Motif® 2.1
Porting Guide

Document Number 007-3951-001

CONTRIBUTORS

Written by Richard Offer
Production by Linda Rae Sande
Engineering contributions by Richard Offer and Richard Hess.
St. Peter's Basilica image courtesy of ENEL SpA and InfoByte SpA. Disk Thrower image courtesy of Xavier Berenguer, Animatica.

Motif® 2.1 Porting Guide
Document Number 007-3951-001

Contents

List of Examples

List of Figures

List of Tables

About This Guide

This document discusses the new Motif feature release. It is not intended to teach Motif programming; instead it is aimed at current Motif 1.2 developers who wish to port their code to the new API.

What You Need to Know About the Libraries

Not all libraries supplied by Silicon Graphics are available in Motif 2.1 versions. The following libraries are available in Motif 2.1 compatible versions:

- Media Warehouse
- Digital media
- Impressario
- GL Drawing area widget

  Because this is available in source, only a single archive library is shipped. It includes both the 1.2 and 2.1 objects.
- WebViewer
- ViewKit

Versions of the above libraries released before Irix 6.5.2 are known to cause problems. Installing the 6.5.2 versions should fix this.

The following are known problems with Motif 2.1 in IRIX 6.5.2:

- Display PostScript does not work.
- sgitcl does not work.
- The custom libraries that the Workshop Debugger, cvd, uses are not available.
- The libraries are not completely corded.

Motif 2.1 does not include a library for the O32 ABI. Any applications that are not yet using the n32 ABI must switch to that ABI before they can use Motif 2.1.

Printing Support

Motif 2.1 supports a print widget for interaction with the X Print Server that comes with X11R6 (and later). The print server and the print widget are not available in this release.

Multithreading

Motif 2.1 was announced as being thread safe. However, tests run by Silicon Graphics show that this is not necessarily the case.

Road Map

In the next major release of IRIX, Silicon Graphics will be making Motif 2.1 the default build environment (while continuing to support Motif 1.2 development much as 2.1 is supported currently).

Later still, the capability to develop with Motif 1.2 will be removed, but the 1.2 execution environment will continue to be shipped.

Execution Environment

The execution environments for all Motif 2.1-supported software (Motif, ViewKit, WebViewer, and so on) are combined, so installing any of the above EOE images from any release, 6.5.2 and later, will provide you with all the shared libraries you need to run applications that have been developed with either Motif 1.2 or 2.1.

To determine which release of IRIX you have, use the following command:

```
/bin/uname -R
```

Additional Reading

Motif 2.1 Programmer's Guide, The Open Group, 1997.

Motif 2.1 User's Guide, The Open Group, 1997.

Motif 2.1 Widget Writer's Guide, The Open Group, 1997

Motif 2.1 Glossary, The Open Group, 1997.

Motif and Common Desktop Environment: Style Guide, The Open Group, 1997.

Motif and Common Desktop Environment: Style Guide Certification Checklist, The Open Group, 1997.

Motif and Common Desktop Environment: Style Guide Reference, The Open Group, 1997.

Most of these books are also available using the IRIS InSight viewer.

Chapter 1

Developing With Motif 2.1

At the start of the Motif 2.1 project, Silicon Graphics made the decision to support multiple (major) releases of the IRIX Interactive Desktop development environment installed on the system at the same time. Using the framework that was added as part of IRIX 6.5, it is now possible to develop both Motif 1.2 and 2.1 versions of an application side by side (subject of course to application build requirements). This allows you to move to Motif 2.1 at the rate that is most suited to your own needs.

The framework takes all version-specific files out of */usr/include*, */usr/bin/X11*, */usr/lib32*, and */usr/lib64* and places them under */usr/Motif-1.2* or */usr/Motif-2.1*, as appropriate.

Default Build Environment

To allow for normal (default) compilation, there are symbolic links that point from the locations used prior to IRIX 6.5 to the correct version of the installed Motif development environment.

Note: The version-stamped DSOs have not been moved and still reside in */usr/lib32* and */usr/lib64*).

