, but use it if appropriate
Revert	Generates a dialog box that allows users to erase current work and revert to last saved version of file	Not standard in Motif, but use it if appropriate
Print	Prints the current file using the current settings of a Print dialog box without displaying the box	Exists in Motif; in Motif, Print covers Print... as well.
Print...	Generates a Print dialog box that allows users to set printing parameters and print the current file	Not standard in Motif, but Motif Print menu item pops up a dialog box if printing information is required.
Quit	Shuts down application; prompts for saving if current version has not been saved	Does not exist in Motif

(continued on next page)

Table A–14 (Cont.) Differences Between File Menu Items

Menu Item	XUI	Motif
Close	Closes the window, leaving the other windows in the application	Removes the primary and associated secondary windows from the workspace, in applications that have more than one primary window. Closing the last primary window of an application causes the application to exit. If data will be lost, the application must prompt users to save changes. The Close menu item from the File menu should have the same effect as the Close menu item from the Window menu.
Exit	Saves file and shuts down application	Shuts down application, prompts for saving if current version has not been saved.

A.4.3 Edit Menu Items

Table A–15 lists the edit menu items and describes the differences between XUI and OSF/Motif.

Table A–15 Differences Between Edit Menu Items

Menu Item	XUI	Motif
Undo	Reverses the effects of a previous operation	Same in Motif.
Redo	Redoes an operation after it has been undone	Not in Motif, but you can add it if appropriate.
Cut	Transfers currently selected information to the clipboard and deletes the information from the application	Same in Motif.
Copy	Transfers the current selection to the clipboard without altering the information in the application	Same in Motif.
Paste	Copies information from the clipboard into the application and retains that information in the clipboard.	Same in Motif.
Clear	Deletes the current selection	Same in Motif.
Delete	Not in XUI	Removes selected portion of data from application and compresses the rest of the data to fill the space that the deleted data occupied.
Select All	Selects all the data in the file	Not in Motif, but you can add it if appropriate.

A.4.4 Help Menu Items

The Help menu items are very different between XUI and OSF/Motif. Table A–16 lists the menu items and their use in both XUI and OSF/Motif. For more information on the Motif Help menu items, see *DECwindows Companion for the OSF/Motif Style Guide*.

Table A–16 Differences Between Help Menu Items

Menu Item	XUI	Motif
Overview	Provides general information about the window from which help was requested	Use "On Window" in Motif.
About	Provides the name and version of the application	Use "On Version" in Motif.
Glossary	Provides definitions of terms	Use "On Terms" in Motif. This menu item is not described in the *OSF/Motif Style Guide*, but use it if appropriate.
On Context	Does not exist as a menu item in XUI. In XUI, users press the Help key and any mouse button.	Initiates context-sensitive help
On Help	Does not exist in XUI	Provides information on how to use your application's Help facility.
On Keys	Does not exist in XUI	Provides information about your application's use of function keys, mnemonics, and accelerators.
Index	Does not exist in XUI	Provides an index for all Help information in your application.
Tutorial	Does not exist in XUI	Provides access to your application's tutorial.

A.5 Standard Message Boxes

Motif message boxes often have a Help push button in the lower right corner.

A.6 Mouse Buttons Behavior

Table A–17 provides a list of differences between the XUI and Motif mouse button behavior.

Table A–17 Differences in Mouse Buttons

Mouse Button	XUI	Motif
MB1	Used for selection	Used for selection. Called the Select button.
MB2	Used to display pop-up menus	Used for direct manipulation of objects and other application-specific needs. Called the Menu button.
MB3	Used for application-specific needs, and for Copy To and Copy From operations, if your application supports them	Used to dislay pop-up menus. Called the Custom button.

B

Porting Filter Summary

This appendix describes the porting filters. Table B–1 lists the name and function of the nine porting filters for VMS systems, in the order in which they are executed by the command procedure DECW$DXM_PORT.COM. Table B–2 lists the name and function of the nine porting filters for ULTRIX systems, in the order in which they are executed by the script file DXm_port.