During the installation process, a shell script (*/usr/Motif-1.2/lib/mksymlinks*) is executed that sets the default build environment. The default environment is Motif 1.2 and will remain so until the next major release of IRIX.

If you need to change the default environment to 2.1, enter (as root) the following command:

```
# /usr/Motif-2.1/lib/mksymlinks
```

To return the default environment to 1.2, enter:

```
# /usr/Motif-1.2/lib/mksymlinks
```

To check which version is the default, use the following command:

```
ls -ld /usr/include/Xm
```

This command prints out a directory listing that shows to which file */usr/include/Xm* is linked:

```
lrwxr-xr-x /usr/include/Xm -> ../Motif-1.2/include/Xm
```

In the case above, anything compiled on this machine (using the default build environment) will use the Motif 1.2 API.

Caution for Using the Default Build Environment

The main problem with relying on the default build environment is one of reproducibility; anyone (as root) can change the environment (even during a build), and you will not know until you start getting random crashes.

For this reason Silicon Graphics does not recommend that the default build environment be used for production builds, or in environments where there may be some doubt as to a machine's state (large multi-user build machines, for example).

It cannot be stressed enough how important it is to ensure that you do not mix code (either individual *.o* files or even whole libraries) that uses both 1.2 and 2.1. If you do successfully get it to build you will see random crashes for which it is very difficult to determine the cause. It is up to the developers to ensure that they have a consistent environment throughout all stages of the project build.

Explicit Build Environment

An explicit build environment is one where it can be predicted in advance which set of libraries and header files will be used. This is done by making use of the structure under */usr/Motif-1.2* or */usr/Motif-2.1*. For example, if you want to ensure that you are building a Motif 1.2 application, you should add the following to the compile/link line (assuming that you are building an n32 binary):

```
-I/usr/Motif-1.2/include -L/usr/Motif-1.2/lib32
```

This must appear on the command line before */usr/include* and */usr/lib32*. If your build process uses *make* to build its Makefiles, Silicon Graphics has configured *mmkmf* to generate the correct arguments. Do not use *xmkmf*. It is not part of the Motif Software Development Kit and could not be modified. It will generate Makefiles using the default environment.

Imake Tips

To build an application that uses the Motif 1.2 SDK, run */usr/Motif-1.2/bin/mmkmf*. For Motif 2.1, use */usr/Motif-2.1/bin/mmkmf*.

To check which version of Motif an *Imake* generated Makefiles will use, you can page through the Makefile, or try the following:

```
grep XmVersion Makefile
```

This returns a line like the following:

```
# ** XmVersion ** Motif 2.1 Generated Makefile *********************
```

Make Tips

If you use plain Makefiles for building, you will have to devise your own mechanism, but if you use *smake*, you can use the following method:

1. Create a new Makefile (Example 1-1).
2. Include the code in Example 1-2 in every Makefile.

Example 1-1 Sample Makefile (src/motifVer)

```
## This file will append the correct include/lib settings to the various LC*
## variables.

## This line sets the default build to be 1.2. To build against 2.1
## all a Makefile has to do is set MOTIFVERSION to 2.1 *before* this file gets
## included (this could also be done on the command line with
## make MOTIFVERSION=2.1 <target>

MOTIFVERSION ?=1.2

# The rest is automatic....
```

```
MOTIFDIR=/usr/Motif-$(MOTIFVERSION)
MOTIFROOTDIR = $(ROOT)$(MOTIFDIR)

LCINCS += -I$(MOTIFROOTDIR)/include
LC++INCS += -I$(MOTIFROOTDIR)/include
LCXXINCS += -I$(MOTIFROOTDIR)/include
```

Example 1-2 Sample Makefile Code (src/Makefile)

```
#!smake
#
# $Id$
# This is an example Makefile to illustrate how one might construct
# a plain Make(not Imake) based build system that can be switched
# to use either version of Motif.

include /usr/include/make/commondefs

TARGETS=buildDemo

include motifVer

LLDLIBS = -lXm -lXt -lX11

$(TARGETS): vertical-writing.o

demo.o: vertical-writing.c
```

This example is for an *or*-based build, where you need to be able to build both 1.2 and 2.1 versions, but not during the same build. An *and*-based build (where both versions are needed concurrently) will probably create two new subdirectories (say *obj* and *obj21*), with a Makefile in each (each setting the *MOTIFVERSION* as appropriate). And rather than linking the source files, simply refer to them using "../".

Chapter 2

Motif API Changes

Official Open Group Changes

With the introduction of Motif 2.0, the Open Software Foundation, now The Open Group, cleaned up the APIs. Most of the public (that is, intended for application use) API was left as it was, but the private (needed by Widget writers) was completely changed. Before 2.0, there were two types of symbols, public (prefixed by **Xm**), and private (prefixed by **_Xm**). Motif 2.0 introduced a new type, (prefixed by **Xme**), and renamed most of the **_Xm** symbols to be **Xme**. There are still some **_Xm** symbols, but these are internal to the library and are not guaranteed to be exported.

Some symbols (**EditDone**, **EditError**, **EditReject**, **XmTextStatus**) have been moved from a private header file (*Xm/TextStrsoP.h*) to a public one (*Xm/Xm.h*). This can cause some problems with applications and libraries which use those symbols themselves. In these cases, there is a namespace conflict at compile time due to redefinition. The problem can be acute because some of these symbols do not have a **Xm** prefix.

Because these symbols can break existing applications, they have been moved back to their old, private locations as the default. To get the standard OSF behavior (with these symbols in the public header file), define **_SGI_OSF_PUBLIC_SYMBOLS** before any Motif header file is included.

Obsolete Symbols

The Table 2-1 lists symbols that were in Motif 1.2 but are not in 2.1, along with any replacement.

Table 2-1 OSF Obsoleted Symbols

Motif 1.2 Symbol	Motif 2.1 Symbol
_XmOSPutenv()	
_XmGetDefaultTime()	XmeGetDefaultTime()
_XmGetColors()	XmGetColors()[a]
_XmDrawShadows()	XmeDrawShadows()[a]
_XmEraseShadow()	XmeClearBorder()
_XmGetArrowDrawRects()	XmDrawArrow()
_XmOffsetArrow()	XmeDrawArrow()[a]
_XmDrawSquareButton()	
_XmDrawDiamondButton()	XmeDrawDiamond()
_XmDrawShadowType()	XmDrawShadows()[a]
_XmDrawBorder()	XmeDrawHighlight()
_XmMoveObject()	XmeConfigureObject()[a]
_XmResizeObject()	XmeConfigureObject()[a]
_XmVirtualToActualKeysym()	XmeVirtualToActualKeysyms()[a]

a. API has changed.

Silicon Graphics Changes

XPM

Motif 2.*x* makes use of the Xpm library for colored pixmaps (implemented internally to the library). ViewKit 1.2 distributed its own copy of this library as well. Silicon Graphics has consolidated this and now ships a standalone copy of *libXpm.so* as part of the Motif 2.1 images. Documentation, in the form of the original PostScript report and examples, may be found in */usr/share/src/X/motif-2.1/xpm*.

Archive Libraries

With IRIX 6.5, Silicon Graphics moved the Motif archive libraries (that is, *libXm.a*) out of the default installation images. With Motif 2.1, archive libraries are no longer shipped; all the required libraries are shipped as shared objects (DSOs) in the standard execution environment.

Consistent APIs

Considerable effort has been made toward tidying up the various Silicon Graphics APIs. Some symbols have been renamed to better highlight which APIs have been added by Silicon Graphics. In the past, some of these symbols were mistakenly prefixed with **Xm**. These have now been renamed to have a prefix of **Sg**. Table 2-2 lists these symbols.

To help in porting older code that may have used these APIs, the old symbol names can be activated by defining **_SGIMOTIF_OBSOLETE_API** before any Motif header file is included. By default, this flag is not enabled.