Table B–1 Porting Filters: Names and Functions (VMS)

Filter Name	What It Does
DECW$DXM_PORT_RESOURCES	Transforms Toolkit resource names
DECW$DXM_PORT_CALL	Transforms callback references and reasons
DECW$DXM_PORT_HILEVELS	Replaces high-level entry points with low level
DECW$DXM_PORT_LOLEVELS	Transforms low-level entry point names
DECW$DXM_PORT_CLASS	Transforms Toolkit class references
DECW$DXM_PORT_DATA	Transforms Toolkit data types
DECW$DXM_PORT_DRM	Transforms DRM references
DECW$DXM_PORT_INCLUDES	Transforms Toolkit include file names
DECW$DXM_PORT_UIL	Transforms UIL source

Table B–2 Porting Filters: Names and Functions (ULTRIX)

Filter Name	What It Does
DXm_port_resources	Transforms Toolkit resource names
DXm_port_call	Transforms callback references and reasons
DXm_port_hilevels	Replaces high-level entry points with low level
DXM_port_lolevels	Transforms low-level entry point names
DXm_port_class	Transforms Toolkit class references
DXm_port_data	Transforms Toolkit data types
DXm_port_drm	Transforms DRM references
DXm_port_includes	Transforms Toolkit include file names
DXm_port_uil	Transforms UIL source

C

Intrinsics Routine Summary

Table C–1 lists the Intrinsics routines you would use if you used the default application context and the routines you would use if you explicitly named an application context.

If your application uses an explicit application context, your custom widgets must support this context. See the *X Window System Toolkit: A Complete Programmer's Guide and Specification* for more information on specifying an application context.

Table C–1 Default and Explicit Application Context Routines

If using a default application context, use these routines:	If using explicit application context, use these routines:
XtAddActions	XtAppAddActions
XtAddInput	XtAppAddInput
XtAddTimeout	XtAppAddTimeout
XtAddWorkProc	XtAppAddWorkProc
XtCreateApplicationShell	XtAppCreateShell
XtError	XtAppError
XtErrorMsg	XtAppErrorMsg
XtGetErrorDatabase	XtAppGetErrorDatabase
XtGetErrorDatabaseText	XtAppGetErrorDatabaseText
XtGetSelectionTimeout	XtAppGetSelectionTimeout
XtInitialize	XtAppInitialize
XtMainLoop	XtAppMainLoop
XtNextEvent	XtAppNextEvent

(continued on next page)

Table C–1 (Cont.) Default and Explicit Application Context Routines

If using a default application context, use these routines:	If using explicit application context, use these routines:
XtPeekEvent	XtAppPeekEvent
XtPending	XtAppPending
XtProcessEvent	XtAppProcessEvent
XtSetSelectionTimeout	XtAppSetSelectionTimeout
XtSetErrorHandler	XtAppSetErrorHandler
XtSetErrorMsgHandler	XtAppSetErrorMsgHandler
XtSetWarningHandler	XtAppSetWarningHandler
XtSetWarningMsgHandler	XtAppSetWarningMsgHandler
XtWarning	XtAppWarning
XtWarningMsg	XtAppWarningMsg

Index

A

Accelerators, 3–32, 3–39
Application
  adding mnenomics, 3–39
  comments in, 3–3
  compatibility with window managers, 1–4
  compiling (ULTRIX RISC), 3–35
  compiling (ULTRIX VAX), 3–36
  compiling (VMS), 3–34
  formatting, 3–3
  linking (ULTRIX RISC), 3–35
  linking (ULTRIX VAX), 3–36
  linking (VMS), 3–34
  testing, 3–36
  updating build procedures, 4–1
  updating documentation, 4–1
Application context
  specifying in application, 3–31, C–1
Application shell
  naming, 3–31
Argument names
  in comments, 3–3
  in UIL files, 3–9
Arguments
  effect of filters on, 3–8

B

Bit gravity
  inherited from superclass, 3–13, 3–21
Build procedures, 4–1
Bulletin board widget, 3–12
Button bindings, 3–38

C

Callback
  expose, 3–13
  incremental selection, 3–33
  in list widget, 3–14
  resize, 3–13
  selection, 3–32
  specifying, 3–33
Callback list
  reference to a, 3–5
Callback reason
  differences between XUI and Motif, A–10
Callback record
  referencing a, 3–22 to 3–23
Callback structure
  command widget, 3–24
  drawing area widget, 3–24
  file selection box widget, 3–25
  list widget, 3–26
  row column widget, 3–26
  selection box widget, 3–27
  toggle button widget, 3–27
Character set
  specifying, 3–8
Class record
  for primitive widgets, 3–20
Client
  relationship with MWM, 3–37
Clipboard
  differences between XUI and Motif, A–12