Table 2-2 New Names for Silicon Graphics Symbols

Silicon Graphics Motif 1.2 Symbol	Silicon Graphics Motif 2.1 Symbol
XmNmenuBarRepeatTimeout	SgNmenuBarRepeatTimeout
XmCMenuBarRepeatTimeout	SgCMenuBarRepeatTimeout
XmNarrowsAdjacent	SgNarrowsAdjacent
XmCarrowLayout	SgCarrowLayout
XmNarrowType	SgNarrowType
XmRArrowType	SgRArrowType
XmRIndPixel	SgRIndPixel
XmRSelectPixel	SgRSelectPixel
XmRArrowType	SgRArrowType
XmNindicatorBackground	SgNindicatorBackground
Xm3D_ONE_OF_MANY	Sg3D_ONE_OF_MANY
Xm3D_N_OF_MANY	Sg3D_N_OF_MANY
XmNOT_ADJACENT	SgNOT_ADJACENT
XmLEFT_ADJACENT	SgLEFT_ADJACENT
XmRIGHT_ADJACENT	SgRIGHT_ADJACENT
XmTOP_ADJACENT	SgTOP_ADJACENT
XmBOTTOM_ADJACENT	SgBOTTOM_ADJACENT
XmNORMAL_ARROWS	SgNORMAL_ARROWS
XmROTATE_ARROWS	SgROTATE_ARROWS
XmNO_THUMB_WHEN_EMPTY	SgNO_THUMB_WHEN_EMPTY

New #define Statements

Table 2-3 lists three basic **#define** statements used in the various libraries and applications. They are listed here primarily for information value. The only one that can be turned off is _SGIMOTIFAPI. _SGIMOTIF and _SGIBUGFIX should never be turned off.

Table 2-3 Silicon Graphics #define Statements

Statement	Effect
#define _SGIMOTIF	All Silicon Graphics look and feel changes. This should not be turned off at application compile time, because internal structures will be incorrectly sized. **NOTE:** If _SGIMOTIF is defined, the application should be able to count on the full Silicon Graphics look and feel (that is, Schemes/desktop integration, and so forth). However, because this has not been fully tested, be cautious in making this assumption.
#define _SGIBUGFIX	A change that has either been sent to TOG for inclusion into the standard source base, or a change that is required for the Silicon Graphics build environment that is not directly related to the look and feel. This is also something that should not be touched by an application.
#define _SGIMOTIFAPI	Wraps extensions to the Motif API. This can be turned off by an application program by using the following statement before any Motif headers are included: `#define _NO_SGIMOTIFAPI` **Note:** Using the `#undefine` statement with _SGIMOTIFAPI does not work.

New Resource: SgNshowVersion

To further help in porting, Silicon Graphics has added a new resource **SgNshowVersion** (type **`Boolean`**). If **SgNshowVersion** is True (the default is False), a message will be printed out to *stdout* indicating that the application is linking against Motif version 2.1 library. The following message will be printed out:

```
SGI Irix Indigo Magic 2.1, based on OSF/Motif 2.1.10
```

New Widgets

For a quick introduction to the new widgets, compile and run either *sampler2_0* demo (in */usr/share/src/X/motif-2.1/osf_demos/programs/sampler2_0*) or *periodic* (in */usr/share/src/X/motif-2.1/osf_demos/programs/periodic*). Each shows the new widgets, and how they can be used. (Sampler2_0 is probably preferable, because it is straight C code, rather than a mix of C and UIL.)

The following sections give a brief overview of the new widgets:

- "Notebook"
- "Container"
- "Combobox"
- "SpinBox"

Notebook

The Notebook is the Motif version of a tabbed-dialog (in Windows terms) widget. It is a generic solution, where any widget that holds the **XmQactivatable** trait can be used as the button. Currently, **XmPushButton**, **XmDrawnButton**, **XmArrowButton**, **XmPushButtonGadget**, and **XmArrowButtonGadget** hold this trait.

Container

The container is a layout widget suitable for use as an outliner (where it is possible to expand or contract individual items). It may also be used as a base on which to place icons that can then be dragged around.

The file */usr/share/src/X/motif-2.1/osf_demos/programs/sampler2_0* shows an example of how to use this widget.

Combobox

This is a drop down list. It may be configured to be editable or not.

SpinBox

A combination of a text field and arrow buttons, a SpinBox is commonly used to step through an array of known values (for instance, the months of the year).

Notable Changes to Existing Widgets

The following sections briefly describe changes to existing widgets.

Toggle Button

The toggle button has a new resource **XmNtoggleMode** to allow the selection of a simple on/off toggle (**XmTOGGLE_BOOLEAN**) or a three state on/off/not set toggle (**XmTOGGLE_INDETERMINATE**).

Scale

The scale has a new visual mode, **XmNslidingMode**, which allows a thermometer type look (using **XmTHERMOMETER**) as well as the existing slider look (**XmSLIDER**).

New Features

Vertical Writing

IRIX 6.2 (and later) allow an application to use vertical writing (for use by Asian languages). This is set with the **XmNlayoutDirection** resource.

Note: Not all the widgets support **XmNlayoutDirection**; **XmTextField** for example, does not.

The code fragments in Example 2-1 illustrate the use of these widgets. Figure A-2 (in Appendix A) shows the resulting dialog box.

Example 2-1 Sample Code: Vertical Writing

```
/* vertical text widget */

vert = XtVaCreateManagedWidget("vertical",xmTextWidgetClass,form,
            XmNtopAttachment, XmATTACH FORM,
            XmNleftAttachment, XmATTACH FORM,
            XmNbottomAttachment, XmATTACH FORM,
            XmNlayoutDirection, XmTOP TO BOTTOM,
            XmNrows, 10,
            NULL);


/* horizontal text widget */

horiz = XtVaCreateManagedWidget("horizontal",xmTextWidgetClass,form,
            XmNtopAttachment, XmATTACH FORM,
            XmNleftAttachment, XmATTACH WIDGET,
            XmNleftWidget, vert,
            XmNrightAttachment, XmATTACH FORM,
            NULL);
```

POSIX Style Message Catalogs

All of the Motif error messages are now stored in a POSIX style message catalog to allow for easy localization.

Appendix A

Examples

Code Examples

The following are the full listings for two examples. Example A-1 builds a simple notebook widget (shown in Figure A-2). Example A-2 builds an example of vertical writing (shown in Figure A-2).

Example A-1 Simple Notebook

```
/* include init */
#include <Xm/XmAll.h>

#define NUM_PAGES 8
#define APPNAME “Notebook”


void pageChangeCB(Widget w, XtPointer client, XtPointer call);

/* end init */


int
main(int argc, char ** argv)
{
    XtAppContext AppContext;
    Widget     TopLevel = XtVaOpenApplication(&AppContext,
                                APPNAME,
                                NULL, 0,
                                &argc, argv,
                                NULL,
                                sessionShellWidgetClass,
                                NULL);

/* include notebook */
    Widget notebook = XtVaCreateWidget(“notebook”,
                                xmNotebookWidgetClass,
                                TopLevel,
```

```
                            NULL);
int i;
XmString xmstr;
char    labelString[32];

for ( i=0; i< NUM_PAGES; i++ ) {

    sprintf(labelString,”Page %d\n”,i);
    xmstr = XmStringCreateLocalized(labelString);

    (void) XtVaCreateManagedWidget(“label”, xmLabelWidgetClass,
                    notebook,
                    XmNpageNumber, i+1,
                    XmNlabelString, xmstr,
                    NULL);

    XmStringFree(xmstr);
}

for ( i=0; i< NUM_PAGES; i+=4 ) {
    int j;

    sprintf(labelString,”Major\nTab %d\n”,i);
    xmstr = XmStringCreateLocalized(labelString);

    (void) XtVaCreateManagedWidget(“button”,
                    xmPushButtonWidgetClass,
                    notebook,
                    XmNpageNumber, i+1,
                    XmNlabelString, xmstr,
                    XmNnotebookChildType,
                    XmMAJOR_TAB,
                    NULL);
    XmStringFree(xmstr);

    for ( j=i; j<i+4; j++ ) {

        sprintf(labelString,”Minor\nTab %d\n”,j);
        xmstr = XmStringCreateLocalized(labelString);

        (void) XtVaCreateManagedWidget(“button”,
                        xmPushButtonWidgetClass,
                        notebook,
                        XmNpageNumber, j+1,
                        XmNlabelString, xmstr,
```

```
                            XmNnotebookChildType,
                            XmMINOR_TAB,
                            NULL);
            XmStringFree(xmstr);
        }
    }

    XtAddCallback(notebook, XmNpageChangedCallback,
                  (XtCallbackProc) pageChangeCB, NULL);

    XtManageChild(notebook);
/* end notebook */

    XtRealizeWidget(TopLevel);

    XtAppMainLoop(AppContext);

}


/* pageChangeCB - comment */
static void
pageChangeCB(Widget w, XtPointer client, XtPointer call)
{

}
```