Color mixing widget
  including in your application, 3–18
Command widget, 3–13
  callback structure, 3–24
Command window widget
  see Command widget
Comments
  effect of filters on, 3–3
  in UIL files, 3–9
Composite widget, 3–20
Compound string
  creating a, 3–9
  differences between XUI and Motif, A–11
  freeing resources, 3–9, 3–28
  retrieving a, 3–28
  usage, 3–9, 3–28
Compound string widget
  including in your application, 3–18
Context-sensitive pop-up menus, 3–41

D

Defaults files, 3–34
Developer decision message, 3–2
Dialog box widget
  see Bulletin board widget
Differences
  betweeen XUI and Motif, A–1
Documentation
  updating, 4–1
Drawing area widget, 3–13
  callback structure, 3–24

E

Enumeration literal
  differences between XUI and Motif, A–8
Event
  expose, 3–13
  handlers, 3–33
  resize, 3–13
Expose event, 3–13

F

Field name changes, 3–24 to 3–28
Files
  header, 3–9 to 3–10, 3–33
  include, 3–9 to 3–10, 3–33
  listing differences between, 3–1
  required for compiling (ULTRIX RISC), 3–35
  required for compiling (ULTRIX VAX), 3–36
  xdefaults, 3–34
File selection box widget, 3–13
  callback structure, 3–25
File selection widget
  see File selection box widget
Filters
  converting high-level XUI subroutine calls, 3–6
  interpreting messages, 2–3, 3–2 to 3–3
  names, B–1
  names not changed, 3–3
  preparation, 2–1
  reviewing output from, 3–1 to 3–9
  running on ULTRIX systems, 2–3
  running on VMS systems, 2–2
Focus
  keyboard, 3–21, 3–38
  policy, 3–38
  tab groups, 3–40
Font list
  differences between XUI and Motif, A–12
Font units, 3–29
Formatting
  effect of filters on, 3–3
Function names
  differences between XUI and Motif, A–3

G

Gadget, 3–20
  as a child of a widget class, 3–33

Gravity, 3–13, 3–21

H

Header files, 3–33
Help push button, A–22
Help widget
  including in your application, 3–18
Hierarchy
  widget class, 3–20
High-level conversions, 3–6
HyperHelp
  description, 3–18

I

ICCCM, 3–37
Include files
  changes between XUI and Motif, 3–33
  required in application, 3–9 to 3–10, 3–33, 3–35, 3–36
Instance fields, 3–33
Intrinsics
  application context routines, 3–31, C–1
  application shell names, 3–31
  changes to routine interfaces, 3–31 to 3–32
  changes to widget semantics, 3–32 to 3–33
  reviewing changes, 3–30

K

Key bindings, 3–38
Keyboard accelerator
  actions on insensitive widgets, 3–32
  adding to an application, 3–39
Keyboard focus, 3–21, 3–38
Keyboard traversal
  adding to an application, 3–40

L

Libraries
  required for linking (ULTRIX RISC), 3–35
  required for linking (ULTRIX VAX), 3–36
Linking applications
  on ULTRIX systems (RISC), 3–35
  on ULTRIX systems (VAX), 3–36
  on VMS systems, 3–34
List box widget
  see List widget
List widget, 3–14
  callback structure, 3–26

M

Main window widget, 3–14
Mapping unknown message, 3–2
Mapping unsupported message, 3–2
Memory
  freed in an application, 3–28
  freeing, 3–9
Menu
  accelerator, 3–39
  closing the window from the, 3–38
  edit, A–20
  file, A–18
  help, A–21
  mnemonics, 3–39
  pop-up, 3–41
  pull-down, 3–39
  standard, A–17
  window, A–16
Menu bar, A–17
Menu widget
  see row column widget, 3–15
Message box, A–22
Message box widget, 3–14
Messages
  from porting filters, 3–2 to 3–3
Mnemonic
  adding to an application, 3–39

Motif applications
compatibility with XUI window manager, 1–4
Motif Resource Manager (MRM)
initializing, 3–17
Motif terminology, A–14
Motif transition message, 3–2, 3–6
Motif widget
summary, A–2
Motif Window Manager
relationship with clients, 3–37
Mouse button bindings, 3–38, A–22
MWM
see Motif Window Manager

N

Name changes
callback reasons, A–10
clipboard, A–12
compound strings, A–11
enumeration literals, A–8
font list, A–12
functions, A–3
not made by filters, 3–3
resource manager, A–13
resources, A–5
widget classes, A–2

P

Porting filters
see filters
Positioning of widgets, 3–29
Primitive widget
class record initialization, 3–20
superclass, 3–20
Print widget
including in your application, 3–19
Pull-down menu
keyboard accelerator use, 3–39
mnemonic use, 3–39
Push button widget, 3–15

R

Readability
checking, 3–3
Release notes, 4–1
Resize event
notifying the application, 3–13
Resolution independence, 3–29
Resource manager
differences between XUI and Motif, A–13
Resource names
differences between XUI and Motif, A–5
Return values
effect of filters on, 3–7
Row column widget, 3–15
callback structure, 3–26

S

Scale widget
setting values, 3–16
Scroll bars, 3–29
Scroll bar widget, 3–16
see also scroll bars
Scrolled list widget, 3–29
Scrolled text widget, 3–30
Scrolled window widget, 3–16
Selection box widget
callback structure, 3–27
Separator widget, 3–17
Shareable images
required for linking (VMS), 3–34
Simple text widget
see Text widget
Structured Visual Navigation widget
including in your application, 3–19
Style
help push button, A–22
Motif requirements, 1–3, 3–38
Subroutines
see XUI routines
Superclass, 3–20

T

Tab group, 3–40
Testing, 3–36 to 3–37
Text widget, 3–17
Toggle Button widget
  callback structure, 3–27
Traversal, 3–40

U

UIL
  converting application to, 2–1
  effect of filters on argument names, 3–9
  effect of filters on comments, 3–9
  initializing DXm, 3–17
  initializing MRM, 3–17
  running the compiler (ULTRIX RISC), 3–35
  running the compiler (ULTRIX VAX), 3–36
  running the compiler (VMS), 3–34

W

Widget
  see also XUI widget
  color mixing, 3–18
  composite, 3–20
  compound string, 3–18
  custom, 3–20 to 3–21
  Digital Extended Motif, 3–17 to 3–19
  help, 3–18
  instance initialization, 3–33
  layout, 3–30
  modal, 1–4
  passing parameters to a, 3–33
  positioning of, 3–29
  primitive, 3–20
  print, 3–19
  scrolled list, 3–29
  scrolled text, 3–30
  semantics, 3–32
  Structured Visual Navigation, 3–19
Widget (cont'd)
  XmBulletinBoard, 3–12
  XmCommand, 3–13
  XmDrawingArea, 3–13
  XmFileSelectionBox, 3–13
  XmList, 3–14
  XmMainWindow, 3–14
  XmMessageBox, 3–14
  XmPushButton, 3–15
  XmRowColumn, 3–15
  XmScale, 3–16
  XmScrollBar, 3–16
  XmScrolledWindow, 3–16
  XmSeparator, 3–17
  XmText, 3–17
Widget class
  differences between XUI and Motif, A–2
  hierarchy, 3–20
  use of gadget children, 3–33
Window menu
  close function, 3–38
Window widget
  see Drawing area widget

X

XmStringFree routine, 3–5, 3–9, 3–28
XUI files
  converting, 2–1
XUI routines
  converting high-level, 3–6
XUI terminology, A–14
XUI widget
  see also widget
  DwtCommandWindow, 3–13
  DwtDialogBox, 3–12
  DwtFileSelection, 3–13
  DwtListBox, 3–14
  DwtMainWindow, 3–14
  DwtMenu, 3–15
  DwtMessageBox, 3–14
  DwtPushButton, 3–15
  DwtScale, 3–16
  DwtScrollBar, 3–16
  DwtScrollWindow, 3–16

XUI widget (cont'd)
DwtSeparator, 3–17
DwtSText, 3–17
DwtWindow, 3–13
summary, A–2
XUI window manager
compatibility with Motif applications, 1–4