Figure A-1 shows the widget that results from the code in Example A-1.

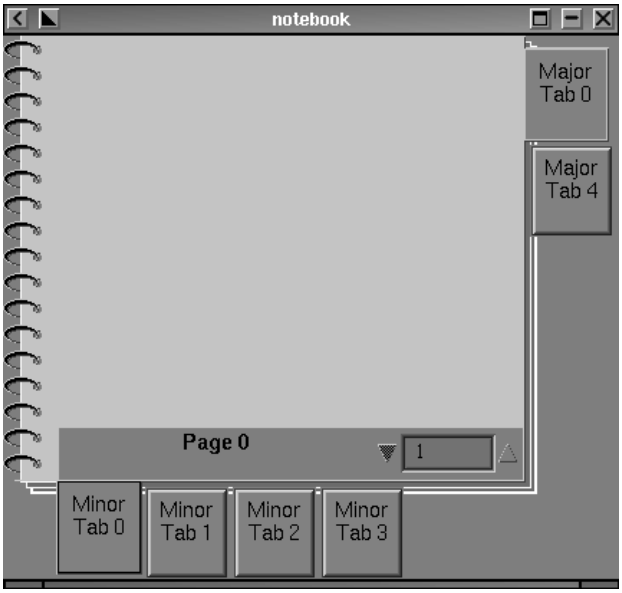

Figure A-1 Simple Notebook Demo

Example A-2 Vertical Writing

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

#define APPNAME “Vertical”

XtAppContext AppContext;
Widget TopLevel;

main(int argc, char ** argv)
{
    Widget form, vert, horiz;

    XtSetLanguageProc(NULL, NULL, NULL);

    TopLevel = XtVaOpenApplication(&AppContext,
                    APPNAME, NULL, 0,
                    &argc, argv,
                    NULL,
                    sessionShellWidgetClass,
                    NULL);

    form = XtVaCreateWidget(“form”, xmFormWidgetClass, TopLevel, NULL);

/* vertical text widget */

    vert = XtVaCreateManagedWidget(“vertical”,
                    xmTextWidgetClass, form,
                    XmNtopAttachment, XmATTACH_FORM,
                    XmNleftAttachment, XmATTACH_FORM,
                    XmNbottomAttachment, XmATTACH_FORM,
                    XmNlayoutDirection,
                    XmTOP_TO_BOTTOM,
                    XmNrows, 10, NULL);

/* horizontal text widget */

    horiz = XtVaCreateManagedWidget(“horizontal”,
                    xmTextWidgetClass, form,
                    XmNtopAttachment, XmATTACH_FORM,
                    XmNleftAttachment,
                    XmATTACH_WIDGET,
                    XmNleftWidget, vert,
                    XmNrightAttachment, XmATTACH_FORM,
                    NULL);
```

```
/* manage form */
    XtManageChild(form);

    XtRealizeWidget(TopLevel);

    XtAppMainLoop(AppContext);
}
```

Figure A-2 shows the results of the code in Example A-2.

Figure A-2 Vertical Writing Demo

Index

O

P

S

X

Tell Us About This Manual

As a user of Silicon Graphics products, you can help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please send the title and part number of the document with your comments. The part number for this document is 007-3951-001.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389